ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ

Η γλώσσα C++ σε βάθος

2η αναθεωρημένη έκδοση

Γνωρίστε τη φιλοσοφία του αντικειμενοστρεφούς προγραμματισμού μέσα από τη γλώσσα C++

Νίκος Μ. Χατζηγιαννάκης

Απαντήσεις ασκήσεων

Περιέχει δωρεάν CD-ROM

Αποσπώμενη Κάρτα Αναφοράς

ΚΛΕΙΔΑΡΙΘΜΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

Ασκήσεις Κεφαλαίου 1

1.1

- ❑ Η C++ είναι μια γλώσσα με αυστηρό έλεγχο.
- ❑ Η C++ συναντάται συνήθως σε ερμηνευτική μορφή.
- ☑ Οι τύποι δεδομένων μπορεί να διαφέρουν σε διαφορετικές γλώσσες προγραμματισμού.
- ❑ Η C++ είναι μια γλώσσα που προορίζεται για διαδικτυακές εφαρμογές όπως η Java και η C#.
- ☑ Ένα πρόγραμμα γραμμένο σε C μπορεί να μεταγλωττιστεί από κάποιον μεταγλωττιστή της C++.

1.2

- ☑ Όλα τα αντικείμενα μιας κλάσης έχουν τα ίδια χαρακτηριστικά και λειτουργίες.
- ❑ Η κληρονομικότητα είναι ένα χαρακτηριστικό του πολυμορφισμού.
- ☑ Από μια κλάση (βασική) μπορεί να παραχθεί μια νέα κλάση (παράγωγη). Η παράγωγη κλάση κληρονομεί όλα τα χαρακτηριστικά και τις λειτουργίες της βασικής.
- ❑ Οι αντικειμενοστρεφείς γλώσσες προγραμματισμού γνωρίζουν πώς να χρησιμοποιούν τους αριθμητικούς τελεστές (+, -, *, κλπ) για τα αντικείμενα των κλάσεων που δημιουργούμε.
- ☑ Η C++ διαθέτει βιβλιοθήκες με έτοιμες κλάσεις και αντικείμενα που μπορεί να χρησιμοποιήσει ο προγραμματιστής.

1.3

- ❑ Το ρολόι σας.
- ❑ Η καφετιέρα του σπιτιού σας.
- ☑ Αυτοκίνητο.
- ☑ Ταξίδι.
- ❑ Ένα ταξίδι που κάνατε με τους γονείς σας.

1.4

Κλάση: *Κινητό* Αντικείμενα: *Το κινητό μου*, *Το κινητό ενός φίλου μου*
Κλάση: *Άνθρωπος* Αντικείμενα: *Εγώ*, *Ο Messi*
Κλάση: *Ταξίδι* Αντικείμενα: *Ένα ταξίδι που έκανα*, *Ένα ταξίδι που θα κάνω*

1.5

Κλάση: *Κινητό*
Χαρακτηριστικά: *Χρώμα*, *Μάρκα*, *Αρ.τηλεφώνου*, *Μέγεθος μνήμης*, κ.λπ.
Λειτουργίες: *Απάντηση*, *Κλήση*, *Αποστολή SMS*, *Λήψη φωτογραφίας*, κ.λπ.

1.6

- ❑ Μπορεί να μην ανήκει σε κλάση.
- ☑ Αποτελεί στιγμιότυπο μιας κλάσης.
- ☑ Έχει όλα τα χαρακτηριστικά και τις λειτουργίες της κλάσης στην οποία ανήκει.
- ❑ Καθορίζει τις προδιαγραφές για την κατασκευή παρόμοιων αντικειμένων.
- ❑ Μοντελοποιεί πάντα ένα χειροπιαστό αντικείμενο του πραγματικού κόσμου.

1.7

- ☑ Είναι ένα σύνολο προδιαγραφών το οποίο περιγράφει τα χαρακτηριστικά και τις λειτουργίες μιας κατηγορίας αντικειμένων.
- ❑ Είναι ένα πρόγραμμα της C++.
- ❑ Είναι μια ομάδα αντικειμένων.
- ☑ Μπορεί να μην έχει κανένα αντικείμενο.
- ☑ Από μία κλάση μπορεί να παραχθεί ένα αντικείμενο αλλά και μια νέα παράγωγη κλάση.

1.8

Κλάση: *Τηλέφωνο* Παράγωγες κλάσεις: *Κινητό*, *Σταθερό*

1.9

Υπάρχει σχέση κληρονομικότητας. Από την κλάση *Όχημα* παράγονται οι κλάσεις *Μηχανάκι* και *Αυτοκίνητο*. Επίσης από την κλάση Αυτοκίνητο παράγεται η κλάση Τζιπ.

1.10

Είναι *κλάση*. **Χαρακτηριστικά** μπορεί να είναι: Η ημερομηνία που ερωτεύτηκα, το άτομο που ερωτεύτηκα, που το είδα για πρώτη φορά. **Λειτουργίες** μπορεί να είναι: Να σταματήσω να είμαι ερωτευμένος, να αυξάνεται ή να μειώνεται η ένταση του έρωτα.

1.11

Τηλεόραση. **Χαρακτηριστικά:** Μάρκα, μέγεθος, τύπος. **Λειτουργίες:** Άνοιγμα, κλείσιμο, αλλαγή καναλιού, αλλαγή έντασης.
Σκύλος. **Χαρακτηριστικά:** Ράτσα, ημερομηνία γέννησης, όνομα. **Λειτουργίες:** Γαυγίζει, κοιμάται, ξυπνάει, κάνει χαρές, δαγκώνει.
Μίσος. **Χαρακτηριστικά:** Ποιον/ποια μισώ, πόσο πολύ μισώ, πότε άρχισα να τον/την μισώ. **Λειτουργίες:** Σταματώ να μισώ, να αυξάνεται ή να μειώνεται η ένταση του μίσους.

1.12

Βασική κλάση: *Ηλεκτρονική συσκευή*

1.13

Από μια κλάση μπορεί να δημιουργηθεί ένα *αντικείμενο* αλλά και μια νέα *κλάση*. Η *ενθυλάκωση* και η *απόκρυψη* υλοποιούν την έννοια της αφαιρετικότητας. Στην *κληρονομικότητα* έχουμε βασικές και παράγωγες κλάσεις. Η υπερφόρτωση τελεστών είναι ένα από τα χαρακτηριστικά του *πολυμορφισμού*.

Ασκήσεις Κεφαλαίου 2

2.1

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c=3;
   a=b=2;
   a=c+b;
   return 0;
}
```

Μεταβλητή	Τιμή
a	5
b	2
c	3

2.2

```
#include <iostream>
#define MM 23
using namespace std;
int main()
{
   const int c=3;
   int a,b;
   a=4+b=2;
   b=c+b+MM;
   return 0;
}
```

Μεταβλητή	Τιμή
a	6
b	28
c	3

2.3

```
#include <iostream>
#define MM 23;
using namespace std;
int main()
{
   const int c=3;
   int a,b;
   a=2;
   float d;
   d=4.3
   a=4+(b=2);
   MM=10;
   3=a;
   c=c+b+MM;
   return 0;
}
```

Η οδηγίες δεν τερματίζονται με ερωτηματικό (;).

Η πρόταση θα έπρεπε να τερματίζεται με ερωτηματικό (;).

Το **MM** δεν είναι μεταβλητή. Δεν μπορεί να του ανατεθεί τιμή.

Αριστερά του τελεστή ανάθεσης επιτρέπεται μόνο μεταβλητή.

2.4

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c=3;
   a=b=2;
   a=c>b;
   b=b==1;
   return 0;
}
```

Οι **a** και **b** θα πάρουν την τιμή 2

Η **a** θα πάρει τιμή 1 διότι η παράσταση c>b (3>2) είναι αληθής.

Η **b** θα πάρει τιμή 0 διότι η παράσταση b==1 (2==1) είναι ψευδής.

Μεταβλητή	Τιμή
a	1
b	0
c	3

2.5

- ☑ Δηλωτικές προτάσεις μπορούν να μπουν σε οποιοδήποτε σημείο του προγράμματος.
- ☑ Ένα πρόγραμμα της C++ μπορεί να χωριστεί σε περισσότερα τμήματα (συναρτήσεις).
- ☑ Μια λογική παράσταση έχει τιμή 1 ή 0.
- ❑ Μια μεταβλητή στη C++, πριν να της δοθεί τιμή, έχει τιμή 0.
- ☑ Η οδηγία #define χρησιμοποιείται για να ορίσει μια σταθερά του προγράμματός μας.
- ❑ Η παράσταση a==5 καταχωρίζει στη μεταβλητή a το 5.
- ☑ Η παράσταση a=5==5 καταχωρίζει στη μεταβλητή a το 1.
- ☑ Η παράσταση a=a!=a καταχωρίζει στη μεταβλητή a το 0.

2.6

```
#include <iostream>
using namespace std;
#define ONE 12
#define TWO 78
#define ΤΕΣΣΕΡΑ 4
int main()
{
   int c=3,a,b,γ;
   float c;
   b = ONE + TWO;
   cout<<"Η γλώσσα C++ σε βάθος";
   cout<<'Τέλος';
   return 0;
}
```

Το όνομα του αρχείου κεφαλίδας είναι **iostream** και όχι **iosteam**!

Δεν επιτρέπονται ελληνικοί χαρακτήρες

Δεν επιτρέπονται ελληνικοί χαρακτήρες

Σε μονά εισαγωγικά επιτρέπεται μόνον ένας χαρακτήρας.

2.7

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c=3;
   float m,n=5.5;
   a=n;
   b=a>=5;
   m=5/2;
   c=a==b;
   return 0;
}
```

Μεταβλητή	Τιμή
a	5
b	1
c	0
m	2
n	5.5

2.8

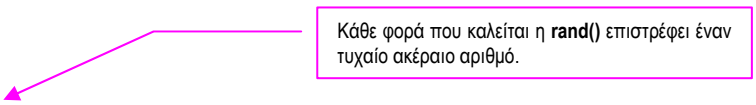
Λογική παράσταση	Τιμή	Παρατηρήσεις
a==(c-b)	1	*Η παράσταση c-b έχει αποτέλεσμα 5 το οποίο ισούται με την τιμή της a, οπότε η παράσταση είναι αληθής (1).*
a>b || b>c	0	*Η παράσταση a>b είναι ψευδής όπως και η b>c*
a==5 && c==15	1	*Η παράσταση a==5 είναι αληθής όπως και η c==15*
a==5 && c>20	0	*Η παράσταση a==5 είναι αληθής αλλά η c>20 ψευδής*

2.9

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c;
   float mo;
   a=3;
   b=7;
   c=21;
   mo=(a+b+c)/3.0;
   return 0;
}
```

2.10

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int a,b,c;
   float mo;
   a=rand();
   b=rand();
   c=rand();
   mo=(a+b+c)/3.0;
   return 0;
}
```



2.11

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   const int a=333,b=777,c=999;
   float sum;
   sum=sqrt(a)+sqrt(b)+sqrt(c);
   // cout << sum;  // αφαιρέστε το σύμβολο του σχολίου αν θέλετε να δείτε το αποτέλεσμα
   return 0;
}
```

2.12

```
#include <iostream>
#include <cmath>
#include <cstdlib>
using namespace std;
int main()
{
   float sum;
   sum=sqrt(rand())+sqrt(rand())+sqrt(rand());
   // cout << sum;  // Αφαιρέστε το σύμβολο του σχολίου αν θέλετε να δείτε το αποτέλεσμα
   return 0;
}
```

Ασκήσεις Κεφαλαίου 3

3.1

```
#include <iostream>
using namespace std;
int main()
{
   int a=4,b=5;
   char ch;
   ch='A';
   cout  << a <<"---"
         << b << endl << "Χαρ=" << ch;
   return 0;
}
```

```
4---5
Χαρ=A
```

3.2

☑ Κατά τη χρήση του αντικειμένου cin μετά από τον τελεστή εξαγωγής >> επιτρέπεται μόνο ένα όνομα μεταβλητής.

❑ Κατά τη χρήση του αντικειμένου cout μετά από τον τελεστή εισαγωγής << επιτρέπεται μόνο ένα όνομα μεταβλητής.

❑ Το μέγεθος ενός τύπου δεδομένων στη C++ είναι πάντα το ίδιο και ανεξάρτητο από το σύστημα στο οποίο δουλεύουμε.

☑ Ο έλεγχος if(a=5) είναι πάντα αληθής.

❑ Αν μέσα σε ένα πρόγραμμα δεν υπάρχει κλήση της exit(), το πρόγραμμα δε θα τερματιστεί ποτέ.

☑ Ο τελεστής sizeof μπορεί να εφαρμοστεί και σε κάποιο τύπο δεδομένων, όπως sizeof(char).

3.3

☑ `cin >> a,b;`

*Μετά τον τελεστή εξαγωγής **>>** επιτρέπεται το όνομα μίας μόνο μεταβλητής. Αν θέλαμε να διαβάσουμε δύο τιμές, η πρόταση θα έπρεπε να ήταν: **cin>>a>>b;***

☑ `cin >> a >> b >> endl;`

*Ο χειριστής **endl** χρησιμοποιείται στο αντικείμενο **cout**. Μετά τον τελεστή εξαγωγής **>>** επιτρέπεται μόνο όνομα μεταβλητής.*

☑ `cin >> "Δώσε δυο τιμές" >> a >> b;`

*Μετά τον τελεστή εξαγωγής **>>** επιτρέπεται μόνο όνομα μεταβλητής και όχι συμβολοσειρά!*

☑ `cin << a << b;`

*Χρησιμοποιήθηκε ο τελεστής εισαγωγής **<<** αντί του τελεστή εξαγωγής **>>**!*

❑ `cin >> a >> b;`

☑ `cin >> "a";`

*Μετά τον τελεστή εξαγωγής **>>** επιτρέπεται μόνο όνομα μεταβλητής και όχι συμβολοσειρά ("a")!*

☑ `cin >> 5 >> a;`

*Μετά τον τελεστή εξαγωγής **>>** επιτρέπεται μόνο όνομα μεταβλητής. Το 5 είναι αριθμητική σταθερά.*

3.4

☑ `cout >> a >> b;`

*Χρησιμοποιήθηκε ο τελεστής εξαγωγής **>>** αντί του τελεστή εισαγωγής **<<**!*

☑ `cout << "NAI" 10;`

*Μετά τον τελεστή εισαγωγής επιτρέπεται κάποια συντακτικά σωστή παράσταση ή κάποιος χειριστής. Το **"NAI" 10** δεν είναι τίποτα από τα δύο. Αν θέλαμε να εμφανιστούν και τα δύο θα έπρεπε η πρόταση να ήταν: **cout<<"NAI"<<10;***

- ❑ `cout << a << endl << b;`
- ☑ `cout << "OXI" << end;`

  *Ο χειριστής αλλαγής γραμμής είναι **endl** και όχι **end**. Μπορεί να θεωρηθεί και συντακτικά σωστή στην περίπτωση που υπάρχει μεταβλητή με όνομα **end**.*
- ❑ `cout << a>b;`
- ☑ `cout << "Δώσε μια τιμή" >> a;`

  *Χρησιμοποιήθηκε ο τελεστής εξαγωγής **>>** αντί του τελεστή εισαγωγής **<<**!*

3.5

Η C++ χρησιμοποιεί το αντικείμενο *cout* για έξοδο πληροφοριών στην οθόνη και το αντικείμενο *cin* για είσοδο δεδομένων από το πληκτρολόγιο. Η αλλαγή γραμμής επιτυγχάνεται με χρήση του χειριστή *endl* ο οποίος χρησιμοποιείται μόνο σε συνδυασμό με το αντικείμενο *cout*. Ο χώρος ονομάτων στον οποίο ορίζονται οι οντότητες της καθιερωμένης βιβλιοθήκης της C++ είναι ο *std*. Η πρόταση με την οποία ορίζουμε αυτόν τον χώρο ονομάτων ως τον προεπιλεγμένο είναι η *using namespace std;*

3.6

```
#include <iostream>
using namespace std;
int main()
{
   int a,b;
   float f;
   char ch;
   cout  << sizeof a << endl << sizeof f << endl
         << sizeof ch << endl;
   cin >> a >> f >> ch;
   cout << a << " " << f << " " << ch << endl;
   return 0;
}
```

☞ Η πρώτη πρόταση cout εμφανίζει τα μεγέθη (σε byte) των μεταβλητών a, f και ch, δηλαδή το 4, το 4 και το 1 αντίστοιχα.

☞ Η πρόταση cin ζητάει να πληκτρολογηθούν τρεις τιμές (δύο αριθμοί και ένας χαρακτήρας) από το πληκτρολόγιο και τις καταχωρίζει στις μεταβλητές a, f και ch αντίστοιχα.

☞ Η τελευταία πρόταση cout εμφανίζει τα περιεχόμενα των μεταβλητών a, f και ch.

3.7

```
#include <iostream>
using namespace std;
int main()
{
   int a,b;
   cin >> a >> b;
   if (a>b) cout << a;
   else  cout << b;
   return 0;
}
```

☞ Η πρόταση cin ζητάει να πληκτρολογηθούν δύο αριθμοί από το πληκτρολόγιο και τους καταχωρίζει στις μεταβλητές a και b αντίστοιχα.

☞ Η εντολή if ελέγχει αν η τιμή της μεταβλητής a είναι μεγαλύτερη από την τιμή της μεταβλητής b. Αν είναι, εμφανίζει την τιμή της a διαφορετικά εμφανίζει την τιμή της b. Δηλαδή, σε κάθε περίπτωση εμφανίζει τον μεγαλύτερο από τους δύο αριθμούς που δώσαμε.

3.8

```
#include <iostream>
int main()
{
   float a,b,sum;
   std::cin >> a >> b;
   sum=a+b;
   std::cout << sum << std::endl << "ΝΙΚΟΣ" << std::endl << "-----";
   return 0;
}
```

3.9

*Επειδή έχει παραληφθεί η πρόταση **using namespace std**, δεν έχει ορισθεί ως χώρος ονομάτων ο **std** στον οποίο ορίζονται οι οντότητες της καθιερωμένης βιβλιοθήκης της C++. Έτσι, ο μεταγλωττιστής δεν αναγνωρίζει τα αντικείμενα **cout**, **cin** και τον χειριστή **endl**. Η λύση, αν δεν θέλουμε να προσθέσουμε την πρόταση **using namespace std,** είναι να προσθέσουμε το πρόθεμα του χώρου ονομάτων **std::** πριν από κάθε οντότητα .*

```
#include <iostream>
int main()
{
   float a,b;
   std::cin >> a >> b;
   std::cout << a << b << std::endl << "ΤΕΛΟΣ" << std::endl;
   return 0;
}
```

3.10

```
#include <iostream>
using namespace std;
int main()
{
   float a,b,c,mo;
   cout << "Δώσε τρεις αριθμούς:";
   cin >> a >> b >> c;
   mo=(a+b+c)/3;
   cout << "Ο μέσος όρος είναι " << mo << endl;
   return 0;
}
```

3.11

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c;
   cin >> a >> b >> c;
   if (a>=b && a>=c)
      cout << a << endl;
   else if (b>=a && b>=c)
      cout << b << endl;
   else
      cout << c;
   return 0;
}
```

3.12

```
#include <iostream>
using namespace std;
int main()
{
   float a,b;
   cin >> a >> b;
   if (a==0 || b==0)
      cout << "Λάθος δεδομένα";
   else
   {
      cout << "Άθροισμα = " << a+b << endl;
      cout << "Γινόμενο = " << a*b << endl;
      cout << "Πηλίκο = " << a/b << endl;
   }
   return 0;
}
```

Ελέγχουμε αν κάποιος από τους δύο αριθμούς είναι 0.

3.13

```
#include <iostream>
using namespace std;
int main()
{
   float a,b,c,mo,gin,sum;
   cout << "Δώσε τον πρώτο αριθμό:";
   cin >> a;
   cout << "Δώσε τον δεύτερο αριθμό:";
   cin >> b;
   cout << "Δώσε τον τρίτο αριθμό:";
   cin >> c;
   mo=(a+b+c)/3;
   gin=a*b*c;
   sum=a+b+c;
   cout << "Το άθροισμα των " << a << "," << b << "," << c
        << " είναι " << sum << endl;
   cout << "Το γινόμενο των " << a << "," << b << "," << c
        << " είναι " << gin << endl;
   cout << "Ο μέσος όρος των " << a << "," << b << "," << c
        << " είναι " << mo << endl;
   return 0;
}
```

3.14

```
#include <iostream>
using namespace std;
int main()
{
   float b1,b2,b3,mo;
   cout << "Δώσε τρεις βαθμούς: ";
   cin >> b1 >> b2 >> b3;
   if (b1>=b3 && b2>=b3)
      mo=(b1+b2)/2;
   else if (b2>=b1 && b3>=b1)
      mo=(b2+b3)/2;
   else
      mo=(b1+b3)/2;
   cout << "Ο μέσος όρος είναι " << mo << endl;
   return 0;
}
```

Ασκήσεις Κεφαλαίου 4

4.1

Πρόταση	Τιμή του x	Τιμή της παράστασης
x++;	101	100
++x;	101	101
x--;	99	100
--x;	99	99

4.2

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,aa,bb,x,y;
   x = y = 100;
   a = ++x;
   b = y++;
   aa = ++x;
   bb = y++;
   cout << "Η τιμή του a είναι " << a << endl;
   cout << "Η τιμή του b είναι " << b << endl;
   cout << "Η τιμή του aa είναι " << aa << endl;
   cout << "Η τιμή του bb είναι " << bb << endl;
   return 0;
}
```

```
Η τιμή του a είναι 101
Η τιμή του b είναι 100
Η τιμή του aa είναι 102
Η τιμή του bb είναι 101
```

4.3

Παράσταση	Τιμή του x	Τιμή του y
x=y;	100	100
x = --y * 4;	396 (99*4)	99
x = y == 100;	1	100
x = y == y++;	0	101
x = y == ++y;	1	101

4.4

```
if (x==z)
   cout << "x==z";
if (x<z)
   cout << "x<z";
if (x>z)
   cout << "x>z";
```

☞ Η μεταβλητή z θα πάρει την τιμή 9 διότι η παράσταση x/y έχει αποτέλεσμα 3 (το ακέραιο πηλίκο των αριθμών). Οπότε στην οθόνη θα εμφανιστεί το "x>z".

4.5

```
z = ++y + x++;
```

4.6

- ❑ Το i++ αυξάνει την τιμή του i κατά 1 ενώ το ++i όχι.
- ☑ Οι τελεστές ++ και -- εφαρμόζονται μόνο σε μεταβλητές.
- ❑ Ο τελεστής ανάθεσης = έχει την υψηλότερη προτεραιότητα.

- ☑ Η παράσταση 5/2 έχει αποτέλεσμα τύπου int (το 2).
- ☑ Η παράσταση 5/2.0 έχει αποτέλεσμα τύπου double (το 2.5).

4.7

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   float a,b,mo;
   cin >> a >> b;
   mo=(a+b)/2;
   cout  << "MO=" << setw(7) << setprecision(2) << fixed       << mo << endl;
   return 0;
}
```

4.8

```
#include <iostream>
#include <cmath>
#define pi 3.141592
using namespace std;
int main()
{
   double r,e;
   cout << "Δώσε ακτίνα :";
   cin >> r;
   e=pow(r,2)*pi;
   cout  << "Το εμβαδόν κύκλου ακτίνας " << r
         << " είναι " << e << endl;
   return 0;
}
```

4.9

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   double a,b;
   float c=5;
   a=10.123;
   b=167.15;
   cout << setw(9) << setprecision(4) << fixed << a << endl;
   cout << setw(8) << setprecision(3) << fixed << b << endl;
   cout << setw(7) << setprecision(2) << fixed << c << endl;
   return 0;
}
```

```
 10.1230
167.150
   5.00
```

4.10

```
#include <iostream>
using namespace std;
int main()
{
   double d;
   int a,b;
   a=5;
   b=6;
```

```
5
```

```
   d=(a+b)/2;
   cout << d;
   return 0;
}
```

☞ Η παράσταση (a+b)/2 θα έχει αποτέλεσμα τύπου int διότι όλα τα μέλη της είναι τύπου int. Οπότε, το αποτέλεσμα θα είναι 5, και όχι 5.5 που θα ήταν το αναμενόμενο.

☞ Αν θέλαμε να υπολογιζόταν σωστά τότε θα έπρεπε να γραφεί ως (a+b)/2.0. Το 2.0 που είναι τύπου double «εξαναγκάζει» την όλη παράσταση να έχει αποτέλεσμα double, οπότε διατηρεί τα δεκαδικά της ψηφία.

4.11

- ❑ Οι μεταβλητές τύπου long double αποθηκεύουν απεριόριστο αριθμό δεκαδικών ψηφίων.
- ❑ Με το αντικείμενο cout δεν μπορούμε να καθορίσουμε τον ακριβή αριθμό των δεκαδικών ψηφίων που θα εμφανίζονται στην οθόνη.
- ❑ Η παράσταση 1+1.0 έχει αποτέλεσμα τύπου float.
- ☑ Η C++ δεν έχει τελεστή για ύψωση σε δύναμη.
- ☑ Οι διαδοχικές πράξεις μεταξύ αριθμών μπορεί να έχουν αποτέλεσμα την απώλεια ακρίβειας σε δεκαδικά ψηφία.

4.12

```
#include <iostream>
using namespace std;
int main()
{
   float d,yp;
   int a,b;
   cin >> d >> a;
   b=d/a;
   yp=d-b*a;
   cout << yp << endl;
   return 0;
}
```

☞ Στη πρόταση b=d/a, η b είναι τύπου int οπότε θα αποθηκευτεί μόνο το ακέραιο τμήμα του αποτελέσματος της παράστασης d/a.

4.13

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   double a,b,gin;
   cout << "Δώσε δύο αριθμούς:";
   cin >> a >> b;
   gin = a*b;
   cout << " " << setw(9) << setprecision(2) << fixed << a << endl;
   cout << "x" << setw(9) << setprecision(2) << fixed << b << endl;
   cout << "==========" << endl;
   cout << " " << setw(9) << setprecision(2) << fixed << gin << endl;
   return 0;
}
```

4.14

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   double a,b,x;
   cout << "Δώσε τιμή για το Α :";
   cin >> a;
   cout << "Δώσε τιμή για το Β :";
   cin >> b;
   x=(a/(a+b))*(b/(a-b))+pow(a,a+b)/pow(b,a-b);
   cout << "Το αποτέλεσμα της παράστασης είναι: " << x;
   return 0;
}
```

4.15

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   cin >> ch;
   if ((ch>='a' && ch<='z'))
      cout << ch;;
   if (ch>='0' && ch<='9')
      cout << "Πατήθηκε ένα ψηφίο" << endl;
   return 0;
}
```

4.16

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   cin >> ch;
   if ((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
   {
      ch++;
      cout << ch << endl;
   }
   if (ch>='0' && ch<='9')
      cout << ch << endl;
   return 0;
}
```

4.17

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   cout << "1-Εκτύπωσε την λέξη \"Hello\"" << endl;
   cout << "2-Εκτύπωσε τον αριθμό 2" << endl;
   cout << "3-Εκτύπωσε \"bye bye\"" << endl;
```

Για να εμφανιστούν τα διπλά εισαγωγικά (") πρέπει να χρησιμοποιηθεί ο χαρακτήρας διαφυγής \". Δεν μπορούμε απλά να τα γράψουμε μέσα σε ένα αλφαριθμητικό διότι έχουν ιδιαίτερο νόημα για τη C++.

```
   cout << "4-Μην κάνεις τίποτα" << endl;
   cout << "Δώσε επιλογή:";
   cin >> ch;
   if (ch=='1') cout << "Hello"<< endl;
   if (ch=='2') cout << "2" << endl;
   if (ch=='3') cout << "bye bye" << endl;
   if (ch!='1' && ch!='2' && ch!='3' && ch!='4')
      cout << "Λάθος επιλογή" << endl;
   return 0;
}
```

4.18

```
#include <iostream>
using namespace std;
int main()
{
   char ch,b='A';
   ch='B';
   if(ch==b)
      cout << "NAI-1\n";
   else
      cout << "OXI-1\n";
   if(ch=b)
      cout << "NAI-2\n";
   else
      cout << "OXI-2\n";
   return 0;
}
```

```
OXI-1
NAI-2
```

☞ Η έκφραση ch==b είναι ψευδής διότι οι δύο μεταβλητές περιέχουν διαφορετικούς χαρακτήρες.

☞ Η έκφραση ch=b είναι αληθής διότι δεν πρόκειται για σύγκριση (==) αλλά για ανάθεση τιμής (=). Το περιεχόμενο της μεταβλητής b καταχωρίζεται στη ch και το αποτέλεσμα της όλης παράστασης είναι το περιεχόμενο της ch. Από τη στιγμή που η τιμή της ch δεν είναι 0, η παράσταση ch=b θεωρείται αληθής.

4.19

☑ Μπορούμε να χειριζόμαστε τους χαρακτήρες ως αριθμούς.

☑ Μια μεταβλητή τύπου char έχει το μέγεθος ενός byte.

☑ Σε μια μεταβλητή χαρακτήρα μπορούμε να καταχωρίσουμε έναν ακέραιο αριθμό.

☑ Οι συμβολοσειρές προσδιορίζονται από τη διεύθυνση όπου είναι αποθηκευμένος ο πρώτος τους χαρακτήρας.

❑ Μια συμβολοσειρά καταλαμβάνει τόσα byte όσοι ακριβώς είναι και οι χαρακτήρες που περιέχει.

4.20

❑ Το string είναι ένας βασικός τύπος δεδομένων της C++.

☑ Το string είναι μια κλάση η οποία διατίθεται από την καθιερωμένη βιβλιοθήκη της C++.

☑ Η σύγκριση αντικειμένων κλάσης string μεταξύ τους ή με συμβολοσιερές μπορεί να γίνει με χρήση των συγκριτικών τελεστών.

❑ Σύγκριση μεταξύ συμβολοσειρών μπορεί να γίνει με χρήση των συγκριτικών τελεστών.

☑ Σε ένα αντικείμενο κλάσης string μπορεί να καταχωριστεί είτε μια συμβολοσειρά είτε ένας χαρακτήρας.

4.21

```
#include <iostream>
using namespace std;
int main()
{
   int code;
   char ch;
   ch='a';
   code=ch;
   cout << "Ο κωδικός ASCII του " << ch << " είναι "<< code << endl;
   ch='*';
   code=ch;
   cout << "Ο κωδικός ASCII του " << ch << " είναι "<< code << endl;
   ch=' ';
   code=ch;
   cout << "Ο κωδικός ASCII του " << ch << " είναι "<< code << endl;
   return 0;
}
```

Στη μεταβλητή **code** καταχωρίζεται ο κωδικός ASCII του χαρακτήρα που περιέχει η μεταβλητή **ch** (αυτόματη μετατροπή τύπου).

4.22

```
#include <iostream>
using namespace std;
int main()
{
   int code;
   char ch;
   code=80;
   ch=code;
   cout << "Ο χαρακτήρας με κωδικό " << code << " είναι ο "<< ch << endl;
   code=125;
   ch=code;
   cout << "Ο χαρακτήρας με κωδικό " << code << " είναι ο "<< ch << endl;
   code=192;
   ch=code;
   cout << "Ο χαρακτήρας με κωδικό " << code << " είναι ο "<< ch << endl;
   return 0;
}
```

Στη μεταβλητή **ch** καταχωρίζεται ο χαρακτήρας με κωδικό ASCII ίσο με τον αριθμό που περιέχει η μεταβλητή **code** (αυτόματη μετατροπή τύπου).

4.23

```
#include <iostream>
using namespace std;
int main()
{
   char ch=68,let='L';
   int a=2,b=4;
   a=ch+let;
   ch=++let;
   cout << ++a << ch << let;
   return 0;
}
```

```
145M
```

☞ Η παράσταση ch+let έχει αποτέλεσμα 144 (68 + 76) αφού ο κωδικός ASCII του 'L' είναι 76. Ακολούθως η τιμή καταχωρίζεται στη μεταβλητή a η οποία με τη πρόταση ++a αυξάνεται κατά 1 και γίνεται 145.

☞ Η πρόταση ++let αυξάνει τη let κατά 1 και την κάνει 77 που είναι ο κωδικός ASCII του 'M'.

4.24

☞ Το 'A' αναφέρεται στον χαρακτήρα 'A' και ισοδυναμεί με τον αριθμό 65 που είναι ο ASCII κωδικός του 'A'.

☞ Το "A" αναφέρεται σε μία συμβολοσειρά και ισοδυναμεί με τη διεύθυνση της πρώτης θέσης μνήμης στην οποία έχει καταχωριστεί η συγκεκριμένη συμβολοσειρά.

4.25

```
#include <iostream>
using namespace std;
int main()
{
   string lex1="AEGEAN",lex2="UNIVERSITY",final;
   char ch1=' ';
   final=lex2+ch1+"OF THE"+ch1+lex1;
   cout << final << endl << "=======================" << endl;
   return 0;
}
```

4.26

☑ Ο τύπος bool είναι από τους βασικούς τύπους της γλώσσας από την απαρχή της γλώσσας.

☑ Οι μεταβλητές τύπου bool μπορούν να λάβουν μόνο δύο τιμές.

❑ Η σταθερά 1.0 είναι τύπου float, ενώ η 1.123456789 είναι τύπου double.

❑ Ο τύπος float έχει πάντα μέγεθος 4 byte ανεξάρτητα από την αρχιτεκτονική του συστήματος και τον μεταγλωττιστή της γλώσσας.

❑ Η παράσταση 5/2 έχει ως αποτέλεσμα 2.5.

4.27

```
#include <iostream>
using namespace std;
int main()
{
   string onom;
   cout << "Δώσε ένα όνομα:";
   cin >> onom;
   if (onom>="C" && onom<"D")
      cout << "Σωστά" << endl;
   else
      cout << "Λάθος" << endl;
   return 0;
}
```

Στην περίπτωση που το όνομα είναι αλφαβητικά μεγαλύτερο ή ίσο από το "C" τότε σίγουρα αρχίζει από C ή από επόμενο γράμμα. Αν τώρα ταυτόχρονα είναι αλφαβητικά μικρότερο από "D" τότε η μόνη περίπτωση για να είναι αληθής η λογική παράσταση είναι το όνομα να αρχίζει από τον χαρακτήρα 'C'!
Σκεφτείτε τον τηλεφωνικό κατάλογο. Ποια ονόματα είναι μετά τον κύριο "C" και πριν από τον κύριο "D";

4.28

1. "++"
2. "100"
3. "35"
4. "C++"
5. "ΦΑΝΗΣ"

Ασκήσεις Κεφαλαίου 5

5.1

```
#include <iostream>
using namespace std;
int main()
{
   float timi,kostos,e;
   int pos;
   cout << "Δώσε ποσότητα και τιμή:";
   cin >> pos >> timi;
   kostos=timi*pos;
   if (pos>100)
   {
      e=kostos*25/100;
      kostos=kostos-e;
   }
   else if (pos>=80 && pos<=100)
   {
      e=kostos*15/100;
      kostos=kostos-e;
   }
   else if (pos<20)
   {
      e=kostos*10/100;
      kostos=kostos+e;
   }
   cout << "Το τελικό κόστος είναι " << kostos << endl;
   return 0;
}
```

5.2

```
#include <iostream>
using namespace std;
int main()
{
   int a,b;
   char ch;
   cin >> ch;
   if ((ch>='A') && (ch<='Z'))
      ++ch;
   else
      --ch;
   cout << ch << endl;
   return 0;
}
```

Περιμένει να πληκτρολογηθεί ένας χαρακτήρας τον οποίο καταχωρίζει στη μεταβλητή **ch**.

Αν ο χαρακτήρας είναι κεφαλαίος λατινικός τότε αυξάνει το περιεχόμενο της **ch** κατά 1, διαφορετικά το μειώνει κατά 1.

Εμφανίζει τον χαρακτήρα με κωδικό **ch**, ο οποίος θα είναι ή ο επόμενος ή ο προηγούμενος από τον χαρακτήρα που δόθηκε αρχικά.

5.3

```
#include <iostream>
using namespace std;
int main()
{
   int b;
   char a;
   cin >> a >> endl;
   if a<0 exit(0);
   b='*';
```

Το αντικείμενο **cin** δεν έχει χειριστή **endl**!

Η συνθήκη της **if** πρέπει να είναι μέσα σε παρενθέσεις.

```
   switch(a)
   {
      case 1:
         cout << a << endl;
         cout << "-------\n";
      case b:
         cout << b << endl;
         break;
      case 'A':
         cout << "aaaaaaaaaaa";
         break;
      case 'A'+1:
         cout << "telos";
         break;
      case 4
         cout << "444444444";
   }
}
```

Η **case** ακολουθείται μόνο από ακέραιες σταθερές. ή σταθερές χαρακτήρων και όχι από μεταβλητές (b).

Η **case** ακολουθείται μόνο από σταθερές. και όχι από παραστάσεις ('A'+1).

Η **case** πρέπει να τερματίζεται με :

Λείπει το **return 0**;

5.4

☞ Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'B' (έχει κωδικό ASCII 66) θα ισχύει η πρώτη περίπτωση, case 66:, και θα εκτελεστούν οι προτάσεις της πρώτης case. Επειδή όμως οι προτάσεις της πρώτης case δεν τερματίζονται με εντολή break, θα εκτελεστούν και οι προτάσεις της case 7:.

☞ Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας 'A' θα εκτελεστούν οι προτάσεις της case 'A':.

☞ Στην περίπτωση που πληκτρολογηθεί ο χαρακτήρας '*', δεν υπάγεται σε καμία από τις περιπτώσεις case και θα εκτελεστούν οι προτάσεις της default:.

```
100
-------
44
```

```
aaaaaaaaaaa
```

```
1234567890
```

5.5

```
#include <iostream>
using namespace std;
int main()
{
   int etos;
   cout << "Δώσε έτος:";
   cin >> etos;
   if (etos%4 == 0)
   {
      if (etos%100 == 0)
      {
         if(etos%400 == 0)
            cout << "Δίσεκτο" << endl;
         else
            cout << "Κανονικό" << endl;
      }
      else
         cout << "Δίσεκτο" << endl;
   }
   else
      cout << "Κανονικό" << endl;
   return 0;
}
```

Για να είναι ένα έτος δίσεκτο πρέπει να διαιρείται ακριβώς με το 4. Η παράσταση **etos%4** δίνει το υπόλοιπο της ακέραιας διαίρεσης με το 4.

Αν διαιρείται ακριβώς με το 100 δεν είναι δίσεκτο.

Εκτός αν διαιρείται ακριβώς με το 400. .

Θα μπορούσαμε επίσης να έχουμε το ίδιο αποτέλεσμα με μία μόνο **if**:

```
if (((etos%4 == 0)&& (etos%100 != 0)) || (etos%400 == 0))
   cout << "Δίσεκτο" << endl;
```

Η λογική παράσταση ελέγχει ταυτόχρονα όλα όσα ελέγχαμε πριν με τις τρεις ένθετες **if**.

```
else
   cout << "Κανονικό" << endl;
```

5.6

```
#include <iostream>
using namespace std;
int main()
{
   int a,b;
   char ch;
   cin >> ch;
   switch(ch)
   {
      case 'h':
         cout << "hello" << endl;
         break;
      case '2':
         cout << "2" << endl;
         break;
      case 'b':
         cout << "bye bye" << endl;
         break;
      default:
         cout << "Λάθος επιλογή" << endl;
   }
   return 0;
}
```

5.7

- ☑ Η πρόταση case πρέπει να ακολουθείται απαραίτητα από ακέραια σταθερά ή σταθερά χαρακτήρα.
- ☑ Η εντολή switch-case μπορεί να αντικατασταθεί από ισοδύναμες εντολές if-else if.
- ❑ Οι εντολές if-else if μπορούν να αντικατασταθούν από ισοδύναμες εντολές switch-case.
- ❑ Η πρόταση a==5 είναι ισοδύναμη με την a=5.
- ❑ Η εντολή if είναι η μοναδική εντολή που διαθέτει η C++ για τον έλεγχο λογικών παραστάσεων.

5.8

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   float ypsos,baros,dms;
   cout << "Δώσε ύψος και βάρος :";
   cin >> ypsos >> baros;
   dms = baros/pow(ypsos,2);
   if (dms<18.5)
      cout << "Λιποβαρής" << endl;
   else if (dms>=18.5 && dms<25)
      cout << "Κανονικός " << endl;
   else if (dms>=25 && dms<30)
      cout << "Υπέρβαρος" << endl;
   else
      cout << "Παχύσαρκος" << endl;
   return 0;
}
```

Το αρχείο κεφαλίδας **cmath** συμπεριλαμβάνεται για να μπορεί να χρησιμοποιηθεί στο πρόγραμμα η συνάρτηση **pow()**.

5.9

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int plithos;
   char eidos;
   float poso;
   cout << "Δώσε πλήθος και είδος:";
   cin >> plithos >> eidos;
   switch(eidos)
   {
      case 'E':
         poso=plithos*0.23;
         break;
      case 'A':
         poso=plithos*0.70;
         break;
      case 'T':
         poso=plithos*0.15;
         break;
      default:
         cout << "Λάθος είδος" << endl;
         exit(1);
   }
   cout << "Ποσό:" << poso << endl;
   return 0;
}
```

Το αρχείο κεφαλίδας **cstdlib** συμπεριλαμβάνεται για να μπορεί να χρησιμοποιηθεί στο πρόγραμμα η συνάρτηση **exit()**.

5.10

```
#include <iostream>
using namespace std;
int main()
{
   int sms;
   float poso;
   cout << "Δώσε πλήθος sms :";
   cin >> sms;
   if (sms<=10)
      poso=sms*2;
   else if (sms<=60)
      poso=10*2 + (sms-10)*1.5;
   else if (sms<=160)
      poso=10*2 + 50*1.5 + (sms-60)*1.2;
   else
      poso=10*2 + 50*1.5 + 100*1.2 + (sms-160)*1;
   cout << "Συνολικό ποσό σε euro: " << poso/100;
   return 0;
}
```

5.11

```
#include <iostream>
using namespace std;
int main()
{
   int epivates,theseis;
   float pos;
```

```
   cout << "Δώσε θέσεις και επιβάτες :";
   cin >> theseis >> epivates;
   pos=epivates*100.0/theseis;      // υπολογισμός ποσοστού πληρότητας
   if (pos>=50)
      cout << "Κέρδος" << endl;
   else if (pos<30)
      cout << "Ζημία" << endl;
   return 0;
}
```

5.12

```
#include <iostream>
using namespace std;
int main()
{
   float milia,litra,kat;
   cout << "Δώσε μίλια και λίτρα :";
   cin >> milia >> litra;
   kat=litra/milia;                 // υπολογισμός κατανάλωσης λίτρων ανά μίλι
   if (kat<=0.9)
      cout << "Πολύ χαμηλή" << endl;
   else if (kat<=1.2)
      cout << "Χαμηλή" << endl;
   else if (kat<=1.8)
      cout << "Κανονική" << endl;
   else
      cout << "Υψηλή" << endl;
   return 0;
}
```

5.13

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   int a=100,b;
   b=44;
   cin >> ch;
   if (ch==66)
   {
      cout << a << endl;
      cout << "-------" << endl;
   }
   else if (ch==7)
   {
      --b;
      cout << b;
   }
   else if (ch=='A')
   {
      cout << "aaaaaaaaaaa";
   }
   else
   {
      b++;
      cout << "1234567890";
   }
   return 0;
}
```

5.14

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int litra,xar_20,xar_10,xar_5,ker_1,ker_2;
   int resta;
   cout << "Δωσε αριθμό λίτρων:";
   cin >> litra;
   resta=50-litra;
   if (resta<0)
   {
      cout << "Δεν φτάνουν τα χρήματα για τόσα λίτρα" << endl;
      exit(1);
   }
   xar_20=resta/20;
   resta=resta % 20;
   xar_10=resta/10;
   resta=resta % 10;
   xar_5=resta/5;
   resta=resta % 5;
   ker_2=resta/2;
   resta=resta % 2;
   ker_1=resta;
   cout << "Ρέστα" << endl;
   if (xar_20>0) cout << "Χατρονομίσματα των 20 euro:" << xar_20 << endl;
   if (xar_10>0) cout << "Χατρονομίσματα των 10 euro:" << xar_10 << endl;
   if (xar_5>0) cout << "Χατρονομίσματα των 5 euro:" << xar_5 << endl;
   if (ker_2>0) cout << "Κέρματα των 2 euro:" << ker_2 << endl;
   if (ker_1>0) cout << "Κέρματα του 1 euro:" << ker_1 << endl;
   return 0;
}
```

Το αρχείο κεφαλίδας **cstdlib** συμπεριλαμβάνεται για να μπορεί να χρησιμοποιηθεί στο πρόγραμμα η συνάρτηση **exit()**.

Τα χαρτονομίσματα των 20€ προκύπτουν από την ακέραια διαίρεση του ποσού των ρέστων με το 20. Το υπόλοιπο ποσό που περισσεύει προκύπτει από την παράσταση **resta % 20**.

Η ίδια διαδικασία ακολουθείται και για τον υπολογισμό των υπόλοιπων χαρτονομισμάτων και κερμάτων.

5.15

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int ar1,ar2,ps1,ps2,ps3,ps4;
   cout << "Δωσε αριθμό :";
   cin >> ar1;
   if (ar1>9999 || ar1<1000)
   {
      cout << "Λάθος αριθμός" << endl;
      exit(1);
   }
   ps1=ar1%10;
   ps2=(ar1/10)%10;
   ps3=(ar1/100)%10;
   ps4=(ar1/1000)%10;
   ar2=ps1*1000+ps2*100+ps3*10+ps4;
   cout << "Κατοπτρικός :" << endl;
   return 0;
}
```

Ελέγχεται αν ο αριθμός είναι θετικός τετραψήφιος.

Στις μεταβλητές **ps1**, **ps2**, **ps3** και **ps4** καταχωρίζονται τα τέσσερα ψηφία του αριθμού.

Δημιουργείται ο κατοπτρικός αριθμός.

Ασκήσεις Κεφαλαίου 6

6.1

Με χρήση της εντολής **while** ...

```
#include <iostream>
using namespace std;
int main()
{
   int a,sum;
   a=sum=0;
   while(a<=1000)
   {
      sum=sum+a;
      a++;
   }
   cout << "Το άθροισμα είναι " << sum << endl;
   return 0;
}
```

Κάθε φορά που εκτελείται η πρόταση **sum=sum+a**, στη **sum** προστίθεται η νέα τιμή της **a**.

Με χρήση της εντολής **for** ...

```
#include <iostream>
using namespace std;
int main()
{
   int a,sum;
   sum=0;
   for(a=0;a<=1000;a++)
   {
      sum=sum+a;
   }
   cout << "Το άθροισμα είναι " << sum << endl;
   return 0;
}
```

6.2

```
#include <iostream>
using namespace std;
int main()
{
   int a,num,num1,num2;
   num1=num2=0;
   for (a=1;a<=100;++a)
   {
      cin >> num;
      switch(num % 2)
      {
         case 0:
            ++num2;
            break;
         case 1:
            ++num1;
            break;
      }
   }
   cout << "num1=" << num1 << endl << "num2=" << num2 << endl;
   return 0;
}
```

Οι προτάσεις της **for** θα εκτελεστούν 100 φορές.

Ζητάει έναν αριθμό και τον καταχωρίζει στη μεταβλητή **num**.

Η παράσταση **num % 2** υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του αριθμού με το 2.

Αν είναι 0, αυτό σημαίνει ότι ο αριθμός είναι ζυγός. Τότε αυξάνει τη μεταβλητή **num2** κατά 1.

Αν είναι 1, αυτό σημαίνει ότι ο αριθμός είναι μονός. Τότε αυξάνει τη μεταβλητή **num1** κατά 1.

☞ Με απλά λόγια, το πρόγραμμα ζητάει να δώσουμε 100 αριθμούς και στο τέλος εμφανίζει πόσοι από αυτούς ήταν μονοί και πόσοι ζυγοί.

☞ Η μεταβλητή num1 χρησιμοποιείται για να «μετράει» τους μονούς αριθμούς ενώ η μεταβλητή num2 χρησιμοποιείται για να «μετράει» τους ζυγούς αριθμούς.

6.3

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   int a,fl=0;
   a=0;
   while (fl<2)
   {
      cin >> ch;
      if (ch=='*') fl++;
      if (fl==1) ++a;
   }
   cout << a << " χαρακτήρες" << endl;
   return 0;
}
```

☞ Το πρόγραμμα ζητάει συνέχεια χαρακτήρες και σταματάει όταν δοθούν δύο αστερίσκοι (όταν η fl γίνει 2). Στο τέλος θα εμφανίσει το πλήθος των χαρακτήρων που πληκτρολογήθηκαν μετά από τον πρώτο αστερίσκο (συμπεριλαμβανομένου και του πρώτου).

6.4

```
#include <iostream>
using namespace std;
int main()
{
   int i,j;
   for (i=1;i<=10;++i)
   {
      for (j=1;j<i;++j)
         cout << j << endl;
   }
   return 0;
}
```

```
1
1
2
1
2
3
1
2
3
4
.
.
```

☞ Ο εσωτερικός βρόχος εκτελείται για τιμές του j από το 1 μέχρι το i (το οποίο καθορίζεται από τον εξωτερικό βρόχο).

☞ Επομένως, την πρώτη φορά που το i είναι 1, ο εσωτερικός βρόχος θα εμφανίσει μόνο το 1. Τη δεύτερη φορά που το i είναι 2 ο εσωτερικός βρόχος θα εμφανίσει το 1 και το 2. Την τρίτη φορά θα εμφανίσει το 1, το 2 και το 3 κ.ο.κ. Την τελευταία φορά που το i θα είναι 10, ο εσωτερικός βρόχος θα εμφανίσει τους αριθμούς από το 1 μέχρι το 10.

6.5

```
#include <iostream>
using namespace std;
int main()
{
   char ch;
   for (ch=32;ch<=122;ch++)
      cout << ch;
```

Η πρόταση **cout** εμφανίζει κάθε φορά τον χαρακτήρα που έχει κωδικό ASCII την τιμή της μεταβλητής **ch**.

```
   return 0;
}
```

και για να αλλάζει γραμμή κάθε 10 χαρακτήρες ...

```
#include <iostream>
using namespace std;
int main()
{
   int i=0;
   char ch;
   for (ch=32;ch<=255;ch++)
   {
      cout << ch;
      i++;
      if((i%10 == 0)) cout << endl;
   }
   return 0;
}
```

Όταν η μεταβλητή **i** γίνεται πολλαπλάσιο του 10 (**i % 10 == 0**) τότε προστίθεται αλλαγή γραμμής με την πρόταση **cout<<endl;**.

6.6

```
#include <iostream>
using namespace std;
int main()
{
   int a;
   double sum;
   for (a=1;a<=100;++a)
      sum=sum+1.0/a;
   cout << "Το άθροισμα είναι:" << sum;
   return 0;
}
```

Η πρόταση θα εκτελεστεί 100 φορές και κάθε φορά στη **sum** θα προστίθεται ένα νέο κλάσμα (1/a). Το **a** κάθε φορά θα περιέχει διαφορετική τιμή, από το 1 μέχρι το 100. Το 1.0 χρειάζεται ώστε η παράσταση **1.0/a** να δίνει αποτέλεσμα τύπου **double**.

6.7

```
#include <iostream>
using namespace std;
int main()
{
   int i,j,k=4;
   for (i=1;i<=10;i=i+2)
   {
      k++;
      for (j=1;j<5;++j)
         k=k+2;
   }
   cout << "k=" << k << endl;
   return 0;
}
```

K=49

☞ Ο εξωτερικός βρόχος θα εκτελεστεί πέντε φορές (για i 1,3,5,7,9) ενώ ο εσωτερικός τέσσερις (για j 1,2,3,4).

☞ Η πρόταση k++ θα εκτελεστεί πέντε φορές (επειδή ανήκει μόνο στον εξωτερικό βρόχο) ενώ η k=k+2 θα εκτελεστεί είκοσι (5*4) φορές (επειδή ανήκει και στους δύο βρόχους).

☞ Κάθε φορά που εκτελείται η k++ η μεταβλητή k αυξάνει κατά 1, οπότε τελικά θα αυξηθεί κατά 5 εφόσον θα εκτελεστεί πέντε φορές. Κάθε φορά που εκτελείται η k=k+2 η k αυξάνει κατά 2, οπότε τελικά θα αυξηθεί κατά 40 εφόσον θα εκτελεστεί είκοσι φορές.

☞ Η τελική τιμή του k θα είναι 4 + 5 + 40 = 49 (το 4 είναι η αρχική τιμή της k).

6.8

```
#include <iostream>
using namespace std;
int main()
{
   int i,j=6,k=4;
   i=(k=k+2,j=k*10);
   cout << "i=" << i << " j=" << j << " k=" << k << endl;
   return 0;
}
```

i=60 j=60 k=6

☞ Η παράσταση k=k+2, j=k*10 έχει αποτέλεσμα την εκτέλεση των παραστάσεων k=k+2 και j=k*10 (οπότε στο j καταχωρίζεται το 60). Το αποτέλεσμα της όλης παράστασης είναι η τιμή της τελευταίας (δηλαδή το 60) η οποία καταχωρίζεται στο i.

6.9

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int a;
   do
   {
      a=rand();
      if(a>=100) continue;
      cout << a << endl;
   } while(a!=0);
   return 0;
}
```

Στην μεταβλητή **a** θα καταχωριστεί ένας τυχαίος αριθμός.

Αν ο αριθμός είναι μεγαλύτερος ή ίσος με το 100, τότε η επόμενη **cout** παρακάμπτεται και η διαδικασία επαναλαμβάνεται.

☞ Το πρόγραμμα εμφανίζει τυχαίους αριθμούς μικρότερους από το 100.

☞ Θα σταματήσει όταν ο τυχαίος αριθμός που επιστρέψει η συνάρτηση rand() είναι 0.

☞ Ο τελευταίος αριθμός που θα δούμε είναι το 0. Δεν θα δούμε ποτέ αριθμούς μεγαλύτερους ή ίσους από το 100.

6.10

```
#include <iostream>
using namespace std;
int main()
{
   int ar,y,p,sum=0;
   cout << "Δώσε αριθμό:";
   cin >> ar;
   do
   {
      y=ar % 10;
      p=ar/10;
      sum=sum+y;
      ar=p;
   } while (p!=0);
   cout << "Το άθροισμα των ψηφίων είναι " << sum << endl;
   return 0;
}
```

Στη μεταβλητή **y** καταχωρίζεται το υπόλοιπο της ακέραιας διαίρεσης του αριθμού με το 10, ενώ στην **p** το πηλίκο.

Η διαδικασία συνεχίζεται αντικαθιστώντας κάθε φορά το **ar** με το πηλίκο, μέχρι το πηλίκο να γίνει 0. Τα υπόλοιπα που υπολογίζονται από την παράσταση **ar%10** αποτελούν τα ψηφία του αριθμού και προστίθενται στη **sum**.

☞ Δείτε το παράδειγμα Π6.5 του βιβλίου.

6.11

```
#include <iostream>
using namespace std;
int main()
{
   int i;
   for (i=1;i<=100;i++)
   {
      cout << i << " ";
      if(i%10 == 0) cout << endl;
   }
   return 0;
}
```

Κάθε φορά που το **i** γίνεται πολλαπλάσιο του 10 (10, 20 ...) η **cout<<endl;** προσθέτει μια αλλαγή γραμμής.
Η παράσταση **i%10** υπολογίζει το υπόλοιπο της ακέραιας διαίρεσης του **i** με το 10, και είναι 0 μόνο όταν το **i** είναι πολλαπλάσιο του 10.

6.12

- ☑ Ο βρόχος do-while εκτελείται τουλάχιστον μία φορά.
- ❑ Στην εντολή for οι τρεις προτάσεις που ελέγχουν τη λειτουργία της (αυτές μέσα στην παρένθεση μετά από τη for) πρέπει να σχετίζονται μεταξύ τους.
- ☑ Η πρόταση while(1) έχει αποτέλεσμα τη συνεχή εκτέλεση ενός βρόχου.
- ❑ Ο τελεστής κόμμα (,) επιστρέφει την τιμή της πρώτης παράστασης.
- ☑ Στην εντολή for ο έλεγχος της συνθήκης γίνεται πριν από την εκτέλεση των προτάσεων του βρόχου.

6.13

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   int i;
   float mo,sum,bathmos,max1,max2;
   sum=0;
   for (i=1;i<=10;i++)
   {
      cout << "Δώσε βαθμό " << i << "->";
      cin >> bathmos;
      sum=sum+bathmos;
      if (i==1)
         max1=bathmos;
      else if (i==2)
         max2=bathmos;
      else
      {
         if (bathmos>max1)
         {
            max2=max1;
            max1=bathmos;
         }
         else if (bathmos>max2)
         {
            max2=bathmos;
         }
      }
   }
   mo=sum/100;
```

Η αρχική τιμή των **max1** και **max2** πρέπει να είναι ο πρώτος και ο δεύτερος βαθμός αντίστοιχα.

Στην περίπτωση που ο βαθμός είναι μεγαλύτερος από το **max1**, καταχωρίζεται το **max1** στο **max2** και ο βαθμός στο **max1**.

Στην περίπτωση που ο βαθμός είναι μεγαλύτερος από το **max2**, καταχωρίζεται στο **max2**.

```
   cout << "MO=" << setprecision(2) << fixed << mo << endl;
   cout << "MAX1=" << setprecision(2)<< fixed << max1 << endl;
   cout << "MAX2=" << setprecision(2) << fixed << max2 << endl;
   return 0;
}
```

6.14

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   double fx,x;
   for (x=0.0;x<=1;x=x+0.05)
   {
      fx=pow(x,4)-5*pow(x,2)+3;
      cout << "x=" << x << " fx=" << fx <<endl;
   }
   return 0;
}
```

6.15

```
#include <iostream>
using namespace std;
int main()
{
   int i,j;
   for (i=1,j=10;i<=j;++i,j--)
      cout << i << "-" << j << endl;
   cout << "Τέλος " << i << "-" <<j << endl;
   return 0;
}
```

```
1-10
2-9
3-8
4-7
5-6
Τέλος 6-5
```

6.16

```
#include <iostream>
using namespace std;
int main()
{
   int a=-10,b=5;
   for (cout <<"Αρχή"<<endl;a+b;cout<<"Λέσβος"<< endl)
   {
      a++;
      cout << "Αιγαίο" << endl;
   }
   cout << "Τέλος" << endl;
   return 0;
}
```

❶ ❷ ❹ ❸

```
Αρχή
Αιγαίο
Λέσβος
Αιγαίο
Λέσβος
Αιγαίο
Λέσβος
Αιγαίο
Λέσβος
Αιγαίο
Λέσβος
Τέλος
```

☞ Η ❶ θα εκτελεστεί μόνο μία φορά στην αρχή και θα εμφανίσει τη λέξη «ΑΡΧΗ».

☞ Μετά θα εκτελούνται οι προτάσεις ❷❸❹ συνέχεια, μέχρι η παράσταση ❷ να γίνει ψευδής.

☞ Η επαναληπτική διαδικασία θα σταματήσει όταν η παράσταση a+b γίνει 0 (δηλαδή ψευδής).

6.17

```
#include <iostream>
using namespace std;
int main()
{
   int a=2,b=10,c;
   c=(b>a)+2;
   a=++b,b+c;
   for (a=c;a<10;a=a+3);
   cout << a << "-" << b << "-" << c << endl;
   return 0;
}
```

12-11-3

ΠΡΟΣΟΧΗ ο βρόχος **for** δεν περιέχει καμία πρόταση!

6.18

```
#include <iostream>
using namespace std;
int main()
{
   int i,j,k;
   k=10;
   for (i=1;i<=10;i=i+2)
   {
      k=k-3;
      for (j=1;j<=4;++j)
         k=k+j;
   }
   cout << "k=" << k << " i=" << i << " j=" << j << endl;
   return 0;
}
```

k=45 i=11 j=5

Θα εκτελεστεί 5 φορές, οπότε το **k** θα μειωθεί κατά **15** (3*5).

Κάθε φορά που εκτελείται το εσωτερικό **for**, το **k** θα αυξάνεται κατά 10 (1+2+3+4). Αυτό θα γίνει 5 φορές, οπότε το **k** συνολικά θα αυξηθεί κατά **50**!

Επομένως, στο τέλος η τιμή του **k** θα είναι 45 (10 που είχε αρχικά -15+50)

6.19

```
// Τμήμα a
int a=5,b=10;
while (a<b)
   cout << a++ << "-" << b-- << endl;
// Τμήμα b
char ch='a';
while (ch++<'z')
   cout << ch << endl;
// Τμήμα c
int a=5,b=13;
do
{
   cout << a << endl;
   if (b<12) continue;
   cout << "=====" << endl;
} while (a++!=b--);
```

5-10
6-9
7-8

Θα εμφανίσει τους χαρακτήρες από τον 'b' μέχρι και τον 'z'

b
c
...
z

5
=====
6
=====
7
8
9

6.20

```
// Τμήμα a
int i,j,s=10;
for (i=1;i<=5;i=i+2)
{
   s=s+15;
   for (j=4;j>=0;j--)
      s=s-j;
}
cout << "i=" << i << " j=" << j << " s=" << s << endl;

// Τμήμα b
int i,j;
for (i=1;i<=5;i++)
{
   for (j=1;j<=5;j++)
   {
      if ((i+j)%2==0) continue;
      if (i*j>=16) break;
      cout << i << "-" << j << endl;
   }
   cout << "=====" << endl;
}
cout << "Τέλος" << endl;
```

```
i=7 j=-1 s=25
```

```
1-2
1-4
=====
2-1
2-3
2-5
=====
3-2
3-4
=====
4-1
4-3
=====
5-2
=====
Τέλος
```

6.21

```
#include <iostream>
using namespace std;
int main()
{
   float poso,sum=0,max,min;
   int plithos=0;
   while (sum<100000)
   {
      cout << "Δώσε ποσό :";
      cin >> poso;
      sum=sum+poso;
      plithos++;
      if (plithos==1) max=min=poso;
      if (poso>max) max=poso;
      if (poso<min) min=poso;
   }
   cout << "Πλήθος ατόμων: " << plithos << endl;
   cout << "Συνολικό ποσό: " << sum << endl;
   cout << "Μεγαλύτερο ποσό: " << max << endl;
   cout << "Μικρότερο ποσό: " << min << endl;
   return 0;
}
```

Στην περίπτωση που είναι το πρώτο ποσό που δόθηκε (plithos==1), το αναθέτουμε ως αρχική τιμή για τις μεταβλητές **max** κα **min**.

6.22

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   int a,b;
   float d,r1,r2;
   for (a=-10;a<=10;a++)
```

```
    {
       for (b=-10;b<=10;b++)
       {
          if (a==0 || b==0) continue;
          d=b*b-4*a*3;
          if (d>=0)
          {
             r1=(-b+sqrt(d))/(2*a);
             r2=(-b-sqrt(d))/(2*a);
             cout <<"a="<<a<<" b="<<b<<" r1="<<r1<<" r2="<<r2<<endl;
          }
          else
             cout <<"a="<<a<<" b="<<b<<" δεν έχει πραγματικές ρίζες"<<endl;
       }
    }
    return 0;
}
```

Στην περίπτωση που ένας από τους δύο συντελεστές είναι 0, προχωράμε στην επόμενη επανάληψη.

Αν η διακρίνουσα είναι μεγαλύτερη ή ίση από το 0 υπάρχουν πραγματικές ρίζες, διαφορετικά όχι.

6.23

```
#include <iostream>
using namespace std;
int main()
{
   int i,j;
   for (i=1;i<=5;i++)
   {
      for (j=1;j<=i;j++)
         cout << '*';
      cout << endl;
   }
   return 0;
}
```

Εμφανίζει τόσους αστερίσκους όση η τιμή του **i**.

Αλλάζει γραμμή

6.24

```
#include <iostream>
using namespace std;
int main()
{
   int i,j,gr,st;
   cout << "Δώσε ύψος και πλάτος ορθογωνίου : ";
   cin >> gr >> st;
   for (j=1;j<=st;j++) cout << '*';
   cout << endl;
   for (i=1;i<=gr-2;i++)
   {
      cout << '*';
      for (j=1;j<=st-2;j++)
         cout << ' ';
      cout << '*';
      cout << endl;
   }
   for (j=1;j<=st;j++) cout << '*';
      cout << endl;
   return 0;
}
```

Εμφανίζει την πρώτη γραμμή με αστερίσκους

Εμφανίζει τις ενδιάμεσες γραμμές

Εμφανίζει την τελευταία γραμμή με αστερίσκους

6.25

```
#include <iostream>
using namespace std;
int main()
{
   int i,j,gr;
   do
   {
      cout << "Δώσε ύψος δένδρου : ";
      cin >> gr;
      if (!(gr>=4 && gr<=20))
         cout << "Η τιμή δεν είναι αποδεκτή ξαναδώστην" << endl;
   } while (!(gr>=4 && gr<=20));

   for (i=0;i<=gr-2;i++)
   {
      for (j=1;j<=gr-2-i;j++) cout << ' ';
      for (j=1;j<=i*2+1;j++) cout << '*';
      cout << endl;
   }
   for (j=1;j<=gr-2;j++) cout << ' ';
   cout << '*' << endl;
   return 0;
}
```

Έλεγχος αποδεκτής τιμής για το ύψος του δένδρου.

Εμφάνιση κάθε μιας γραμμής του δένδρου

Εμφάνιση της τελευταίας γραμμής με το μοναδικό αστεράκι!

6.26

```
#include <iostream>
using namespace std;
int main()
{
   string onom,minonom,maxonom;
   float b1,b2,b3,min,max,mo;
   int i;
   max=0;
   min=10;
   for (i=1;i<=10;i++)
   {
      cout << "Δώσε όνομα φοιτητή : ";
      cin >> onom;
      do
      {
         cout << "Δώσε τρεις βαθμούς : ";
         cin >> b1 >> b2 >> b3;
         if (!(b1>=0 && b1<=10 && b2>=0 && b2<=10 && b3>=0 && b3<=10))
            cout << "Μη αποδεκτοί βαθμοί (0~10) ξαναδώστους" << endl;
      } while (!(b1>=0 && b1<=10 && b2>=0 && b2<=10 && b3>=0 && b3<=10));
      mo=(b1+b2+b3)/3;
      cout << "Ο μέσος όρος του " <<onom<<" είναι "<<mo<<endl;
      if (mo>=max)
      {
         max=mo;
         maxonom=onom;
      }
      else if (mo<=min)
      {
         min=mo;
         minonom=onom;
      }
```

Δίνονται αρχικές τιμές στις μεταβλητές **max** και **min** η μικρότερη δυνατή και η μεγαλύτερη δυνατή τιμή, αντίστοιχα.

Στην περίπτωση που κάποιος από τους βαθμούς δεν είναι μέσα στα αποδεκτά όρια (0~10), τότε τους ξαναζητάει.

Στην περίπτωση που ο μέσος όρος είναι μεγαλύτερος από τον τρέχοντα μέγιστο (max), καταχωρίζεται αυτός στη θέση του **max**. Επίσης, στο **maxonom** καταχωρίζεται το όνομα του φοιτητή με τον τρέχοντα μέγιστο μέσο όρο.

Στην περίπτωση που ο μέσος όρος είναι μικρότερος από τον τρέχοντα ελάχιστο (min), καταχωρίζεται αυτός στη θέση του **min**. Επίσης, στο **minonom** καταχωρίζεται το όνομα του φοιτητή με τον τρέχοντα ελάχιστο μέσο όρο.

```
    }
    cout << "Φοιτητής με τον μεγαλύτερο ΜΟ:"<<maxonom<<endl;
    cout << "Φοιτητής με τον μικρότερο ΜΟ:"<<minonom<<endl;
    return 0;
}
```

6.27

```
#include <iostream>
using namespace std;
int main()
{
   int ar,y,p,mx=0,mn=9;
   cout << "Δώσε αριθμό:";
   cin >> ar;
   do
   {
      y=ar % 10;
      p=ar/10;
      if (y>mx) mx=y;
      if (y<mn) mn=y;
      ar=p;
   } while (p!=0);
   cout << "Μικρότερο "<< mn << " Μεγαλύτερο " << mx << endl;
   return 0;
}
```

Οι μεταβλητές **mx** και **mn** χρησιμοποιούνται γα την αποθήκευση του μεγαλύτερου και μικρότερου ψηφίου. Προσοχή στις αρχικές τους τιμές 0 και 9 αντίστοιχα.

Στη μεταβλητή **y** καταχωρίζεται το υπόλοιπο της ακέραιας διαίρεσης του αριθμού με το 10, ενώ στην **p** το πηλίκο. Τα υπόλοιπα που υπολογίζονται από την παράσταση **ar%10** και καταχωρίζονται στη μεταβλητή **y**, αποτελούν τα ψηφία του αριθμού.

Στην περίπτωση που βρεθεί μεγαλύτερο ψηφίο από το **mx** καταχωρίζεται στο **mx**, ενώ αν είναι μικρότερο από το **mn** καταχωρίζεται στο **mn**. Η διαδικασία συνεχίζεται αντικαθιστώντας κάθε φορά το **ar** με το πηλίκο μέχρι το πηλίκο να γίνει 0.

6.28

```
#include <iostream>
using namespace std;
int main()
{
   int ar,i;
   bool protos=true;
   cout << "Δώσε αριθμό:";
   cin >> ar;
   for (i=2;i<ar/2;i++)
   {
      if (ar%i==0)
      {
         if (protos)
            cout << "Ο αριθμός δεν είναι πρώτος και έχει τους παρακάτω
                     διαιρέτες:"<<endl;
            protos=false;
            cout << i << endl;
      }
   }
   if (protos)
      cout << "Ο αριθμός είναι πρώτος" << endl;
   return 0;
}
```

Ελέγχονται όλοι οι ακέραιοι αριθμοί από το 2 μέχρι το μισό του αριθμού που δόθηκε.

Στην περίπτωση που ο αριθμός **i** διαιρεί τον **ar**, σημαίνει ότι ο **ar** δεν είναι πρώτος.

Καταχωρίζεται το γεγονός ότι ο αριθμός δεν είναι πρώτος

Εμφανίζεται ο διαιρέτης

6.29

```
#include <iostream>
using namespace std;
int main()
{
   int ar,y,p,sum=0;
   cout << "Δώσε αριθμό:";
   cin >> ar;
   do
   {
      y=ar % 10;
      p=ar/10;
      sum=sum*10+y;
      ar=p;
   } while (p!=0);
   cout << sum << endl;
   return 0;
}
```

☞ Το πρόγραμμα ζητάει έναν αριθμό και εμφανίζει τον κατοπτρικό του. Δηλαδή, όταν δοθεί ο 12345 θα εμφανίσει 54321 και όταν δοθεί ο 54321 θα εμφανίσει 12345!

Ασκήσεις Κεφαλαίου 7

7.1

```
#include <iostream>
using namespace std;
int main()
{
   float x,y;
   cout << "*************" << endl;
   cin >> x >> y;
   cout << "mo=" << mo(x,y) << endl;
   return 0;
}
float mo(int a, b);
{
   float mesos;
   if (a==0 && b==0)
      return 0;
   else
      mesos=(a+b)/2.0;
   return mesos;
}
```

Οι τυπικές παράμετροι της **mo()** είναι τύπου **int**, επομένως και τα ορίσματα θα έπρεπε να είναι τύπου **int** και όχι **float** όπως τα **x** και **y**. Δεν είναι συντακτικό λάθος αλλά θα έχουμε απώλεια πληροφορίας!

Ο ορισμός της συνάρτησης δεν πρέπει να τερματίζεται με ερωτηματικό. Επίσης θα έπρεπε να υπάρχει προκαταβολική δήλωση της συνάρτησης πριν από τη **main()**.

Δεν δηλώνεται ο τύπος της δεύτερης παραμέτρου **b**.

7.2

```
float calc(int a, float b, float c)
{
   switch(a)
   {
      case 1:
         return b+c;
         break;
      case 2:
         return b*c;
         break;
      case 3:
         return (b+c)/2;
         break;
      default:
         cout << "Αντικανονική κλήση συνάρτησης\n";
         exit(1);
   }
}
```

7.3

```
int total(int ar)
{
   int i,sum=0;
   for (i=1;i<=ar;i++)
      sum=sum+i;
   return sum;
}
```

7.4

☞ Η συνάρτηση δέχεται ως παράμετρο έναν χαρακτήρα.

☞ Αν ο χαρακτήρας δεν είναι λατινικός κεφαλαίος, η συνάρτηση επιστρέφει τιμή 0. Αν είναι λατινικός κεφαλαίος, εμφανίζει όλους τους χαρακτήρες από το 'A' μέχρι τον χαρακτήρα της παραμέτρου και επιστρέφει τιμή 1.

☞ Στη περίπτωση που κληθεί με όρισμα 'J' θα εμφανίσει όλους τους χαρακτήρες από το 'A' μέχρι το 'J΄ και θα επιστρέψει τιμή 1.

☞ Όταν κληθεί με όρισμα '9' δεν θα εμφανίσει τίποτα και θα επιστρέψει τιμή 0.

☞ Τιμή 0 επιστρέφει όταν το όρισμά της δεν είναι λατινικός κεφαλαίος χαρακτήρας.

7.5

```
void print_it1()
{
   int i;
   for(i=1;i<=10;i++)
      cout << "Κατανοήστε τη γλώσσα C++\n";
}
void print_it2(int a)
{
   int i;
   for(i=1;i<=a;i++)
      cout << "Κατανοήστε τη γλώσσα C++\n";
}
```

☞ Η συνάρτηση δηλώνεται τύπου void διότι δεν επιστρέφει καμία τιμή.

7.6

☑ Μια συνάρτηση που δεν επιστρέφει τιμή πρέπει να δηλωθεί ως τύπου void.

☑ Για να χρησιμοποιηθεί μια συνάρτηση βιβλιοθήκης στον κώδικα του προγράμματος, πρέπει να συμπεριλάβουμε με #include το αρχείο κεφαλίδας στο οποίο δηλώνεται.

❑ Αν μια συνάρτηση δεν έχει εντολή return, δεν επιστρέφει ποτέ στον κώδικα που την κάλεσε.

☑ Όταν καλούμε μια συνάρτηση, ο τύπος και το πλήθος των ορισμάτων της πρέπει να είναι αντίστοιχα με τις παραμέτρους της.

☑ Συναρτήσεις που ορίζονται μετά από τη main() πρέπει να δηλωθούν και πριν από αυτήν (με προκαταβολική δήλωση – forward declaration) αν πρόκειται να χρησιμοποιηθούν πριν από τον ορισμό τους.

❑ Η ρητή μετατροπή τύπου μπορεί να αλλάξει μόνιμα τον τύπο μιας μεταβλητής.

❑ Όταν γίνεται αυτόματη μετατροπή τύπου, δεν υπάρχει ποτέ απώλεια πληροφορΙών.

7.7

```
#include <iostream>
#include <cmath>
using namespace std;
double calc(int ar);
int main()
{
   double p;
   p=calc(1000)*2;
   cout << "π=" << p << endl;
```

```
   return 0;
}
double calc(int ar)
{
   int i;
   double p1;
   p1=1.0;
   for(i=2;i<=ar;i=i+2)
      p1=p1*pow(i,2.0)/((i-1)*(i+1));
   return p1;
}
```

7.8

- ☑ Η υπερφόρτωση συναρτήσεων είναι ένα από τα χαρακτηριστικά του πολυμορφισμού.
- ☑ Οι προκαθορισμένες τιμές στις παραμέτρους μιας συνάρτησης επιτρέπουν την κλήση της με μικρότερο αριθμό ορισμάτων.
- ❑ Σε μια υπερφορτωμένη συνάρτηση, η έκδοση που καλείται εξαρτάται *μόνο* από το πλήθος των ορισμάτων κατά την κλήση.
- ☑ Ασάφεια προκύπτει όταν ο μεταγλωττιστής έχει πολλές επιλογές κατά την κλήση μιας υπερφορτωμένης συνάρτησης.
- ☑ Κατά την κλήση μιας υπερφορτωμένης συνάρτησης, η σειρά που ακολουθείται στη διαδικασία εντοπισμού της έκδοσης που συμφωνεί είναι: ακριβής συμφωνία – μετατροπές τύπων – προαγωγές τύπων.

7.9

```
#include <iostream>
using namespace std;
bool disekto(int et);
int main()
{
   int apo,eos,xr;
   cout<<"Από έτος:";
   cin >> apo;
   cout<<"Εως έτος:";
   cin >> eos;
   for (xr=apo;xr<=eos;xr++)
   {
      if (disekto(xr)) cout << xr << " Δίσεκτο" << endl;
   }
   return 0;
}
bool disekto(int et)
{
   if (((et%4 == 0)&& (et%100 != 0))||(et%400 == 0))
      return true;
   else
      return false;
}
```

Η συνάρτηση **disekto()** θα επιστρέψει τιμή **true** (αληθής) στην περίπτωση που το έτος **xr** είναι δίσεκτο, διαφορετικά θα επιστρέψει τιμή **false** (ψευδής) όταν το έτος δεν είναι δίσεκτο.

7.10

```
#include <iostream>
using namespace std;
float dms(float y, float b);
void display(float d);
int main()
{
   float ypsos,baros;
   do
   {
      cout << "Δώσε ύψος και βάρος :";
      cin >> ypsos >> baros;
      if (ypsos==0 || baros==0) break;
      display(dms(ypsos,baros));
   } while (1);
   return 0;
}
float dms(float y, float b)
{
   float d;
   d = b/(y*y);
   return d;
}
void display(float d)
{
   if (d<18.5)
      cout << "Λιποβαρής" << endl;
   else if (d>=18.5 && d<25)
      cout << "Κανονικός " << endl;
   else if (d>=25 && d<30)
      cout << "Υπέρβαρος" << endl;
   else
      cout << "Παχύσαρκος" << endl;
}
```

Προκαταβολικές δηλώσεις συναρτήσεων

Στη συνάρτηση **display()** μεταβιβάζεται ο ΔΜΣ που επιστρέφει η συνάρτηση **dms()**.

7.11

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
float f(float xx);
int main()
{
   float x;
   for (x=0.0;x<=1;x=x+0.05)
   {
      cout << "x=" << setprecision(3) << fixed << x;
      cout << "fx=" << setprecision(3) << fixed << f(x) << endl;
   }
   return 0;
}
float f(float xx)
{
   return pow(xx,4)-10*pow(xx,2)+2;
}
```

Η μεταβλητή **x** παίρνει τιμές από 0 μέχρι 1 με βήματα του 0.05

7.12

```
#include <iostream>
using namespace std;
int myabs(int x)                    // 1η έκδοση με παράμετρο int
{
   if (x<0) return -x;
}
short int myabs(short int x)        // 2η έκδοση με παράμετρο short int
{
   if (x<0) return -x;
}
float myabs(float x)                // 3η έκδοση με παράμετρο float
{
   if (x<0) return -x;
}
double myabs(double x)              // 4η έκδοση με παράμετρο double
{
   if (x<0) return -x;
}
long double myabs(long double x)    // 5η έκδοση με παράμετρο long double
{
   if (x<0) return -x;
}
int main()
{
   short si=-12;
   int i=-100000;
   float f=-2.34;
   double d=-2.1234567;
   long double ld=-4.123456789;
   cout << myabs(si) << endl ;
   cout << myabs(i) << endl ;
   cout << myabs(f) << endl ;
   cout << myabs(d) << endl ;
   cout << myabs(ld) << endl ;
   return 0;
}
```

Η συνάρτηση **myabs()** υπερφορτώνεται πέντε φορές, μία για κάθε διαφορετικό τύπο της παραμέτρου της.

```
12
100000
2.34
2.12346
4.12346
```

Κλήση της **myabs()** με διαφορετικό τύπο ορίσματος.

7.13

```
#include <iostream>
using namespace std;

int myabs(int x) // 1η έκδοση για όρισμα short int ή int
{
   if (x<0) return -x;
}

double myabs(double x) // 2η έκδοση για όρισμα float ή double
{
   if (x<0) return -x;
}

long double myabs(long double x) // 3η έκδοση για παράμετρο long double
{
   if (x<0) return -x;
}
```

Λόγω της προαγωγής τύπου, οι εκδόσεις για παραμέτρους τύπου **short int** και τύπου **float** μπορούν να παραλειφθούν.

Η έκδοση αυτή καλείται είτε το όρισμα είναι τύπου **short int** είτε **int**. Το όρισμα τύπου **short int** προάγεται σε τύπου **int.**

Η έκδοση αυτή καλείται είτε το όρισμα είναι τύπου **float** είτε **double**. Το όρισμα τύπου **float** προάγεται σε τύπου **double.**

```
int main()
{
   short si=-12;
   int i=-100000;
   float f=-2.34;
   double d=-2.1234567;
   long double ld=-4.123456789;
   cout << myabs(si) << endl ;
   cout << myabs(i) << endl ;
   cout << myabs(f) << endl ;
   cout << myabs(d) << endl ;
   cout << myabs(ld) << endl ;
   return 0;
}
```

```
12
100000
2.34
2.12346
4.12346
```

Κλήση της **myabs()** με διαφορετικό τύπο ορίσματος.

7.14

```
long double myabs(long double x)
{
   if (x<0) return -x;
}
```

☞ Η μόνη έκδοση που πρέπει να παραμείνει είναι η έκδοση με παράμετρο τύπου **long double**. Η κλήση της με όρισμα διαφορετικού τύπου έχει ως αποτέλεσμα την μετατροπή του τύπου του ορίσματος σε **long double** χωρίς καμία απώλεια.

7.15

```
#include <iostream>
using namespace std;
float myway(float x)            // 1η έκδοση με μια παράμετρο
{
   return x/2;
}
float myway(float x, float y) // 2η έκδοση με δύο παραμέτρους
{
   return (x+y)/2;
}
int main()
{
   float a,b;
   cin>>a>>b;
   cout <<"Το μισό του"<<a<<" είναι "<<myway(a)<<endl;
   cout <<"Ο μέσος όρος των "<<a<<" και "<<b
        <<" είναι "<<myway(a,b)<<endl;
   return 0;
}
```

7.16

```
#include <iostream>
using namespace std;
void myfunc(short int x)    // 1η έκδοση με παράμετρο short int
{
   cout << "short int\n";
}
void myfunc(long int x)     // 2η έκδοση με παράμετρο long int
{
   cout << "long int\n";
}

void myfunc(double x)       // 3η έκδοση με παράμετρο double
{
```

```
   cout << "double\n";
}
void myfunc(long double x)  // 4η έκδοση με παράμετρο long double
{
   cout << "long double\n";
}
int main()
{
   int i=-100000;
   float f=-2.34;
   long double ld=-4.123456789;
   myfunc(i);
   myfunc(f);
   myfunc(ld);
   return 0;
}
```

Θα γίνει μετατροπή τύπου του ορίσματος, και έτσι θα υπάρχει συμφωνία με πολλές εκδόσεις της συνάρτησης myfunc(), το οποίο θα προκαλέσει ασάφεια.

Θα γίνει προαγωγή τύπου από **float** σε **double** και θα κληθεί η 3η έκδοση της συνάρτησης.

Υπάρχει ακριβής συμφωνία και θα κληθεί η 4η έκδοση της. συνάρτησης.

7.17

```
#include <iostream>
using namespace std;
void display(int ar=10, string onom="Νίκος")
{
   int i;
   for (i=1;i<=ar;i++)   cout << onom << endl;
}
int main()
{
   display();
   display(5);
   display(20,"Πάτροκλος");
   return 0;
}
```

7.18

```
#include <iostream>
using namespace std;
void print_it();
void print_it(char a);
void print_it(string s);
void print_it(char a, char b);
int print_it(int a, int b);
int main()
{
   int a;
   print_it('D');
   print_it('C','F');
   print_it();
   print_it("C++");
   a=print_it(5,10);
   cout << a <<endl;
   return 0;
}
void print_it()  // 1η έκδοση
{
   int i;
   for (i=1;i<=15;i++)
      cout << "=";
   cout << endl;
}
```

Καλείται η 2η έκδοση. Θα εμφανίσει τους χαρακτήρες από το 'A' μέχρι το 'D'

Καλείται η 4η έκδοση. Θα εμφανίσει τους χαρακτήρες από το 'C' μέχρι το 'F'

Καλείται η 1η έκδοση. Θα εμφανίσει 15 '='

Καλείται η 3η έκδοση. Θα εμφανίσει "Η ΓΛΩΣΣΑ C++" και 3 φορές το "ΣΕ ΒΑΘΟΣ".

Καλείται η 5η έκδοση. Θα εμφανίσει όλους τους αριθμούς από το 5 μέχρι το 10 και θα επιστρέψει ως τιμή το άθροισμα τους που είναι 45!

Εμφανίζει 15 '='

```
ABCD
CDEF
==========
Η ΓΛΩΣΣΑ C++
ΣΕ ΒΑΘΟΣ
ΣΕ ΒΑΘΟΣ
ΣΕ ΒΑΘΟΣ
5
6
7
8
9
10
45
```

```
void print_it(char a) // 2η έκδοση
{
   char i;
   for (i='A';i<=a;i++)
      cout << i;
   cout << endl;
}
void print_it(string s) // 3η έκδοση
{
   int i;
   cout << "Η ΓΛΩΣΣΑ " << s << endl;
   for (i=1;i<=3;i++)
      cout << "ΣΕ ΒΑΘΟΣ" << endl;
}
void void print_it(char a, char b) // 4η έκδοση
{
   char i;
   for (i=a;i<=b;i++)
      cout << i;
   cout << endl;
}
int print_it(int a, int b) // 5η έκδοση
{
   int i,s=0;
   for (i=a;i<=b;i++)
   {
      cout << i << endl;
      s=s+i;
   }
   return s;
}
```

Εμφανίζει τους χαρακτήρες από το 'A' μέχρι τον χαρακτήρα της παραμέτρου.

Εμφανίζει το "Η ΓΛΩΣΣΑ", μετά τη συμβολοσειρά της παραμέτρου, και από κάτω τρεις φορές το "ΣΕ ΒΑΘΟΣ".

Εμφανίζει τους χαρακτήρες ξεκινώντας από τον χαρακτήρα της πρώτης παραμέτρου μέχρι τον χαρακτήρα της δεύτερης.

Εμφανίζει όλους τους αριθμούς ξεκινώντας από τον αριθμό της πρώτης παραμέτρου μέχρι τον αριθμό της δεύτερης. Επιστρέφει ως τιμή το άθροισμα των αριθμών που εμφάνισε.

☞ Η συνάρτηση print_it() είναι υπερφορτωμένη 5 φορές!

7.19

```
#include <iostream>
using namespace std;
void print_it(int a=5, char b='A', char c='Z');
int main()
{
   print_it();
   print_it(3);
   print_it(10,'X');
   print_it(20,'D','L');
   return 0;
}
void print_it(int a, char b, char c)
{
   int i;
   char ch;
   for (i=1;i<=a;i++)
      cout << "*";
   cout << endl;
   for (ch=b;ch<=c;ch++)
      cout << ch;
   cout << endl;
}
```

Χρησιμοποιούνται οι προκαθορισμένες τιμές και των τριών παραμέτρων. Θα εμφανίζει 5 αστερίσκους και τους χαρακτήρες από το 'A' έως το 'Z'

Θα εμφανίσει 3 αστερίσκους και τους χαρακτήρες από το 'A' έως το 'Z'

Θα εμφανίσει 10 αστερίσκους και τους χαρακτήρες από το 'X' έως το 'Z'

Θα εμφανίσει 20 αστερίσκους και τους χαρακτήρες από το 'D' έως το 'L'

```
*****
ABCDEFGHIJKLMNOPQRSTUVWXYZ
***
ABCDEFGHIJKLMNOPQRSTUVWXYZ
**********
XYZ
********************
DEFGHIJKL
```

☞ Η συνάρτηση print_it() αρχικά εμφανίζει τόσους αστερίσκους όση είναι η τιμή της πρώτης παραμέτρου της (a). Ακολούθως εμφανίζει τους χαρακτήρες ξεκινώντας από τον χαρακτήρα της δεύτερης παραμέτρου (b) μέχρι τον χαρακτήρα της τρίτης (c).

☞ Στη προκαταβολική δήλωση της συνάρτησης, και οι τρεις παράμετροι έχουν προκαθορισμένες τιμές.

☞ Αν κατά την κλήση δεν οριστεί τιμή για την πρώτη παράμετρο, τότε αυτή παίρνει τη προκαθορισμένη τιμή 5, όταν δεν οριστεί τιμή για την δεύτερη παράμετρο τότε αυτή παίρνει την προκαθορισμένη τιμή 'A', και όταν δεν οριστεί τιμή για την τρίτη παράμετρο τότε αυτή παίρνει την προκαθορισμένη τιμή 'Z'.

7.20

☞ Από τη στιγμή που η δεύτερη παράμετρος b έχει προκαθορισμένη τιμή **πρέπει** και όλες οι υπόλοιπες παράμετροι **μετά από αυτήν** να έχουν προκαθορισμένες τιμές. Στην περίπτωση μας η μόνη παράμετρος που ακολουθεί είναι η παράμετρος c, για την οποία όμως δεν ορίζεται προκαθορισμένη τιμή.

7.21

```
#include <iostream>
using namespace std;
int do_it(int a,int b);
void do_it(int a);
void do_it();
double do_it(double a);
int main()
{
   int m=3,s;
   do_it(2);
   s=do_it(m,4);
   cout<<s<<endl;
   do_it();
   cout << do_it(20.0) << endl;
   return 0;
}
int do_it(int a, int b) // 1η έκδοση
{
   return a+b;
}
void do_it(int a) // 2η έκδοση
{
   int i;
   for (i=1;i<=a;i++)
      cout<<"Αιγαίο"<<endl;
}
void do_it() // 3η έκδοση
{
   int i;
   for (i=1;i<=3;i++)
      cout<<"Λέσβος"<<endl;
}
double do_it(double a) // 4η έκδοση
{
   int i;
   cout << a << endl;
   return a/2;
}
```

```
Αιγαίο
Αιγαίο
7
Λέσβος
Λέσβος
Λέσβος
20
10
```

Καλείται η 2η έκδοση. Θα εμφανίσει 2 φορές τη λέξη "Αιγαίο".

Καλείται η 1η έκδοση. Θα επιστρέψει τον αριθμό 7 (3+4).

Καλείται η 3η έκδοση. Θα εμφανίσει 3 φορές τη λέξη "Λέσβος".

Καλείται η 4η έκδοση διότι το όρισμα είναι τύπου **double**. Θα εμφανίσει το 20 και θα επιστρέψει τιμή 10.

Επιστρέφει ως τιμή το άθροισμα των παραμέτρων.

Εμφανίζει τη λέξη "Αιγαίο" τόσες φορές όσες η τιμή της παραμέτρου **a**.

Εμφανίζει τη λέξη "Λέσβος" 3 φορές.

Εμφανίζει την τιμή της παραμέτρου και επιστρέφει ως τιμή το μισό της!

☞ Η συνάρτηση do_it() είναι υπερφορτωμένη 4 φορές!

7.22

```
#include <iostream>
#include <cmath>
#define pi 3.14159
using namespace std;
double emvadon(double r);
double emvadon(double a, double b);
double emvadon(double a, double b, double c);
int main()
{
   cout<<"Εμβαδόν κύκλου:"<<emvadon(20)<<endl;
   cout<<"Εμβαδόν ορθογωνίου παραλληλογράμμου:"<<emvadon(5,10)<<endl;
   cout<<"Εμβαδόν ορθογωνίου παραλληλεπιπέδου:"<<emvadon(4,6,8)<<endl;
   return 0;
}
double emvadon(double r) // 1η έκδοση
{
   return pi*pow(r,2);
}
double emvadon(double a, double b) // 2η έκδοση
{
   return a*b;
}
double emvadon(double a, double b, double c) // 3η έκδοση
{
   return 2*(a*b+a*c+b*c);
}
```

Καλείται η 1η έκδοση και επιστρέφει ως τιμή το εμβαδόν κύκλου ακτίνας 20!

Καλείται η 2η έκδοση και επιστρέφει ως τιμή το εμβαδόν ορθογωνίου παραλληλογράμμου 5x10!

Καλείται η 3η έκδοση και επιστρέφει ως τιμή το εμβαδόν ορθογωνίου παραλληλεπιπέδου 4x6x8!

Υπολογίζει και επιστρέφει ως τιμή το εμβαδόν κύκλου ακτίνας **r**.

Υπολογίζει και επιστρέφει ως τιμή το εμβαδόν ορθογωνίου παραλληλογράμμου **axb**.

Υπολογίζει και επιστρέφει ως τιμή το εμβαδόν ορθογωνίου παραλληλεπιπέδου **axbxc**.

7.23

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#define pi 3.14159
using namespace std;
double emvadon(double a=0, double b=0, double c=0);
int main()
{
   cout<<emvadon(20)<<endl;
   cout<<emvadon(5,10)<<endl;
   cout<<emvadon(4,6,8)<<endl;
   return 0;
}
double emvadon(double a, double b, double c)
{
   if (a==0 && b==0 && c==0)
   {
      cout << "Ωχ!!! Λάθος" << endl;
      exit(1);
   }
   else if (b==0 && c==0)
   {
      cout << "Κύκλος:";
      return pi*pow(a,2);
   }
```

Όταν η συνάρτηση κληθεί χωρίς όρισμα, όλες οι παράμετροι θα έχουν την προκαθορισμένη τιμή 0. Προφανώς είναι μια λάθος κλήση της συνάρτησης.

Όταν η συνάρτηση κληθεί με ένα όρισμα, οι παράμετροι **b** και **c** θα έχουν την προκαθορισμένη τιμή 0. Στη περίπτωση αυτή πρόκειται για κύκλο!

```
        else if (c==0)
        {
           cout << "Ορθογώνιο:";
           return a*b;
        }
        else
        {
           cout << "Παραλληλεπίπεδο:";
           return 2*(a*b+a*c+b*c);
        }
}
```

Όταν η συνάρτηση κληθεί με δύο ορίσματα, μόνο η παράμετρος **c** θα έχει την προκαθορισμένη τιμή 0. Στη περίπτωση αυτή πρόκειται για ορθογώνιο παραλληλόγραμμο!

Όταν η συνάρτηση κληθεί με τρία ορίσματα, καμία παράμετρος δεν θα έχει την προκαθορισμένη τιμή 0. Στη περίπτωση αυτή πρόκειται για ορθογώνιο παραλληλεπίπεδο!

Ασκήσεις Κεφαλαίου 8

8.1

```
#include <iostream>
using namespace std;
void set();
void pp();
float a;
float mo(int k, int l);
int x,y;
int main()
{
   int x,y;
   x=y=4;
   set();
   a=mo(x,y);
   pp();
   return 0;
}
void set()
{
   int aa;
   x=10;
   y=20;
}
void pp()
{
   int x,y;
   cout << "a=" << a << endl;
}
float mo(int k, int l)
{
   float mesos;
   mesos=(k+l)/2.0;
   return mesos;
}
```

a,x,y

x,y

aa

x,y

k,l,mesos

Η κλήση της **set()** καταχωρίζει στις καθολικές μεταβλητές **x** και **y** τις τιμές 10 και 20 αντίστοιχα.

Η καθολική μεταβλητή **a** θα πάρει τιμή 4 διότι τα **x** και **y** που μεταβιβάζονται στην **mo()** είναι οι τοπικές μεταβλητές **x** και **y** με τιμή 4 και 4 αντίστοιχα.

a=4

8.2

```
#include <iostream>
#include <cstdlib>
using namespace std;
void disp();
int get();
int step;
int main()
{
   step=2;
   disp();
   return 0;
}
void disp()
{
   int a,b;
   cout << "Μια συνάρτηση\n";
   a=get();
   b=get();
   if (a==0 && b==0)
      exit(0);
   else
   {
      int i;
      for (i=a;i<=b;i=i+step) cout << i << endl;
   }
}
int get()
{
   int x;
   cin >> x;
   return x;
}
```

step

a,b

x

Στις μεταβλητές **a** και **b** θα καταχωρηστούν αριθμοί που θα πληκτρολογήσει ο χρήστης (δείτε τη συνάρτηση get()).

Αν οι αριθμοί είναι και οι δύο 0, η συνάρτηση επιστρέφει χωρίς να κάνει απολύτως τίποτα.

Διαφορετικά εμφανίζει ανά 2 τους αριθμούς που δόθηκαν, από τον πρώτο μέχρι τον δεύτερο. Π.χ αν έχουν δοθεί οι αριθμοί 4 και 10 θα εμφανίσει 4,6,8 και 10.

Η συνάρτηση **get()** ζητάει από τον χρήστη να πληκτρολογήσει έναν ακέραιο αριθμό και τον επιστρέφει ως τιμή.

☞ Με απλά λόγια, το παραπάνω πρόγραμμα ζητάει δύο αριθμούς και εμφανίζει ανά 2 όλους τους αριθμούς που δόθηκαν, από τον πρώτο μέχρι τον δεύτερο. Στην περίπτωση που και οι δύο αριθμοί είναι 0, δεν κάνει τίποτα.

8.3

```
#include <iostream>
using namespace std;
int step;
int main()
{
   step=2;
   disp(4);
   return 0;
}
void disp(st)
{
   int a,b,c;
   cout << "Μια συνάρτηση\n";
   a=get();
   c=x;
   b=get();
   if (a==0 && b==0) exit(0);
```

Δεν δηλώνεται ο τύπος της παραμέτρου **st**. Επίσης, θα έπρεπε να υπάρχει και προκαταβολική δήλωση της συνάρτησης πριν από τη **main()**.

Η μεταβλητή **x** είναι άγνωστη στη συνάρτηση **disp()**. Η **x** είναι τοπική μεταβλητή της **get()**.

```
   else
   {
      int i;
      for (i=a;i<=b;i=i+step)
         cout << i << endl;
   }
   cout << "i=" << i << endl;
   cout << "step=" << step << endl;
}
get()
{
   float x;
   cin >> x;
   return x;
}
```

Η μεταβλητή **i** είναι άγνωστη σε αυτό το σημείο. Η **i** έχει εμβέλεια *μόνο* μέσα στη σύνθετη πρόταση της **else**.

Δεν δηλώνεται ο τύπος της συνάρτησης. Επίσης θα έπρεπε να υπάρχει και προκαταβολική δήλωση της συνάρτησης πριν από τη **main()**.

8.4

```
#include <iostream>
using namespace std;
void out(int a, int b, int c);
void func2(int d);
int func1(int x, int y);
int d,e;
int main()
{
   int m=2,n=3,j;
   j=func1(m,n);
   func2(j);
   out(m,n,4);
   return 0;
}
int func1(int x, int y)
{
   d=12;
   return x+y;
}
void func2(int d)
{
   d=20;
   e=d;
}
void out(int a, int b, int c)
{
   cout << "a=" << a << " b=" << b << endl;
   cout << "c=" << a << " d=" << d << " e=" << e << endl;
}
```

a=2 b=3
c=4 d=12 e=20

d,e

m,n,j

x,y

d

a,b,c

Η **func1()** επιστρέφει τιμή 5 (2+3) η οποία καταχωρίζεται στην **j**. Επίσης, η **func1()** καταχωρίζει στην καθολική μεταβλητή **d** το 12.

Η **func2()** καταχωρίζει στην καθολική μεταβλητή **e** το 20.

Η **out(m,n,4)** μεταβιβάζει τις τιμές 2,3 και 4 στις παραμέτρους **a**,**b** και **c** και εμφανίζει τις τιμές τους.
Η **out()** εμφανίζει επίσης τις τιμές των καθολικών μεταβλητών **d** και **e** που είναι 12 και 20 αντίστοιχα.

Η παράμετρος **d** είναι τοπική μεταβλητή της **func2()** και δεν έχει καμία σχέση με την καθολική μεταβλητή **d**.

8.5

```
#include <iostream>
using namespace std;

int x=10;
void out1();
void out2();
void out3();

int main()
{
   int i;
   for(i=1;i<=5;i++) out1();
   for(i=1;i<=5;i++) out2();
   for(i=1;i<=5;i++) out3();
   return 0;
}

void out1()
{
   static int x=4;
   cout << x++ << endl;
}

void out2()
{
   static int x;
   x=4;
   cout << x++ << endl;
}

void out3()
{
   cout << x++ << endl;
}
```

Κάθε φορά που καλείται η **out1()** εμφανίζει τον επόμενο αριθμό από αυτόν που εμφάνισε την προηγούμενη φορά (5,6,7 ....). Την πρώτη φορά που καλείται η **out1()**, εμφανίζει το 4.

Η πρόταση **static int x=4** εκτελείται μόνο στην πρώτη κλήση της **out1()** και η **x** παίρνει αρχική τιμή 4 *μόνο* την πρώτη φορά.

```
4
5
6
7
8
4
4
4
4
4
10
11
12
13
14
```

Κάθε φορά που καλείται η **out2()** εμφανίζει το 4 διότι κάθε φορά η **x** παίρνει την τιμή 4 (x=4).

Η **x** είναι καθολική μεταβλητή με αρχική τιμή 10. Κάθε φορά που εκτελείται η **out3()** εμφανίζει την τιμή της **x** και την αυξάνει κατά 1 (10,11,12 ....).

8.6

- ❑ Η εμβέλεια μιας στατικής μεταβλητής είναι ίδια με την εμβέλεια μιας καθολικής μεταβλητής.
- ☑ Η παράμετρος μιας συνάρτησης είναι και τοπική μεταβλητή της συνάρτησης.
- ☑ Οι στατικές μεταβλητές διατηρούν την τιμή τους ανάμεσα στις κλήσεις μιας συνάρτησης.
- ☑ Μια μεταβλητή μπορεί να δηλωθεί και μέσα στην παρένθεση μιας εντολής for.
- ☑ Μόλις λήξει η εμβέλεια των τοπικών μεταβλητών, αυτές χάνουν ταυτόχρονα και τα περιεχόμενά τους.
- ❑ Όλες οι τοπικές μεταβλητές με το ίδιο όνομα χρησιμοποιούν την ίδια θέση μνήμης.
- ❑ Μια στατική μεταβλητή που ορίζεται μέσα σε μια συνάρτηση μπορεί να προσπελαστεί και από το εξωτερικό τής συνάρτησης.
- ☑ Ο χρόνος ζωής μιας τοπικής στατικής μεταβλητής είναι ίδιος με αυτόν μιας καθολικής μεταβλητής.
- ☑ Μια τοπική μεταβλητή «γεννιέται» με τη δήλωσή της και «πεθαίνει» μόλις λήξει η εμβέλειά της.

8.7

☞ Φυσικά μέσω παραμέτρων! Με τον τρόπο αυτόν η συνάρτηση είναι ανεξάρτητη από το υπόλοιπο πρόγραμμα και δεν μπορεί ούτε να επηρεαστεί, αλλά και ούτε να επηρεάσει κατά λάθος τιμές μεταβλητών του υπόλοιπου προγράμματος.

☞ Κατ' εξαίρεση, και με μεγάλη προσοχή, μπορεί να χρησιμοποιήσουμε καθολικές μεταβλητές για δεδομένα στα οποία έχουν πρόσβαση σχεδόν όλες οι συναρτήσεις του προγράμματός μας, για να αποφύγουμε τη χρήση πολλών παραμέτρων. Αυτό επιτρέπεται κατ' εξαίρεση και μόνο σε μικρά προγράμματα στα οποία ο έλεγχος είναι σχετικά πιο εύκολος.

8.8

```
#include <iostream>
using namespace std;
void out();
int main()
{
   int i;
   for (i=1;i<=15;i++)
      out();
   return 0;
}
void out()
{
   static char ch='A';
   cout << ch << endl;
   ch++;
   if (ch>'E') ch='A';
}
```

Η συνάρτηση **out()** εμφανίζει κάθε φορά τον επόμενο χαρακτήρα ξεκινώντας από το 'A'. Μόλις εμφανίσει το 'E' αρχίζει πάλι από το 'A'.

```
A
B
C
D
E
A
B
C
D
E
A
B
C
D
E
```

8.9

```
void out()
{
   static char ch;
   ch='A';
   cout << ch << endl;
   ch++;
   if (ch>'E') ch='A';
}
```

Θα εμφάνιζε πάντα το 'A', επειδή η αρχική τιμή του **ch** δίνεται σε ξεχωριστή πρόταση και όχι στην πρόταση δήλωσης.

8.10

```
#include <iostream>
using namespace std;
int next_5();
int main()
{
   int i;
   for (i=1;i<=15;i++)
      cout << next_5() << endl;
   return 0;
}
int next_5()
{
   static int a=0;
   a=a+5;
   return a;
}
```

8.11

```
void out()
{
   static int pl=1;
   int i;
   for (i=1;i<=pl;i++)
      cout << '*';
   cout << endl;
   pl++;
   if (pl>20) pl=1;
}
```

Εμφανίζει τόσους αστερίσκους όση είναι η τιμή της μεταβλητής **pl**.

Όταν η μεταβλητή **pl**. ξεπεράσει το 20, ξαναγίνεται 1!

☞ Κάθε φορά που καλείται η συνάρτηση εμφανίζει μια γραμμή από αστερίσκους. Την πρώτη φορά 1, τη δεύτερη 2, την τρίτη 3, κ.ο.κ. Όταν εμφανίσει 20 αστερίσκους, την επόμενη φορά αρχίζει πάλι από το 1, κ.λπ.

8.12

```
#include <iostream>
using namespace std;
int next_first();
int main()
{
   int i;
   for (i=1;i<=100;i++)
      cout << "Επόμενος πρώτος " << next_first() << endl;
   return 0;
}
int next_first()
{
   static int num=0;
   int i,found;
   while (found)
   {
      found=0;
      num=num+1;
      for (i=2;i<=num/2;i++)
         if (num%i==0) found=1;
   }
   return num;
}
```

Κάθε φορά που καλείται η συνάρτηση **next_first()** επιστρέφει τον επόμενο πρώτο αριθμό.

Η **num** διατηρεί την τιμή της σε κάθε κλήση της συνάρτησης. Κάθε φορά αυξάνεται μέχρι να εντοπιστεί ο επόμενος πρώτος αριθμός.

Στην περίπτωση που βρεθεί αριθμός ο οποίος διαιρεί ακριβώς την **num**, η **num** αυξάνεται κατά 1 και η διαδικασία επαναλαμβάνεται. Στην περίπτωση που εντοπιστεί αριθμός που δεν διαιρείται με κανέναν άλλον (από το 2 μέχρι το μισό του) η **found** παραμένει 0 και η διαδικασία σταματάει. Η συνάρτηση επιστρέφει την τιμή της **num**.

8.13

```
#include <iostream>
using namespace std;
int add(int x, int y);
int gin(int x, int y);
void out(int aa, int gg);
int main()
{
   int m,n,a,g;
   cin >> m >> n;
   a=add(m,n);
   g=gin(m,n);
   out(a,g);
   return 0;
}
int add(int x, int y)
```

```
{
   return x+y;
}

int gin(int x, int y)
{
   return x*y;
}

void out(int aa, int gg)
{
   cout << "Αθροισμα=" << aa << endl;
   cout << "Γινόμενο=" << gg << endl;
}
```

8.14

```
#include <iostream>
using namespace std;
void display_stars(int n);
int main()
{
   int ar,psifio;
   cout << "Δώσε αριθμό :";
   cin >> ar;
   while (ar>0)
   {
      psifio=ar%10;
      display_stars(psifio);
      ar=ar/10;
   }
   return 0;
}
void display_stars(int n)
{
   int i;
   for (i=1;i<=n;i++)
      cout << '*';
   cout << endl;
}
```

Στη μεταβλητή **psifio** καταχωρίζονται κάθε φορά ένα-ένα τα ψηφία του αριθμού, με αρχή το λιγότερο σημαντικό.

Η συνάρτηση εμφανίζει στην οθόνη τόσους αστερίσκους όση η τιμή της παραμέτρου της.

8.15

```
char next_char()
{
   static char ch='a';
   if(ch>'z') ch='a';
   return ch++;
}
```

8.16

```
#include <iostream>
using namespace std;
string all_strings(string lex);
int main()
{
   string fr;
   do
   {
      cin >> fr;
      cout << all_strings(fr) << endl;
   } while (fr!="STOP");
```

Η λέξη που πληκτρολογεί ο χρήστης καταχωρίζεται στο **fr**.

Η συμβολοσειρά που πληκτρολόγησε ο χρήστης, μεταβιβάζεται στη συνάρτηση **all_strings()**, η οποία επιστρέφει το σύνολο των χαρακτήρων των συμβολοσειρών που έδωσε μέχρι εκείνη τη στιγμή.

Όταν ο χρήστης πληκτρολογήσει τη λέξη "STOP", η επαναληπτική διαδικασία σταματάει.

```
   return 0;
}
string all_strings(string lex)
{
   static string all="";
   all=all+lex;
   return all;
}
```

Κάθε φορά, στη στατική μεταβλητή (στη πραγματικότητα αντικείμενο κλάσης string) **all** προστίθεται η συμβολοσειρά της παραμέτρου **lex**. Επειδή η **all** είναι στατική, διατηρεί την τιμή της ανάμεσα στις κλήσεις της συνάρτησης.

8.17

```
#include <iostream>
using namespace std;
#define TITLE "Η Γλώσσα C++ σε βάθος"
void info();
float metriseis_imeras(int pl);
void display(float m);
int main()
{
   float therm, mesi,ath;
   int imera=1,am,i;
   info(); // Καλείται η συνάρτηση info() για την εμφάνιση των στοιχείων
   while (1)
   {
      cout <<"Δώσε πλήθος μετρήσεων ημέρας "<<imera<<" :";
      cin >> am;
      if (am<=0) break;
      // Καλείται η συνάρτηση metriseis_imeras() στην οποία μεταβιβάζεται ο αριθμός μετρήσεων am που έδωσε ο χρήστης
      // Η συνάρτηση ζητάει τις θερμοκρασίες των μετρήσεων και επιστρέφει ως τιμή τον μέσο όρο τους
      mesi=metriseis_imeras(am);
      // Καλείται η συνάρτηση display() στην οποία μεταβιβάζεται η μέση τιμή (mesi) των μετρήσεων της ημέρας
      // Η συνάρτηση εμφανίζει τα αποτελέσματα
      display(mesi);
      imera++;
   }
   return 0;
}
void info()
{
   cout<<"########################################"<<endl;
   cout<<TITLE<<endl;
   cout<<"----------------------------------------"<<endl;
}
float metriseis_imeras(int pl)
{
   float ath=0,therm;
   int i;
   for (i=1;i<=pl;i++)
   {
      cout << "Δώσε θερμοκρασία μέτρησης No "<<i<<" :";
      cin >> therm;
      ath=ath+therm;
   }
   return ath/pl;
}
void display(float m)
{
   if (m>=35) cout <<"Καύσωνας "<<m<<" βαθμοί"<<endl;
   else if (m>=25) cout <<"Ζέστη "<<m<<" βαθμοί"<<endl;
   else if (m>=18) cout <<"Κανονική "<<m<<" βαθμοί"<<endl;
   else if (m>=10) cout <<"Κρύο "<<m<<" βαθμοί"<<endl;
   else cout <<"Παγωνιά "<<m<<" βαθμοί"<<endl;
   cout <<"========================================"<<endl<<endl;
}
```

Προκαταβολικές δηλώσεις των συναρτήσεων που ορίζονται μετά από τη **main()**.

Η συνάρτηση **info()** εμφανίζει τα στοιχεία του βιβλίου στην οθόνη.

Η συνάρτηση **metriseis_imeras()** ζητάει από τον χρήστη τόσες τιμές όση η τιμή της παραμέτρου **pl** και επιστρέφει ως τιμή τον μέσο όρο τους.

Η συνάρτηση **display()** εμφανίζει τα αποτελέσματα ανάλογα με την τιμή της παραμέτρου **m**.

Ασκήσεις Κεφαλαίου 9

9.1

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c,*m,*p;
   a=100;
   b=50;
   m=&a;
   p=&b;
   c=*p + *m;
   (*p)++;
   p=m;
   (*p)--;
   return 0;
}
```

Ο δείκτης **m** θα πάρει τη διεύθυνση της **a** και ο **p** τη διεύθυνση της **b**.

Η μεταβλητή **c** θα πάρει τιμή 150 (100+50) αφού η παράσταση ***p** αναφέρεται στη μεταβλητή **a** και η παράσταση ***m** αναφέρεται στη μεταβλητή **b**.

Η παράσταση αυτή θα αυξήσει κατά 1 τη θέση στην οποία δείχνει ο δείκτης **p** (δηλαδή την b). Μετά από την πρόταση αυτή, η τιμή της **b** θα είναι 101.

Στον δείκτη **p** καταχωρίζεται το περιεχόμενο του δείκτη **m** δηλαδή η διεύθυνση της **a**. Τώρα και οι δύο δείκτες δείχνουν στη μεταβλητή **a**. Η πρόταση **(*p)--** θα μειώσει το περιεχόμενο της **a** κατά 1.

Μεταβλητή	Τιμή
a	99
b	51
c	150

9.2

```
#include <iostream>
using namespace std;
int main()
{
   char *p;
   int a=5,b=10;
   p="Τα περιεχόμενα των a και b είναι:";
   cout << p << a << "," << b << endl;
   return 0;
}
```

Θα εμφανίσει το διπλανό αποτέλεσμα. Όταν στην είσοδο του αντικείμενου cout στείλουμε έναν δείκτη σε char, τότε στην έξοδό του εμφανίζει το σύνολο χαρακτήρων από το σημείο που δείχνει ο δείκτης μέχρι τον χαρακτήρα τερματισμού '\0'.

```
Τα περιεχόμενα των a και b είναι:5,10
```

Τι θα εμφάνιζε το πρόγραμμα αν αντί για τη πρόταση

```
cout << p << a << "," << b << endl;
```

είχαμε τη πρόταση

```
cout << p+3 << a << "," << b << endl;
```

Στην περίπτωση αυτή, στο αντικείμενο cout μεταβιβάζεται η διεύθυνση p+3 η οποία δεν δείχνει στην αρχή της φράσης αλλά 3 χαρακτήρες μετά. Έτσι, δεν θα εμφανιστεί ολόκληρη η φράση αλλά οι χαρακτήρες από τον 3ο και μετά.

```
Τα περιεχόμενα των a και b είναι:5,10
```

9.3

```
#include <iostream>
using namespace std;
int main()
{
   char *p,ch;
   int ar=0;
   p="αρνάκι άσπρο και παχύ";
   cin >> ch;
   while(*p!='\0')
   {
      if(*p==ch) ar++;
      p++;
   }
   cout << "Το γράμμα " << ch << " υπάρχει " << ar << " φορές\n";
   return 0;
}
```

Η επαναλαμβανόμενη διαδικασία θα σταματήσει όταν ο δείκτης **p** φτάσει να δείχνει στον χαρακτήρα τερματισμού '\0'.

Η μεταβλητή **ar** μετράει τους χαρακτήρες της συμβολοσειράς που ισούνται με τον χαρακτήρα **ch** που πληκτρολογήθηκε.

Αυξάνει τον δείκτη **p** ώστε να δείχνει στον επόμενο χαρακτήρα.

☞ Ο δείκτης p αρχικά περιέχει τη διεύθυνση της πρώτης θέσης μνήμης που καταλαμβάνει η συμβολοσειρά «αρνάκι άσπρο και παχύ». Δηλαδή δείχνει στον πρώτο χαρακτήρα της συμβολοσειράς.

☞ Να έχουμε υπόψη ότι κάθε συμβολοσειρά τερματίζεται με τον χαρακτήρα τερματισμού '\0'.

9.4

```
#include <iostream>
using namespace std;
int main()
{
   char *p;
   p="αρνάκι άσπρο και παχύ";
   while(*p!='\0')
   {
      if(*p==' ')
         cout << endl;
      else
         cout << *p;
      p++;
   }
   return 0;
}
```

αρνάκι
άσπρο
και
παχύ

Όταν ο δείκτης **p** δείξει σε χαρακτήρα διαστήματος, τότε αλλάζει γραμμή στην οθόνη, διαφορετικά εμφανίζει τον χαρακτήρα που δείχνει ο **p**.

Αυξάνει τον δείκτη **p** ώστε να δείχνει στον επόμενο χαρακτήρα.

9.5

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,c,*p1,*p2,*p3;
   p1=&a;
   p2=&b;
   p3=&c;
   cin >> *p1 >> *p2;
   *p3=*p1 + *p2;
   cout << "Το περιεχόμενο της c είναι : " << *p3 << endl;
   return 0;
}
```

Οι παραστάσεις ***p1** και ***p2** αναφέρονται στις μεταβλητές **a** και **b** αντίστοιχα. Οι αριθμοί που θα πληκτρολογηθούν θα καταχωριστούν στις **a** και **b**.

Το ***p3** αναφέρεται στη μεταβλητή **c**, το ***p1** αναφέρεται στη μεταβλητή **a** και το ***p2** αναφέρεται στη μεταβλητή **b**. Επομένως, η πρόταση αυτή είναι ισοδύναμη με την **c=a+b**;

9.6

Πρόταση	Ερμηνεία
p=&a	*Καταχωρίζει στη μεταβλητή δείκτη **p** τη διεύθυνση της μεταβλητής **a***
*p=23	*Καταχωρίζει στη μεταβλητή που δείχνει ο δείκτης **p** (δηλαδή στην a) το 23.*
m=&b	*Καταχωρίζει στη μεταβλητή δείκτη **m** τη διεύθυνση της μεταβλητής **b***
c= *p + *m	*Προσθέτει το περιεχόμενο της μεταβλητής στην οποία δείχνει ο **p** (της a) με το περιεχόμενο της μεταβλητής στην οποία δείχνει ο **m** (της b) και το αποτέλεσμα καταχωρίζεται στη μεταβλητή **c**.*
m=p	*Καταχωρίζει στη μεταβλητή δείκτη **m** το περιεχόμενο της μεταβλητής δείκτη **p**. Τώρα και οι δύο μεταβλητές δείκτη **m** και **p** δείχνουν στην ίδια μεταβλητή **a**.*
p++	*Αυξάνει το περιεχόμενο της μεταβλητής δείκτη **p** κατά 4! (βλέπε αριθμητική δεικτών).*
(*p)++	*Αυξάνει το περιεχόμενο της μεταβλητής στην οποία δείχνει ο δείκτης **p** (δηλαδή της **a**) κατά 1. Ισοδυναμεί με την πρόταση **a++**;*
m=120	*Λάθος. Σε μια μεταβλητή δείκτη μπορούμε να καταχωρίσουμε μόνο διεύθυνση και όχι μια αριθμητική σταθερά.*

9.7

```
#include <iostream>
using namespace std;
int main()
{
   char *p,*m;
   p="αρνάκι άσπρο και παχύ";
   m=p;
   while (*p!='\0') p++;
   --p;
   while (p>=m) cout << *(p--);
   return 0;
}
```

ύχαπ ιακ ορπσά ικάνρα

Η πρόταση αυτή αυξάνει τον δείκτη **p** μέχρι να δείξει στον χαρακτήρα τερματισμού '\0'.

Ο δείκτης **p** μειώνεται κατά 1 ώστε να δείχνει στον τελευταίο χαρακτήρα.

Η πρόταση αυτή ξεκινάει να εμφανίζει τους χαρακτήρες από τον τελευταίο μέχρι τον πρώτο.

☞ Το πρόγραμμα εμφανίζει τους χαρακτήρες της συμβολοσειράς με αντίστροφη σειρά: από τον τελευταίο στον πρώτο.

☞ Η μεταβλητή m χρησιμοποιείται για την αποθήκευση της αρχικής διεύθυνσης της συμβολοσειράς, επειδή μεταβάλλουμε τον δείκτη p.

9.8

```
   int a,b,c=0,*ptr,*m;
   a=b=10;
1  ptr=&a;
   *ptr=34;
2  m=&b;
3  c=*ptr + *m;
   ptr=m;
4  *ptr=100;
5  m++;
```

Στη μεταβλητή δείκτη **ptr** καταχωρίζεται η διεύθυνση της **a** (1000).

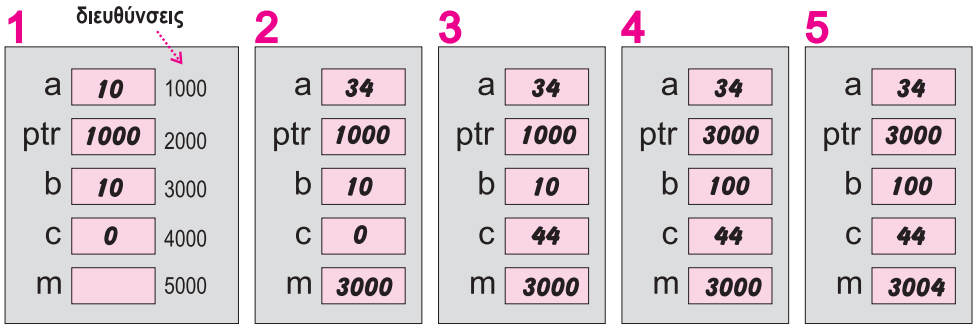
Στη μεταβλητή που δείχνει ο δείκτης **ptr** (δηλαδή στην a) καταχωρίζεται το 34.

Στη μεταβλητή δείκτη **m** καταχωρίτζεαι η διεύθυνση της **b** (3000).

Ισοδυναμεί με **c=a+b**; Στη **c** θα καταχωριστεί το 44

Επειδή η **ptr** περιέχει τώρα τη διεύθυνση της **b**, Ισοδυναμεί **b=100**;

Επειδή ο δείκτης **m** είναι τύπου **int**, θα αυξηθεί κατά 4.



9.9

- ☑ Ένας δείκτης μπορεί να περιέχει *μόνο* τη διεύθυνση μιας θέσης μνήμης.
- ☑ Ο τελεστής αναφοράς * χρησιμοποιείται για την προσπέλαση μιας θέσης μνήμης μέσω ενός δείκτη ο οποίος περιέχει τη διεύθυνσή της.
- ❑ Αν εφαρμόσουμε τον τελεστή ++ σε μια μεταβλητή δείκτη, αυτή αυξάνεται κατά 1.
- ☑ Το μέγεθος μιας μεταβλητής δείκτη εξαρτάται από το σύστημα στο οποίο εκτελείται το πρόγραμμά μας.
- ❑ Μια μεταβλητή δείκτη char και μια μεταβλητή δείκτη int έχουν διαφορετικό μέγεθος.

9.10

```
#include <iostream>
using namespace std;
int space(char *p);
int main()
{
   char *ptr;
   ptr="Η γλώσσα C++ σε βάθος";
   cout << space(ptr) << endl;
   return 0;
}

int space(char *p)
{
   int c=0;
   while (*p!='\0')
   {
      if (*p==' ') c++;
      p++;
   }
   return c;
}
```

4

Η συνάρτηση **space()** μετράει το πλήθος των χαρακτήρων διαστήματος που υπάρχουν στη συμβολοσειρά η διεύθυνση της οποίας μεταβιβάζεται στην παράμετρο **p**. Στη συγκεκριμένη περίπτωση, η συνάρτηση θα επιστρέψει τον αριθμό 4, ο οποίος θα εμφανιστεί στην οθόνη.

9.11

```
int digits(char *p)
{
   int c=0;
   while (*p != '\0')
   {
      if (*p >='0' && *p<='9')
      {
         cout << *p;
         c++;
      }
      p++;
   }
   return c;
}
```

Ο χαρακτήρας εμφανίζεται μόνο στην περίπτωση που είναι αριθμητικός (μεταξύ 0 και 9).

Η μεταβλητή **c** μετράει τους χαρακτήρες που εμφανίστηκαν.

9.12

```
bool is(char *p1, char *p2)
{
   while ((*p1)!='\0' || (*p2)!='\0')
      if (*(p1++)!=*(p2++)) return false;
   return true;
}
```

Η συνάρτηση δέχεται ως παραμέτρους διευθύνσεις σε δύο συμβολοσειρές. Η συνάρτηση επιστρέφει την τιμή **true** (1) στη περίπτωση που οι δύο συμβολοσειρές περιέχουν ακριβώς τους ίδιους χαρακτήρες, ενώ διαφορετικά επιστρέφει τιμή **false** (0).

Για παράδειγμα, αν η συνάρτηση is() κληθεί από τον παρακάτω κώδικα:

```
cout << is("Νίκος","Νίκος") << endl;
cout << is("Νίκας","Νίκος") << endl;
```

θα εμφανίσει την τιμή 1 στην πρώτη κλήση της και την τιμή 0 στη δεύτερη.

9.13

```
#include <iostream>
using namespace std;
int main()
{
   int a=5,b=10,*p1,**p2;
   p1=&a;
   p2=&p1;
   **p2=50;
   p1=&b;
   **p2=*p1+20;
   cout << "a=" << a << endl << "b=" << b << endl;
   return 0;
}
```

a=50
b=30

Στη μεταβλητή στην οποία δείχνει ο δείκτης στον οποίο δείχνει ο **p2** (δηλαδή ο p1) θα καταχωριστεί το 50. Αυτή είναι η μεταβλητή **a**.

Τώρα ο δείκτης **p1** δείχνει στη **b**. Έτσι η παράσταση ****p2** αναφέρεται τώρα στη μεταβλητή **b**.

Η πρόταση αυτή είναι ισοδύναμη με την **b=b+20;**

9.14

Ενέργεια	Πρόταση
Να καταχωρίζει στη μεταβλητή a το 10 χρησιμοποιώντας τον δείκτη p.	`*p = 10;`
Να καταχωρίζει στη μεταβλητή a το 20 χρησιμοποιώντας τον δείκτη m.	`**m = 20;`
Να καταχωρίζει στον δείκτη p τη διεύθυνση της μεταβλητής b, χωρίς όμως να χρησιμοποιείται το όνομά του.	`*m = &b;`
Να αυξάνει τη μεταβλητή b κατά 1 χρησιμοποιώντας τον δείκτη p.	`(*p)++;`
Να μειώνει τη μεταβλητή b κατά 1 χρησιμοποιώντας τον δείκτη m.	`(**m)++;`

Ασκήσεις Κεφαλαίου 10

10.1

```
#include <iostream>
using namespace std;
void hd(double &x, double &y)
{
   cout << x << endl;
   cout << y << endl;
   x=x/2;
   y=2*y;
}
int main()
{
   double i=25,j=50;
   hd(i,j);
   cout << i << ", " << j << endl;
   return 0;
}
```

```
25
50
12.5, 100
```

Η συνάρτηση **hd()** μεταβάλλει τις τιμές των ορισμάτων με τις οποίες καλείται. Διαιρεί το πρώτο όρισμα διά δύο και πολλαπλασιάζει το δεύτερο επί 2.

☞ Επειδή οι δύο παράμετροι της συνάρτησης είναι αναφορικές μεταβλητές, η μεταβίβαση των ορισμάτων γίνεται κατ' αναφορά.

10.2

```
#include <iostream>
using namespace std;
void doit(char *ptr);
int main()
{
   char *p;
   p="ΑΝΑΝΑΣ";
   doit(p);
   cout << p << endl;
   return 0;
}
void doit(char *ptr)
{
   char ch;
   while(*ptr != '\0')
   {
      ch=(*ptr)+1;
      cout << ch;
      ptr++;
   }
   cout << endl;
}
```

```
ΒΕΒΕΒΤ
ΑΝΑΝΑΣ
```

Η συνάρτηση **doit()** δέχεται ως παράμετρο έναν δείκτη σε μια συμβολοσειρά και εμφανίζει τον επόμενο για κάθε έναν από τους χαρακτήρες της.
Δηλαδή αντί για 'Α' εμφανίζει 'Β', αντί για 'Ν' το 'Ξ', κ.ο.κ.

10.3

```
#include <iostream>
using namespace std;
void double_it(double &x)
{
   x=2*x;
}
int main()
{
   double d=12.45;
```

Η συνάρτηση διαθέτει μια αναφορική παράμετρο.

```
   double_it(d);
   cout << d << endl;
   return 0;
}
```

10.4

```
#include <iostream>
using namespace std;
char *last_char(char *ptr);
int main()
{
   char *p;
   p="ΑΝΑΝΑΣ";
   cout << last_char(p) << endl;
   return 0;
}
char *last_char(char *ptr)
{
   while(*ptr != '\0')
   {
      ptr++;
   }
   ptr--;
   return ptr;
}
```

Στη παράμετρο **ptr** μεταβιβάζεται η διεύθυνση μιας συμβολοσειράς.

Διασχίζεται η συμβολοσειρά μέχρι να εντοπιστεί ο χαρακτήρας τερματισμού '\0'.

Μειώνεται η τιμή του δείκτη **ptr** κατά ένα, ώστε να δείχνει στον τελευταίο χαρακτήρα της συμβολοσειράς. Η συνάρτηση επιστρέφει ως τιμή την τιμή του **ptr**.

10.5

- ❑ Μια συνάρτηση που επιστρέφει δείκτη μπορεί να βρίσκεται στα αριστερά του τελεστή =.
- ☑ Η κλήση μιας συνάρτησης με αναφορά μπορεί να γίνει μόνο με μεταβλητές ως ορίσματα.
- ☑ Μια αναφορική μεταβλητή πρέπει οπωσδήποτε να αναφέρεται σε μια άλλη μεταβλητή η οποία θα ορίζεται στην πρόταση δήλωσής της.
- ☑ Μια συνάρτηση που επιστρέφει αναφορά μπορεί να βρίσκεται στα αριστερά του τελεστή =.
- ☑ Η κλήση μιας συνάρτησης με αναφορά είναι ένας έμμεσος τρόπος με τον οποίο η συνάρτηση μπορεί να επιστρέψει περισσότερες από μία τιμές στον κώδικα που την κάλεσε.

10.6

```
#include <iostream>
using namespace std;
inline void double_it(double &x)
{
   x=2*x;
}
int main()
{
   double d=12.45;
   double_it(d);
   cout << d << endl;
   return 0;
}
```

Η μόνη διαφορά είναι η προσθήκη του προσδιοριστικού **inline**.

10.7

```
#include <iostream>
using namespace std;
int f1(int x, int y)
{
   return x*y;
}
int f2(int x, int y)
{
   return x+y;
}
int main()
{
   int (*ptr1) (int x, int y), (*ptr2) (int x, int y);
   int (*ptr) (int x, int y);
   int a=10,b=0;
   bool flag=true;
   ptr1=&f1;
   ptr2=&f2;
   while (a>b)
   {
      if (flag)
         ptr=ptr1;
      else
         ptr=ptr2;
      cout << ptr(a,b) << endl;
      a--;
      b++;
      flag=!flag;
   }
   return 0;
}
```

```
0
10
16
10
24
```

Οι δείκτες **ptr1** και **ptr2** δείχνουν στις συναρτήσεις **f1** και **f2** αντίστοιχα.

Ανάλογα με την τιμή της μεταβλητής **flag**, ο δείκτης **ptr** δείχνει πότε στη συνάρτηση **f1()** και πότε στην **f2()**.

Καλούνται εναλλάξ πότε η συνάρτηση **f1()** και πότε η **f2()**.

10.8

```
#include <iostream>
using namespace std;
void display(int x, int y)
{
   cout << x << endl;
   cout << y << endl;
}
int main()
{
   void (*ptr) (int x, int y);
   ptr=&display;
   ptr(10,20);
   return 0;
}
```

Δηλώνεται ένας δείκτης **ptr** ο οποίος θα δείχνει σε συναρτήσεις τύπου **void** με δύο παραμέτρους τύπου **int**.

Στον δείκτη **ptr** καταχωρίζεται η διεύθυνση της συνάρτησης **display()**. Τώρα ο δείκτης δείχνει σε αυτή τη συνάρτηση.

Με χρήση του δείκτη **ptr** καλείται η συνάρτηση **display()** με ορίσματα 10 και 20.

10.9 :

```
#include <iostream>
using namespace std;
int main()
{
   int a=10,b=20;
   char *p;
   p="AEGEAN SEA";
   .....
   f1(5,8);
```

Καλείται η συνάρτηση **f1()** με ορίσματα δύο ακέραιες σταθερές. Οι τιμές αυτές μεταβιβάζονται στις παραμέτρους της συνάρτησης.

```
   .....
   f2(&a,&b);
   .....
   f3(a,b);
   .....
   cout << f4(p) << endl;
   .....
   cout << *f5(p,'A') << endl;
   return 0;
}
```

Καλείται η συνάρτηση **f2()** με ορίσματα τις διευθύνσεις δύο μεταβλητών οι οποίες μεταβιβάζονται στις παραμέτρους-δείκτες της συνάρτησης.

Καλείται η συνάρτηση **f3()** με ορίσματα δύο μεταβλητές οι οποίες μεταβιβάζονται στις αναφορικές παραμέτρους της συνάρτησης.

Καλείται η συνάρτηση **f4()** με όρισμα έναν δείκτη σε **char** ο οποίος μεταβιβάζεται στην παράμετρο δείκτη της συνάρτησης. Ο χαρακτήρας που επιστρέφει εμφανίζεται στην οθόνη.

Καλείται η συνάρτηση **f5()** με ορίσματα έναν δείκτη σε **char** και έναν χαρακτήρα τα οποία μεταβιβάζονται στις αντίστοιχες παραμέτρους της συνάρτησης. Η συνάρτηση επιστρέφει έναν δείκτη σε **char**. Με χρήση του τελεστή αναφοράς * εμφανίζεται ο χαρακτήρας στον οποίο δείχνει ο δείκτης που επιστρέφει η συνάρτηση!

10.10

```
#include <iostream>
using namespace std;
bool order1(int *x, int *y)
{
   int temp;
   if (*x>=*y)
      return false;
   else
   {
      temp=*x;
      *x=*y;
      *y=temp;
   }
   return true;
}
bool order2(int &x, int &y)
{
   int temp;
   if (x>=y)
      return false;
   else
   {
      temp=x;
      x=y;
      y=temp;
   }
   return true;
}
int main()
{
   int a,b;
   cin >> a>>b;
   order1(&a,&b);
   cout <<a<<"-"<<b<<endl;
   cin >> a>>b;
   order2(a,b);
   cout <<a<<"-"<<b<<endl;
   return 0;
}
```

Η συνάρτηση **order1()** διαθέτει ως παραμέτρους δείκτες στους οποίους μεταβιβάζονται οι διευθύνσεις των μεταβλητών. Σε αυτή την περίπτωση, η μεταβίβαση των ορισμάτων γίνεται με τιμή.

Η πρόσβαση στα ορίσματα-μεταβλητές γίνεται μέσω του τελεστή αναφοράς *. Για παράδειγμα η παράσταση ***x** αναφέρεται στη μεταβλητή στην οποία δείχνει ο δείκτης **x** δηλαδή στη μεταβλητή του πρώτου ορίσματος με το οποίο καλείται η συνάρτηση.

Η συνάρτηση **order2()** διαθέτει ως παραμέτρους δύο αναφορικές μεταβλητές οι οποίες αναφέρονται στα ορίσματα κλήσης της! Σε αυτή την περίπτωση, η μεταβίβαση των ορισμάτων γίνεται με αναφορά.

Οι μεταβλητές **x** και **y** της συνάρτησης **order2()** αναφέρονται στις μεταβλητές των αντίστοιχων ορισμάτων με τα οποία καλείται η συνάρτηση. Αλλαγές στις μεταβλητές αυτές επηρεάζουν άμεσα τις αντίστοιχες μεταβλητές-ορίσματα με τις οποίες κλήθηκε.

Κλήση της συνάρτησης **order1()** με ορίσματα τις διευθύνσεις των μεταβλητών **a** και **b**.

Κλήση της συνάρτησης **order2()** με ορίσματα τις ίδιες τις μεταβλητές **a** και **b**.

Ασκήσεις Κεφαλαίου 11

11.1

```
#include <iostream>
using namespace std;
int main()
{
   int a[10]={5,8,9,4,2,1,7,5,6,2},i,m=0,c=0;
   for (i=0;i<10;i++)
   {
      if (a[i]%2!=0) m++;
      if (a[i]>=5) c++;
   }
   cout << m << c << endl;
   return 0;
}
```

```
5 6
```

☞ Το πρόγραμμα χρησιμοποιεί τη μεταβλητή m ως μετρητή για να μετράει πόσοι από τους αριθμούς του πίνακα δεν είναι ζυγοί (a[i]%2!=0) και τη μεταβλητή c για να μετράει το πλήθος των αριθμών που είναι μεγαλύτεροι ή ίσοι από το 5 (a[i]>=5).

☞ Το αποτέλεσμα θα είναι 5 και 6, αφού στον πίνακα υπάρχουν 5 μονοί αριθμοί και 6 αριθμοί μεγαλύτεροι ή ίσοι από το 5!

11.2

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   int a[100],i,max,thesi;
   for (i=0;i<100;i++)
      a[i]=rand();
   max=a[0];
   thesi=1;
   for (i=0;i<100;i++)
   {
      if (a[i]>max)
      {
         max=a[i];
         thesi=i;
      }
   }
   cout << "Μέγιστος "<<max <<" στη θέση "<<thesi<<endl;
   return 0;
}
```

Ως αρχική τιμή της **max** ανατίθεται η πρώτη θέση του πίνακα.

Ελέγχεται η κάθε θέση του πίνακα. Στην περίπτωση που έχει τιμή μεγαλύτερη της **max**, ανατίθεται η τιμή αυτή στη **max** και αποθηκεύεται στη μεταβλητή **thesi** η θέση όπου εντοπίστηκε.

11.3

```
#include <iostream>
using namespace std;
int main()
{
   int i,pl;
   float bath[10],ath,mo;
   for (i=0;i<10;i++)
   {
      cout << "Δώσε βαθμό φοιτητή No "<< i+1 << " -> ";
      cin >> bath[i];     // Καταχωρίζει στον πίνακα bath τους βαθμούς των 10 φοιτητών
```

```
   }
   ath=0;
   for (i=0;i<10;i++)
      ath=ath+bath[i];    // Στο ath προστίθενται όλοι οι βαθμοί των φοιτητών
   mo=ath/10;             // Υπολογίζει τον μέσο όρο
   for (i=0;i<10;i++)
   if (bath[i]>mo) cout << bath[i] << endl;   // Εμφανίζει τους βαθμούς που είναι μεγαλύτεροι από τον μέσο όρο
   return 0;
}
```

11.4

```
#include <iostream>
using namespace std;
int main()
{
   float a[100],ath,mo;
   int i;
   for (i=0;i<100;i++)
      cin >> a[i];     // Καταχωρίζει στον πίνακα a τους αριθμούς που πληκτρολογεί ο χρήστης
   for (i=0;i<100;i++)
      ath=ath+a[i];    // Στο ath προστίθενται όλοι οι αριθμοί του πίνακα
   mo=ath/100;         // Υπολογίζει τον μέσο όρο
   cout << "Ο μέσος όρος είναι " << mo << endl;
   return 0;
}
```

11.5

- ☑ Ένας πίνακας ορίζει έμμεσα και έναν δείκτη με αρχική τιμή τη διεύθυνση της πρώτης θέσης μνήμης του πίνακα.
- ❑ Σε έναν πίνακα χαρακτήρων μίας διάστασης μπορούν να καταχωριστούν πολλές συμβολοσειρές.
- ☑ Για να έχει μια συνάρτηση πρόσβαση σε έναν πίνακα μίας διάστασης, αρκεί να της μεταβιβάσουμε μόνο τη διεύθυνση του πίνακα.
- ☑ Μπορούμε να αντιμετωπίζουμε ένα αντικείμενο string ως πίνακα χαρακτήρων μίας διάστασης.
- ☑ Οι θέσεις μνήμης ενός πίνακα είναι πάντα συνεχόμενες.

11.6

```
#include <iostream>
using namespace std;
int main()
{
   float a[100];
   int i=0;
   cout << "Δώσε τον αριθμό για τη θέση " << i << " -> " ;
   cin >> a[0]; // Διαβάζει και καταχωρίζει τον πρώτο αριθμό που πληκτρολογεί ο χρήστης
   for (i=1;i<100;i++)
   {
      do
      {
         cout << "Δώσε τον αριθμό για τη θέση " << i << " -> " ;
         cin >> a[i];
         if (!(a[i]>=a[i-1])) cout << "ΛΑΘΟΣ - Πρέπει να είναι μεγαλύτερος ή ίσος από
                                       τον προηγούμενο που έδωσες" << endl;
      } while (!(a[i]>=a[i-1]));
   }
   return 0;
}
```

Αν ο αριθμός δεν είναι αποδεκτός εμφανίζει το μήνυμα λάθους.

Αν ο αριθμός δεν είναι αποδεκτός τον ξαναζητάει.

11.7

```
#include <iostream>
using namespace std;
int main()
{
   int a[10],i,*p;
   for (i=0;i<10;i++)
      *(a+i)=i;
   p=a+9;
   while (p>=a)
      cout << *(p--) << endl;
   return 0;
}
```

```
9
8
7
6
5
4
3
2
1
0
```

☞ Η πρόταση *(a+i) είναι ισοδύναμη με την a[i], οπότε το πρώτο for θα καταχωρίσει στον πίνακα τους αριθμούς από το 0 μέχρι το 9.

☞ Στη συνέχεια, στον δείκτη p καταχωρίζεται η διεύθυνση της τελευταίας θέσης του πίνακα.

☞ Με τον βρόχο while εμφανίζονται μία-μία οι θέσεις που δείχνει ο δείκτης p, ξεκινώντας από την τελευταία και προχωρώντας μέχρι την πρώτη. Επομένως εμφανίζονται τα περιεχόμενα του πίνακα με αντίστροφη σειρά.

11.8

```
#include <iostream>
using namespace std;
int main()
{
   int a[10]={5,8,9,4,2,1,7,5,6,2},i;
   for (i=0;i<10;i++)
      if (*(a+i)%2!=0) (*(a+i))++;
   for (i=0;i<10;i++)
      cout << *(a+i) << endl;
   return 0;
}
```

☞ Η πρόταση *(a+i) είναι ισοδύναμη με την a[i].

☞ ΠΡΟΣΟΧΗ: η πρόταση *(a+i)++ δεν είναι ισοδύναμη με την a[i]++, επειδή ο τελεστής ++ έχει μεγαλύτερη προτεραιότητα από τον τελεστή αναφοράς *. Επομένως, για να είναι ισοδύναμη, πρέπει να γραφεί ως (*(a+i))++, ώστε να εφαρμοστεί πρώτα ο τελεστής αναφοράς.

11.9

```
#include <iostream>
using namespace std;
void do_it(int p[]);
int main()
{
   int a[10]={5,8,9,4,2,1,7,5,6,2},i;
   do_it(a);
   for (i=0;i<10;i++)
      cout << a[i] << endl;
   return 0;
}
void do_it(int p[])
{
   int i;
   for (i=0;i<10;i++)
      if (p[i]%2!=0) p[i]++;
}
```

Στη συνάρτηση μεταβιβάζεται η διεύθυνση του πίνακα **a**.

Η συνάρτηση έχει πρόσβαση στον πίνακα **a** μέσω του ονόματος **p**.

11.10

```
int megaloi(float pin[], float ar)
{
   int i,m=0;
   for (i=0;i<100;i++)
      if (pin[i]>=ar) m++;
   return m;
}
```

11.11

```
#include <iostream>
#include <cstdlib>
using namespace std;

int megaloi(int p1[], int p2[], int timi);

int main()
{
   int i,pl,a[100],b[100],ar;
   for (i=0;i<100;i++)
      a[i]=rand(); // Συμπληρώνουμε τον πίνακα με τυχαίους αριθμούς
   cout << "Δώσε έναν αριθμό:";
   cin >> ar;
   pl=megaloi(a,b,ar);
   cout << "Αντιγράφηκαν " << pl << " αριθμοί" << endl;
   cout << "Πάτησε 1 για να τους δεις ή άλλο αριθμό για έξοδο"<<endl;
   cin >> ar;
   if (ar==1)
   {
      for (i=0;i<pl;i++)
         cout << b[i]<<endl;
   }
   return 0;
}

int megaloi(int p1[], int p2[], int timi)
{
   int i,m=0;
   for (i=0;i<100;i++)
   {
      if (p1[i]>=timi)
      {
         p2[m]=p1[i];
         m++;
      }
   }
   return m;
}
```

Στη συνάρτηση μεταβιβάζονται οι πίνακες **a** και **b** καθώς και ο αριθμός που έδωσε ο χρήστης. Η τιμή που επιστρέφει η συνάρτηση καταχωρίζεται στη μεταβλητή **pl**.

Εμφάνιση των περιεχομένων του πίνακα **b**.

Στον **p2** αντιγράφονται οι τιμές του πίνακα **p1**, με τιμή μεγαλύτερη από την παράμετρο **timi**.

Η συνάρτηση επιστρέφει ως τιμή το πλήθος των τιμών που αντέγραψε.

11.12

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string lex;
   bool einai=true;
   int i,j=0;
   cout << "Δώσε μια λέξη:";
   cin >> lex;
```

```
   while (lex[j]!='\0') ++j;
   --j;
   i=0;
   while (i<j)
   {
      if (lex[i++]!=lex[j--])
      {
         einai=false;
         break;
      }
   }
   if (einai)
      cout << "NAI, είναι παλινδρομική" << endl;
   else
      cout << "OXI, δεν είναι παλινδρομική" << endl;
   return 0;
}
```

Υπολογίζει το πλήθος χαρακτήρων του αντικειμένου **lex**!

Στις μεταβλητές **i** και **j** καταχωρίζονται οι θέσεις του πρώτου και του τελευταίου χαρακτήρα αντίστοιχα.

Συγκρίνονται οι χαρακτήρες των θέσεων **i** και **j**, και το **i** αυξάνει κατά 1 δείχνοντας στον επόμενο χαρακτήρα, ενώ το **j** μειώνεται κατά 1 δείχνοντας στον προηγούμενο. Αν οι χαρακτήρες βρεθούν διαφορετικοί, η μεταβλητή **einai** αποκτά τιμή **false** και η διαδικασία διακόπτεται. Αν όχι, η μεταβλητή **einai** θα παραμείνει με τιμή **true**.

11.13

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string frasi;
   int i,j,k,cnt,epomenos,mikos;
   cout << "Δώσε μια φράση:";
   getline(cin,frasi);
   mikos=0;
   while (frasi[mikos]!='\0') mikos++;
   for (i=0;i<mikos;i++)
   {
      cnt=1;
      epomenos=0;
      for (k=0;k<i;k++)
         if(frasi[i]==frasi[k])
         {
            epomenos=1;
            break;
         }
      if (epomenos) continue;
      for (j=i+1;j<mikos;j++)
         if(frasi[i]==frasi[j]) cnt++;
      cout << "Ο χαρακτήρας "<<frasi[i]<<" υπάρχει "<<cnt<<" φορές"<<endl;
   }
   return 0;
}
```

Υπολογίζει το πλήθος χαρακτήρων του αντικειμένου **frasi**!

Ελέγχει αν ο χαρακτήρας **i** βρίσκεται στις προηγούμενες θέσεις του πίνακα (από τη 0 μέχρι την i-1)

Στην περίπτωση που ο χαρακτήρας έχει ήδη εξεταστεί, προχωρούμε στον επόμενο παρακάμπτοντας τον υπόλοιπο κώδικα του βρόχου.

Μετράει τις φορές που εμφανίζεται ο χαρακτήρας στις επόμενες θέσεις της φράσης.

11.14

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
   char lex[30]="Η γλώσσα C++ σε βάθος";
   int i,s=0;
   for (i=0;i<strlen(lex);i++)
   {
      if (lex[i]==lex[i+1]) s=i;
      if (lex[i]==' ') lex[i]='-';
```

Η-γλώσσα-C++-σε-βάθος
++-σε-βάθος

Στην περίπτωση που εντοπιστεί διπλός συνεχόμενος χαρακτήρας κρατάει τη θέση του. Επομένως, θα κρατήσει τη θέση του τελευταίου διπλού χαρακτήρα.

Στην περίπτωση που εντοπιστεί χαρακτήρας διαστήματος, τον αντικαθιστά με παύλα.

```
   }
   cout << lex << endl;
   cout << lex+s << endl;
   return 0;
}
```

Εμφανίζει τη φράση από τον διπλό χαρακτήρα και μετά.

11.15

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string frasi;
   int i=0,cnt=0;
   cout << "Δώσε μια φράση:";
   getline(cin,frasi);
   while (frasi[i]!='\0')
   {
      if (frasi[i]==' ') cnt++;
      i++;
   }
   cout << "Ο χαρακτήρας του διαστήματος υπάρχει "<<cnt<<" φορές"<<endl;
   return 0;
}
```

Μόλις φτάσουμε στο τέλος της φράσης, η επαναληπτκή διαδικασία σταματάει.

Στην περίπτωση που ο χαρακτήρας είναι το κενό διάστημα, αυξάνεται κατά 1 η τιμή της **cnt**.

11.16

```
#include <iostream>
#include <string>
using namespace std;
bool yparxei(string p[],int pos);
int main()
{
   string o[50];
   int i,j;
   for (i=0;i<50;i++)
   {
      do
      {
         cout << "Δωσε όνομα "<<i+1<<" ->";
         getline(cin,o[i]);
         if (yparxei(o,i))
            cout << "Το όνομα υπάρχει ήδη - δώστε άλλο" << endl;
         else
            break;
      } while (1);
   }
   return 0;
}

bool yparxei(string p[],int pos)
{
   bool brika=false;
   int j;
   for (j=0;j<pos-1;j++)
      if (p[j]==p[pos]) brika=true;
   return brika;
}
```

Στη συνάρτηση **yparxei()**, μεταβιβάζεται ο πίνακας **o** καθώς και η θέση **i** στην οποία καταχωρίστηκε το τελευταίο όνομα.

Όσο το όνομα που δίνουμε υπάρχει, δεν γίνεται αποδεκτό και το ξαναζητάει.

Ελέγχει αν στις προηγούμενες θέσεις του πίνακα (από την 0 μέχρι τη pos-1), υπάρχει το όνομα που βρίσκεται στη θέση **pos**. Αν το εντοπίσει επιστρέφει true, διαφορετικά false.

11.17

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string lex;
   int i,ar=0,s=0;
   lex="Η γλώσσα C++ σε βάθος";
   while (lex[ar]!='\0') ar++;
   cout << ar << endl;
   for (i=0;i<ar;i++)
      if (lex[i]=='+') lex[i]='-';
   cout << lex << endl;
   return 0;
}
```

21
Η γλώσσα C-- σε βάθος

Μετράει το πλήθος των χαρακτήρων του αντικειμένου **lex**, και το καταχωρίζει στη μεταβλητή **ar**. Εμφανίζει το περιεχόμενό της που είναι 21!

Διασχίζει όλους τους χαρακτήρες του αντικειμένου **lex** και αντικαθιστά όλα τα '+' με '-'.

11.18 :

- Η πρόταση char lex5="ΜΥΤΙΛΗΝΗ"; είναι λάθος γιατί δεν μπορούμε σε μια μεταβλητή τύπου char να καταχωρίσουμε μια συμβολοσειρά.
- Η πρόταση lex2="ΠΟΛΙΤΙΣΜΙΚΗ ΤΕΧΝΟΛΟΓΙΑ"; είναι λάθος γιατί δεν μπορούμε σε έναν πίνακα χαρακτήρων να καταχωρίσουμε μια συμβολοσειρά με τον τελεστή ανάθεσης. Αυτό μπορεί να γίνει μόνο στη πρόταση δήλωσης του πίνακα όπως στην τρίτη και τέταρτη πρόταση του κώδικα. Αν θέλουμε να το κάνουμε εκ των υστέρων, θα πρέπει να χρησιμοποιήσουμε τη συνάρτηση strcpy().

11.19

```
int lexeis(string str)
{
   int cnt=0,i=0;
   while (str[i]!='\0')
   {
      // Αυξάνει το i όσο δείχνει σε χαρακτήρες διαστήματος
      while (str[i]==' ' && str[i]!='\0') i++;
      // Διακόπτει την επαναληπτική διαδικασία όταν φτάσουμε στο τέλος
      if (str[i]=='\0') break;
      // Αυξάνει το i όσο δεν δείχνει σε χαρακτήρες διαστήματος
      while (str[i]!=' ' && str[i]!='\0') i++;
      cnt++;      // Αυξάνει τον μετρητή λέξεων κατά 1
   }
   return cnt;   // Επιστρέφει ως τιμή τον μετρητή λέξεων
}
```

Στη συνάρτηση μεταβιβάζεται ένα αντικείμενο κλάσης **string**.

Η επαναληπτική διαδικασία συνεχίζεται όσο δεν έχουμε φτάσει στο τέλος της συμβολοσειράς.

11.20

```
#include <iostream>
using namespace std;
int main()
{
   int a[100][20],max;
   int i,j;
   .....
   for (i=0;i<100;i++)
   {
      max=a[i][0];
      for (j=0;j<20;j++)
      {
         if (a[i][j]>max)
            max=a[i][j];
```

Η αρχική τιμή της **max** είναι κάθε φορά η τιμή της πρώτης θέσης κάθε γραμμής (i).

Στη **max** τελικά καταχωρίζεται η μεγαλύτερη από τις τιμές των θέσεων μνήμης της γραμμής **i**.

```
      }
      cout << "Μέγιστος σειράς " << i << "=" << max;
   }
   return 0;
}
```

☞ Στο παραπάνω πρόγραμμα, οι μέγιστες τιμές κάθε γραμμής απλώς εμφανίζονται στην οθόνη. Αν θέλαμε να υπάρχουν και κάπου καταχωρισμένες, θα έπρεπε να χρησιμοποιήσουμε έναν πίνακα 100 θέσεων, π.χ max[100], όπου σε κάθε θέση μνήμης του πίνακα max[] θα αποθηκεύαμε τη μέγιστη τιμή της αντίστοιχης γραμμής του πίνακα a[].

11.21

```
#include <iostream>
using namespace std;
int main()
{
   int b[50][3],i,j,pl_ap=0,pl_ar=0;
   float sum,mo;
   for (i=0;i<50;i++)
   {
      cout << "Μαθητής "<<i+1<<endl;
      cout << "--------------" << endl;
      for (j=0;j<3;j++)
      {
         cout << "Μάθημα " <<j+1 << ":";
         cin >> b[i][j];
      }
      cout << "==============" << endl;
   }
   for (i=0;i<50;i++)
   {
      sum=0;
      for (j=0;j<3;j++) sum=sum+b[i][j];
      mo=sum/3;
      if (mo>=18.5) pl_ar++;
      if (mo<9.5) pl_ap++;
   }
   cout << "Αποτυχόντες: " << pl_ap << "Ποσοστό %=" << pl_ap*100/50.0;
   cout << "Αριστούχοι: " << pl_ar << "Ποσοστό %=" << pl_ar*100/50.0;
   return 0;
}
```

Οι βαθμολογίες θα καταχωριστούν σε έναν πίνακα 50 γραμμών και 3 στηλών.

Καταχώριση των βαθμών στον πίνακα **b**.

Υπολογισμός του μέσου όρου κάθε μαθητή.

Καταμέτρηση του πλήθους των αριστούχων και των αποτυχόντων.

11.22

```
#include <iostream>
using namespace std;
int main()
{
   int xreosi[12][31],i,j,max;
   for (i=0;i<12;i++)
   {
      cout << "Μήνας "<<i+1<<endl;
      cout << "-----------------" << endl;
      for (j=0;j<31;j++)
      {
         cout << "Χρέωση "<<j+1<<"/"<<i+1<<":";
         cin >> xreosi[i][j];
      }
   }
   max=xreosi[0][0];
   for (i=0;i<12;i++)
```

Θεωρούμε ότι όλοι οι μήνες έχουν 31 ημέρες. Αν θέλαμε να το κάναμε απόλυτα σωστά, το πρόγραμμα θα γινόταν αρκετά πιο περίπλοκο!

Καταχώριση των ημερήσιων χρεώσεων στον πίνακα **xreosi**.

```
      for (j=0;j<31;j++)
         if (xreosi[i][j]>max) max=xreosi[i][j];
   cout << "Ημέρες με τη μεγαλύτερη χρέωση " << max << endl;
   cout << "-----------------------------------" << endl;
   for (i=0;i<12;i++)
      for (j=0;j<31;j++)
         if (xreosi[i][j]==max) cout << j+1 << "/" << i+1 << endl;
   return 0;
}
```

Υπολογισμός της μέγιστης χρέωσης

Εμφάνιση των ημερών με τη μέγιστη χρέωση.

11.23

```
#include <iostream>
using namespace std;
int main()
{
   int a[3][5]={5,8,9,4,2,
                1,7,5,6,2,
                4,3,0,3,5};
   int i,j,s=0;
   for (i=0;i<3;i++)
      for (j=0;j<5;j++)
         if ((i+j)%2==0) s=s+a[i][j];
   cout << s << endl;
   return 0;
}
```

38

5	8	9	4	2
1	7	5	6	2
4	3	0	3	5

☞ Το πρόγραμμα θα εμφανίζει το άθροισμα των περιεχομένων του πίνακα που βρίσκονται στις θέσεις μνήμης όπου το άθροισμα της γραμμής και της στήλης είναι ζυγός αριθμός. Δηλαδή των a[0][0], a[0][2], a[0][4], a[1][1], κ.ο.κ.

11.24

```
int sum(int pin[][20][5])
 {
   int i,j,k,ss;
   ss=0;
   for (i=0;i<10;i++)
      for (j=0;j<20;j++)
         for (k=0;k<5;k++)
            ss=ss+pin[i][j][k];
   return ss;
}
```

Τα **i**, **j**, και **k** μεταβάλλουν την πρώτη, τη δεύτερη, και την τρίτη διάσταση αντίστοιχα.

Στη μεταβλητή **ss** αθροίζονται όλες οι θέσης μνήμης του πίνακα **pin**.

11.25

```
void synola(int p[][5], int s[])
{
   int i,j,sum;
   for (i=0;i<10;i++)
   {
      sum=0;
      for (j=0;j<5;j++)  sum=sum+p[i][j];
      s[i]=sum;
   }
}
```

Η συνάρτηση πρέπει να έχει δύο παραμέτρους. Στην πρώτη μεταβιβάζεται ο πίνακας 10 × 5 και στη δεύτερη ένας άλλος πίνακας 10 θέσεων, στον οποίο η συνάρτηση θα καταχωρίσει τα αθροίσματα των γραμμών.

Το άθροισμα κάθε γραμμής του πίνακα **p** (της πρώτης παραμέτρου) καταχωρίζεται στην αντίστοιχη θέση του πίνακα **s** (της δεύτερης παραμέτρου).

11.26

☞ Η συνάρτηση συμπληρώνει έναν πίνακα δύο διαστάσεων 35x10 με αριθμούς. Ο αριθμός που καταχωρίζεται σε μια θέση μνήμης του πίνακα προκύπτει από το γινόμενο του αριθμού γραμμής και του αριθμού στήλης που ανήκει η θέση. Π.χ στη θέση pin[30][5] θα καταχωριστεί ο αριθμός 150 (30*5).

☞ Η συνάρτηση πρέπει να κληθεί με όρισμα έναν πίνακα τύπου int, δύο διαστάσεων 35x10.

☞ Αν η δήλωση int pin[][10] ήταν int pin[5][10], η συνάρτηση θα δούλευε σωστά διότι αγνοεί πλήρως την τιμή της πρώτης διάστασης της παραμέτρου.

11.27

☑ Σε ένα πίνακα μίας διάστασης με αντικείμενα string μπορούν να καταχωριστούν πολλές συμβολοσειρές.

☑ Για να προσπελάζουμε τους μεμονωμένους χαρακτήρες κάθε συμβολοσειράς σε έναν πίνακα μίας διάστασης τύπου string, μπορούμε να τον αντιμετωπίσουμε ως πίνακα δύο διαστάσεων.

☑ Για να έχει μια συνάρτηση πρόσβαση σε έναν πίνακα με περισσότερες από μία διαστάσεις, πρέπει να γνωρίζει, εκτός από τη διεύθυνση του πίνακα, και τις τιμές των διαστάσεών του, εκτός από την πρώτη.

❑ Δεν μπορούμε να απεικονίσουμε με εποπτικό τρόπο πίνακες με περισσότερες από τρεις διαστάσεις.

11.28

```
#include <iostream>
#include <string>
#include <iomanip>
#define PLHTHOS 100 // Αλλάξτε τη σταθερά PLHTHOS σε μικρότερο αριθμό ώστε να δοκιμάσετε το πρόγραμμα
using namespace std;

void display_synolo_paravaseon(string ak[], int pr[][12],int sp[]);
void display_10_top(string ak[], int sp[]);
void display_minimum(string ak[], int sp[]);

int main()
{
   string ar_kykl[PLHTHOS];
   int par[PLHTHOS][12],syn_par[PLHTHOS],i,j;
   for (i=0;i<PLHTHOS;i++)
   {
      cout << "Δώσε αριθμό κυκλοφορίας αυτοκινήτου "<<i+1 << ":";
      cin>>ar_kykl[i];
   }

   for (i=0;i<PLHTHOS;i++)
   {
      cout << "Αυτοκίνητο "<<ar_kykl[i]<<endl;
      cout << "-----------------"<< endl;
      for (j=0;j<12;j++)
      {
         cout << "Παραβάσεις μηνός "<<j+1<<":";
         cin>>par[i][j];
      }
   }
   cout << "============================" << endl;
   display_synolo_paravaseon(ar_kykl,par,syn_par);
   cout << "============================" << endl;
   display_10_top(ar_kykl,syn_par);
   cout << "============================" << endl;
   display_minimum(ar_kykl,syn_par);
   return 0;
}
```

Στον πίνακα **ar_kykl** θα καταχωρίζονται οι αριθμοί κυκλοφορίας των αυτοκινήτων (δεν ξεχνάμε ότι περιέχουν και χαρακτήρες π.χ. ΥΥΑ3456). Στον πίνακα **par** καταγράφονται οι παραβάσεις και στον πίνακα **syn_par** θα υπολογίζονται οι συνολικές παραβάσεις όλου του έτους.

Ανάγνωση των αριθμών κυκλοφορίας των αυτοκινήτων από το πληκτρολόγιο.

Ανάγνωση των παραβάσεων όλων των αυτοκινήτων για κάθε μήνα, από το πληκτρολόγιο.

```
void display_synolo_paravaseon(string ak[], int pr[][12], int sp[])
{
   int i,j,sum;
   cout << "Αρ. Κυκλ.    Παραβ"<<endl;
   cout << "----------   -----"<<endl;
   for (i=0;i<PLHTHOS;i++)
   {
      sum=0;
      for (j=0;j<12;j++)
      {
         sum=sum+pr[i][j];
      }
      sp[i]=sum;
      cout << setw(10)<<ak[i]<<setw(5)<<sum<<endl;
   }
}
void display_10_top(string ak[], int sp[])
{
   int i;
   cout << "TOP-10"<<endl;
   cout << "Αρ. Κυκλ.    Παραβ"<<endl;
   cout << "----------   -----"<<endl;
   for (i=0;i<PLHTHOS;i++)
   {
      if (sp[i]>10) cout << setw(10)<<ak[i]<<setw(5)<< sp[i]<<endl;
   }
}
void display_minimum(string ak[], int sp[])
{
   int i,min;
   cout << "Minimum Παραβάσεις"<<endl;
   cout << "Αρ. Κυκλ.    Παραβ"<<endl;
   cout << "----------   -----"<<endl;
   min=sp[0];
   for (i=0;i<PLHTHOS;i++)
      if (sp[i]<min) min=sp[i];
   for (i=0;i<PLHTHOS;i++)
      if (sp[i]==min) cout << setw(10)<<ak[i]<<setw(5)<< sp[i]<<endl;
}
```

11.29

```
void polla_nai(char apa[][20])
{
   int i,j,nai,oxi,syn=0;
   for (i=0;i<1000;i++)
   {
      nai=oxi=0;
      for (j=0;j<20;j++)
         if (apa[i][j]=='N')
            nai++;
         else if (apa[i][j]=='O')
            oxi++;
      if (nai>oxi) syn++;
   }
   cout <<"Πλήθος :"<<syn<<endl;
}
```

Στις μεταβλητές **nai** και **oxi** δίνεται αρχική τιμή 0 πριν από το πέρασμα κάθε γραμμής.

Η μεταβλητή **nai** μετράει τους χαρακτήρες 'N' κάθε γραμμής.

Η μεταβλητή **oxi** μετράει τους χαρακτήρες 'O' κάθε γραμμής.

Η μεταβλητή **syn** αυξάνει κατά 1 όταν τα 'N' είναι περισσότερα από τα 'O'.

☞ Η μεταβλητή nai μετράει το πλήθος των χαρακτήρων 'N' κάθε γραμμής και η μεταβλητή oxi μετράει το πλήθος των χαρακτήρων 'O' κάθε γραμμής. Τέλος η μεταβλητή syn μετράει το πλήθος των γραμμών όπου η τιμή της nai είναι μεγαλύτερη από την τιμή της oxi.

11.30

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string onomata[10];
   char ch;
   int i;
   for (i=0;i<10;i++)
   {
      cout << "Δώσε όνομα "<< i+1 << endl;
      cin >> onomata[i];
   }
   cout << "Δώσε αρχικό χαρακτήρα:";
   cin >> ch;
   cout << "Τα ονόματα που αρχίζουν από "<<ch<<" είναι:"<<endl;
   for (i=0;i<10;i++)
   {
      if (onomata[i][0]==ch) cout<<onomata[i];
   }
   return 0;
}
```

Καταχώριση των 10 ονομάτων στον πίνακα **onomata**.

Ζητείται από τον χρήστη ο αρχικός χαρακτήρας.

Ελέγχεται ο πρώτος χαρακτήρας κάθε ονόματος.

11.31

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string fraseis[10];
   string (*ptr)[10];
   int i,j,k=0;
   for (i=0;i<10;i++)
   {
      cout << "Δώσε φράση "<<i+1<<"->";
      getline(cin,fraseis[i]);
   }
   ptr=&fraseis;
   for (i=0;i<10;i++)
   {
      j=0;
      while ((*ptr)[i][j]!='\0')
      {
         if ((*ptr)[i][j]==' ') k++;
         j++;
      }
   }
   cout << "Υπάρχουν "<<k<<" κενά διαστήματα"<<endl;
   return 0;
}
```

Δηλώνεται ένας δείκτης σε πίνακα κλάσης **string** 10 θέσεων με όνομα **ptr**.

Στο δείκτη **ptr** καταχωρίζεται η διεύθυνση του δείκτη του πίνακα **fraseis**.

Η πρόσβαση στον πίνακα **fraseis** γίνεται μέσω του δείκτη **ptr**.

11.32

```
#include <iostream>
using namespace std;
int main()
{
   int i,a[100],b[100],k=0,s=0;
   int (*ptr)[100];
   ptr=&a; // Ο δείκτης δείχνει στον πίνακα a
```

```
   for (i=0;i<100;i++)
      (*ptr)[i]=rand();
   ptr=&b; // Ο δείκτης δείχνει τώρα στον πίνακα b
   for (i=0;i<100;i++)
      (*ptr)[i]=i;
   for (i=0;i<100;i++)
      s=s+(*ptr)[i];
   ptr=&a; // Ο δείκτης δείχνει ξανά στον πίνακα a
   for (i=0;i<100;i++)
      if ((*ptr)[i]>1000) k++;
   cout << s << " " << k << endl;
   return 0;
}
```

Συμπληρώνει τον πίνακα **a** (μέσω του δείκτη ptr), με 100 τυχαίους αριθμούς.

Συμπληρώνει τον πίνακα **b** (μέσω του δείκτη ptr ο), με τους αριθμούς από το 0 μέχρι το 99.

Υπολογίζει το άθροισμα των αριθμών του πίνακα **b**.

Υπολογίζει το πλήθος των αριθμών του πίνακα **a** που είναι μεγαλύτεροι από το 1000.

11.33

```
int *ptr[100];
```

☞ Δημιουργεί έναν πίνακα δεικτών σε μεταβλητές int, 100 θέσεων!

```
int (*ptr)[100];
```

☞ Δημιουργεί έναν δείκτη σε πίνακες int 100 θέσεων!

```
int (*ptr[5])[100];
```

☞ Δημιουργεί έναν πίνακα δεικτών πέντε θέσεων, σε πίνακες int 100 θέσεων!

```
int (*ptr)[100][20];
```

☞ Δημιουργεί έναν δείκτη σε πίνακες int 100 × 20!

```
int (*ptr[5])[100][20];
```

☞ Δημιουργεί έναν πίνακα δεικτών πέντε θέσεων, σε πίνακες int 100 × 20!

11.34

☞ Στη συνάρτηση μεταβιβάζεται ένας πίνακας δεικτών σε πίνακες int 10 θέσεων. Ο πίνακας αυτός διαθέτει 5 θέσεις.

☞ Η συνάρτηση, μέσω των δεικτών, καταχωρίζει στους 5 πίνακες στους οποίους «δείχνουν» από 10 τυχαίους αριθμούς.

Το πρόγραμμα που ακολουθεί χρησιμοποιεί τη συνάρτηση fill για να συμπληρώσει τους πίνακες a, b, c, d, και e με τυχαίους αριθμούς. Στη συνάρτηση μεταβιβάζεται ο πίνακας δεικτών σε πίνακες p, ο οποίος περιέχει δείκτες προς τους πέντε παραπάνω πίνακες.

```
#include <iostream>
using namespace std;
int main()
{
   int a[10],b[10],c[10],d[10],e[10];
   int (*p[5])[10];
   int i;
   p[0]=&a;
   p[1]=&b;
   p[2]=&c;
   p[3]=&d;
   p[4]=&e;
   fill(p);
   for (i=0;i<10;i++)
        cout <<setw(8)<<a[i]<<setw(8)<<b[i]<<setw(8)<<c[i]
             <<setw(8)<<d[i]<<setw(8)<<e[i]<<endl;
   return 0;
}
```

Στη συνάρτηση μεταβιβάζεται ο πίνακας δεικτών σε πίνακες **p**!

11.35

```
.....
int i;
for (i=0;i<100;i++)
{
   ptr[i](10,20);
}
.....
```

11.36

```
#include <iostream>
using namespace std;
int add(int x, int y);
void calc(int (*ptr)(int x, int y));

int main()
{
   int (*p)(int x, int y);
   p=add;
   calc(p);
   return 0;
}
int add(int x, int y)
{
   return x+y;
}
void calc(int (*ptr)(int x, int y))
{
   int a,b;
   cout << "Δώσε δύο αριθμούς :";
   cin >> a >> b;
   cout << ptr(a,b);
}
```

Δηλώνεται ένας δείκτης με όνομα **p**. Ο δείκτης θα δείχνει σε συναρτήσεις τύπου **int** με δύο παραμέτρους επίσης τύπου **int**.

Ο δείκτης **p** τώρα δείχνει στη συνάρτηση **add()**.

Στη συνάρτηση **calc()** μεταβιβάζεται ο δείκτης **p** που δείχνει στην **add()**!

Στη παράμετρο **ptr** μεταβιβάζεται ο δείκτης **p** που δείχνει στη συνάρτηση **add()**.

Μέσω του δείκτη **ptr**, καλείται η συνάρτηση **add()** με ορίσματα τα **a** και **b**. Η συνάρτηση **add()** επιστρέφει το άθροισμα των ορισμάτων της.

☞ Στη συνάρτηση calc() μεταβιβάζεται ένας δείκτης σε συνάρτηση τύπου int με δύο παραμέτρους, επίσης τύπου int. Η συνάρτηση calc() στη συνέχεια καλεί τη συνάρτηση στην οποία δείχνει ο δείκτης ptr (που της μεταβιβάστηκε), με ορίσματα τους αριθμούς που ζητάει από τον χρήστη.

☞ Επομένως για να εμφανίζει το άθροισμα των δύο αριθμών, η συνάρτηση η οποία καλείται μέσω του δείκτη ptr πρέπει να επιστρέφει ως τιμή το άθροισμα των ορισμάτων της.

☞ Για τον λόγο αυτό δημιουργείται η συνάρτηση add(), η οποία επιστρέφει το άθροισμα των ορισμάτων της. Στην calc() μεταβιβάζεται ένας δείκτης προς τη συνάρτηση add().

11.37

Το όνομα, το βάρος και το ύψος 100 αθλητών.

```
string onom[100];
float varos[100],ypsos[100];
```

Στο αγώνισμα του ακοντισμού, τις 6 βολές 100 αθλητών σε 20 αγώνες.

```
float voles[100][20][6];
```

Ένα σχολείο έχει έξι τάξεις, κάθε τάξη έχει 3 τμήματα και κάθε τμήμα 25 μαθητές. Κάθε χρόνο σε κάθε τάξη διδάσκονται 10 μαθήματα. Θέλουμε να καταχωρίσουμε τις ετήσιες απουσίες κάθε μαθητή του σχολείου σε κάθε μάθημα, καθώς και τα ονόματα όλων των μαθητών του σχολείου.

```
int apousies[6][3][25][10];
string onom[6][3][25];
```

Ασκήσεις Κεφαλαίου 12

12.1

```
#include <iostream>
#include <string>
using namespace std;
struct cars
{
   string ar_kykl;
   string xroma;
   string marka;
   int kybika;
   int ipodynami;
};
int main()
{
   cars mycar;
   cout << "Αριθμός κυκλοφορίας:";
   cin >> mycar.ar_kykl;
   cout << "Χρώμα:";
   cin >>mycar.xroma;
   cout << "Μάρκα:";
   cin >>mycar.marka;
   cout << "Κυβικά:";
   cin >> mycar.kybika;
   cout << "Ιπποδύναμη:";
   cin >> mycar.ipodynami;
   return 0;
}
```

Ορισμός της δομής **cars** με πέντε μέλη. Παρατηρούμε ότι σε αυτό το σημείο δεν δηλώνεται καμία μεταβλητή δομής.

Δήλωση της μεταβλητής **mycar** τύπου δομής **cars**.

Καταχώριση στοιχείων στα μέλη της μεταβλητής **mycar**.

12.2

```
#include <iostream>
#include <string>
using namespace std;
struct book
{
   string titlos;
   string syggrafeas;
   int etos;
   int ar_sel;
   float timi;
};
int main()
{
   book my_books[100];
   int i;
   for (i=0;i<100;i++)
   {
      cout << "Τίτλος:";
      getline(cin,my_books[i].titlos);
      if (my_books[i].titlos=="") break;
      cout << "Συγγραφέας:";
      getline(cin,my_books[i].syggrafeas);
      cout << "Έτος έκδοσης:";
      cin >> my_books[i].etos;
      cout << "Αριθμός σελίδων:";
      cin >> my_books[i].ar_sel;
      cout << "Τιμή:";
      cin >> my_books[i].timi;
```

Ορισμός της δομής **book** με πέντε μέλη. Σε αυτό το σημείο δεν δηλώνεται καμία μεταβλητή δομής.

Δήλωση ενός πίνακα **my_books** 100 θέσεων τύπου δομής **book**.

Καταχώριση στοιχείων στον πίνακα δομών

Αν καταχωριστεί κενός τίτλος, η επαναληπτική διαδικασία σταματάει.

```
    }
    cout << "Καταχώρισα "<<i<<" βιβλία"<<endl;
    return 0;
}
```

12.3

```
void print_it(stoixeia pin[])
{
   int i;
   for(i=0;i<100;i++)
   {
      cout  << setw(20) << pin[i].eponymo
            << setw(7) << pin[i].taxi
            << setw(6) << setprecision(1) << fixed
            << pin[i].mesos_oros << emdl;
   }
}
```

Στη συνάρτηση, δηλώνουμε ως παράμετρο έναν πίνακα ίδιου τύπου με τον πίνακα **mathites**.

Προβλέπονται 20 θέσεις στην οθόνη για το επώνυμο, 7 για την κλάση και 6 για τον μέσο όρο ο οποίος εμφανίζεται με ένα δεκαδικό ψηφίο.

12.4

```
int max_ilikia(stoixeia pin[])
{
   int i,max;
   max=pin[0].ilikia;
   for(i=0;i<100;i++)
   {
      if(pin[i].ilikia>max)
         max=pin[i].ilikia;
   }
   return max;
}
```

Στη συνάρτηση, δηλώνουμε ως παράμετρο έναν πίνακα ίδιου τύπου με τον πίνακα **mathites**.

Καταχωρίζουμε ως αρχική τιμής της **max** την ηλικία του πρώτου μαθητή (θέση 0 του πίνακα).

Αν η ηλικία του μαθητή **i** είναι μεγαλύτερη από την τρέχουσα μέγιστη ηλικία (**max**), τότε στη θέση του **max** καταχωρίζουμε αυτή την ηλικία.

12.5

```
#include <iostream>
using namespace std;
enum days {mon,tue,wed,thu,fri,sat,sun} birth_day,today;
union ar_kykl
{
   string aritmos_kykloforias;
   char grammata[3];
};
typedef int meres;
int main()
{
   meres a,b;
   days my_date;
   a=1;
   my_date=mon;
   wed=5;
   meres=23;
   today=3;
   birth_day=fri;
   return 0;
}
```

Μια ένωση (**union**) δεν μπορεί να έχει μέλη κλάσης **string**.

Το **web** δεν είναι μεταβλητή

Το **meres** δεν είναι μεταβλητή αλλά τύπος απαρίθμησης.

12. 6

- ☑ Μια δομή δεν μπορεί να περιέχει δύο μέλη με το ίδιο όνομα.
- ❑ Σε μια μεταβλητή τύπου enum δεν μπορούμε να καταχωρίσουμε άλλη τιμή πέρα από τις τιμές που ορίσαμε στη δήλωση του τύπου enum στον οποίο ανήκει.
- ☑ Για να έχουμε πρόσβαση στο μέλος μιας δομής μέσω ενός δείκτη, χρησιμοποιούμε τον τελεστή βέλους –>.
- ☑ Όταν μεταβιβάζουμε μια δομή σε μια συνάρτηση, η τυπική παράμετρος *πρέπει* να είναι του ίδιου τύπου δομής με το όρισμα που θα της μεταβιβάσουμε.
- ☑ Όταν έχουμε δύο μεταβλητές του ίδιου τύπου δομής, με τον τελεστή ανάθεσης τιμής = μπορούμε να καταχωρίσουμε όλα τα περιεχόμενα της μιας στην άλλη. Για παράδειγμα, η filos=pelatis καταχωρίζει όλα τα μέλη της μεταβλητής δομής pelatis στα αντίστοιχα μέλη της μεταβλητής δομής filos.
- ❑ Μια ένωση μπορεί να περιέχει μέλη τύπου string.
- ☑ Μια δομή μπορεί να περιέχει ως μέλη συναρτήσεις.
- ❑ Οι συναρτήσεις-μέλη μιας δομής δεν έχουν πρόσβαση στις μεταβλητές-μέλη της δομής. Θα πρέπει να τα μεταβιβάσουμε μέσω παραμέτρων.
- ☑ Τα μέλη μιας δομής μπορεί να είναι πίνακες αλλά και μεταβλητές άλλων δομών.
- ❑ Η typedef δημιουργεί ένα νέο τύπο δεδομένων.
- ☑ Οι κλάσεις είναι μια εξέλιξη των δομών και διαθέτουν όλα τα χαρακτηριστικά τους. Η χρήση κλάσεων κάνει περιττή τη χρήση δομών.

12.7

```
#include <iostream>
#include <string>
using namespace std;
#define AR 50
struct metoxi
{
   string onoma;
   int posotita;
   float timi;
   char eidos;
};
int main()
{
   metoxi kiniseis[AR];
   float syn_axia;
   int i,syn_a,syn_p;
   for (i=0;i<AR;i++)
   {
      cout << "Στοιχεία κίνησης No: "<<i+1<<endl;
      cout << "======================="<<endl;
      cout << "Είδος κίνησης (A/P):";
      cin >> kiniseis[i].eidos;
      cout << "Ονομασία μετοχής:";
      cin >> kiniseis[i].onoma;
      cout << "Ποσότητα:";
      cin >> kiniseis[i].posotita;
      cout << "Τιμή:";
      cin >> kiniseis[i].timi;
   }
```

Ορισμός της δομής **metoxi** με τα κατάλληλα πεδία.

Ορισμός ενός πίνακα **kiniseis**, δομών **metoxi**, με AR (50) θέσεις.

Στη μεταβλητή **syn_axia** θα υπολογιστεί η συνολική αξία των κινήσεων. Στις μεταβλητές **syn_a** και **syn_b** θα καταγραφεί το πλήθος των μετοχών που αγοράστηκαν και πουλήθηκαν αντίστοιχα.

Διαβάζονται τα στοιχεία κάθε κίνησης και καταχωρίζονται στις θέσεις του πίνακα **kiniseis** (στα αντίστοιχα πεδία).

```
   syn_axia=0;
   syn_a=0;
   syn_p=0;
   for (i=0;i<AR;i++)
   {
      if (kiniseis[i].eidos=='A')
         syn_a=syn_a+kiniseis[i].posotita;
      if (kiniseis[i].eidos=='Π')
         syn_p=syn_p+kiniseis[i].posotita;
      syn_axia=syn_axia+kiniseis[i].timi*kiniseis[i].posotita;
   }
   cout << "Πλήθος μετοχών που αγοράστηκαν = "<<syn_a<<endl;
   cout << "Πλήθος μετοχών που πουλήθηκαν = "<<syn_p<<endl;
   cout << "Συνολική αξία διακίνησης = "<<syn_axia<<endl;
   return 0;
}
```

Δίνονται αρχικές τιμές στις μεταβλητές.

Υπολογισμός του συνόλου των μετοχών που αγοράστηκαν και πουλήθηκαν, καθώς και της συνολικής αξίας των κινήσεων.

12.8

```
#include <iostream>
using namespace std;

#define AR 10

struct ptisi
{
   int ar_ptisis;
   int theseis;
   int epibates;
};
int main()
{
   ptisi ptiseis[AR];
   int i,syn_epibaton,plirotita;
   for (i=0;i<AR;i++)
   {
      cout << "Στοιχεια πτήσης No: "<<i+1<<endl;
      cout << "======================="<<endl;
      cout << "Αριθμός πτήσης:";
      cin >> ptiseis[i].ar_ptisis;
      cout << "Πλήθος θέσεων:";
      cin >> ptiseis[i].theseis;
      cout << "Πλήθος επιβατών:";
      cin >> ptiseis[i].epibates;
   }
   syn_epibaton=0;
   plirotita=0;
   for (i=0;i<AR;i++)
   {
      syn_epibaton=syn_epibaton+ptiseis[i].epibates;
      if (ptiseis[i].epibates>=ptiseis[i].theseis*80/100)
         plirotita++;
   }
   cout << "Συνολικό πλήθος επιβατών = "<<syn_epibaton<<endl;
   cout << "Πλήθος πτήσεων με πληρότητα = "<<plirotita<<endl;
   return 0;
}
```

Ορισμός της δομής **ptisi** με τα κατάλληλα πεδία.

Δήλωση ενός πίνακα **ptiseis**, δομών **ptisi**, με **AR** (10) θέσεις.

Στη μεταβλητή **syn_epibaton** θα υπολογιστεί το σύνολο των επιβατών όλων των πτήσεων. Στη μεταβλητή **plirotita** θα καταγραφεί το πλήθος των πτήσεων με πληρότητα.

Διαβάζονται τα στοιχεία κάθε πτήσης και καταχωρίζονται στις θέσεις του πίνακα **ptiseis** (στα αντίστοιχα μέλη).

Δίνονται αρχικές τιμές στις μεταβλητές.

Υπολογισμός του συνόλου των επιβατών και του πλήθους των πτήσεων με πληρότητα.

12.9

```
#include <iostream>
#include <string>
using namespace std;
#define AR 15

struct math
{
   string eponymo
   string onoma;
   float bathmos;
   int ilikia;
};

struct tmima
{
   string titlos;
   int ar_mat;
   struct math m[25];
} tmimata[AR];

int main()
{
   int i,j;
   for (i=0;i<AR;i++)
   {
      cout << "Στοιχεία τμήματος No:"<<i+1<<endl;
      cout << "======================"<<endl;
      cout << "Τίτλος τμήματος:";
      cin >> tmimata[i].titlos;
      cout << "Πλήθος μαθητών:";
      cin >> tmimata[i].ar_mat;
      cout << "Οι "<< tmimata[i].ar_mat << " μαθητές του τμήματος"<<endl;
      cout << "==================================="<<endl;
      for (j=0;j<tmimata[i].ar_mat;j++)
      {
         cout << "Στοιχεία μαθητή No: "<<j+1<<endl;
         cout << "======================"<<endl;
         cout << "Επώνυμο:";
         cin >> tmimata[i].m[j].eponymo;
         cout << "Ονομα:";
         cin >> tmimata[i].m[j].onoma;
         cout << "Βαθμός:";
         cin >> tmimata[i].m[j].bathmos;
         cout << "Ηλικία:";
         cin >> tmimata[i].m[j].ilikia;
         cout << endl;
      }
      cout << endl<<endl;
   }
   return 0;
}
```

Στον πίνακα **tmimata** θα καταχωρίζονται τα στοιχεία των 15 τμημάτων του σχολείου.

Διαβάζονται τα στοιχεία κάθε τμήματος και καταχωρίζονται στις θέσεις του πίνακα **tmimata** (στα αντίστοιχα μέλη).

Διαβάζονται τα στοιχεία κάθε μαθητή του τμήματος και καταχωρίζονται στις θέσεις του πίνακα **mathites** (στα αντίστοιχα μέλη).

12.10

```
void display(struct tmima tm[])
{
   int synolo_mathiton,i,j;
   float max_mo,mo[AR],sum;
   for (i=0;i<AR;i++)
   {
      synolo_mathiton=synolo_mathiton+tmimata[i].ar_mat;
   }
   max_mo=0;
   for (i=0;i<AR;i++)
   {
      sum=0;
      for (j=0;j<tmimata[i].ar_mat;j++)
      {
         sum=sum+tmimata[i].m[j].bathmos;
      }
      mo[i]=sum/tmimata[i].ar_mat;
      cout << "Ο μέσος όρος του τμήματος "<< tmimata[i].titlos
           <<" είναι"<<mo[i]<<endl;
      // Εύρεση του μέγιστου μέσου όρου
      if (mo[i]>max_mo) max_mo=mo[i];
   }
   // Εμφάνιση των τμημάτων με μέσο όρο ίσο με το μέγιστο μέσο όρο.
   cout << "Τα τμήματα με τον μεγαλύτερο μέσο όρο "<<max_mo<<endl;
   for (i=0;i<AR;i++)
   {
      if (mo[i]==max_mo) cout << tmimata[i].titlos;
   }
   // Εμφάνιση των μαθητών με βαθμό πάνω από 18.5
   cout << "Μαθητές με βαθμό μεγαλύτερο από 18.5"<<endl;
   cout << "================================"<<endl;
   for (i=0;i<AR;i++)
      for (j=0;j<tmimata[i].ar_mat;j++)
      {
         if (tmimata[i].m[j].bathmos>=18.5)
            cout << tmimata[i].m[j].eponymo;
      }
}
```

Στον πίνακα **mo** θα καταχωρίζονται οι μέσοι όροι των βαθμολογιών των τμημάτων.

Στη μεταβλητή **synolo_mathiton** υπολογίζεται το σύνολο των μαθητών από όλα τα τμήματα.

Στη μεταβλητή **sum** υπολογίζεται το σύνολο των βαθμών των μαθητών ενός τμήματος.

Στον πίνακα **mo[]** καταχωρίζονται οι μέσοι όροι των βαθμολογιών κάθε τμήματος.

12.11

```
#include <iostream>
#include <cstring>
using namespace std;
union ar_kykl
{
   char all[8];
   struct
   {
      char polis[3];
      char pavla;
      char num[4];
   } details;
};
int main()
{
   union ar_kykl mycar;
   strcpy(mycar.all,"MYA-3465");
   cout<<mycar.details.num<<endl;
   return 0;
}
```

3465

Η συμβολοσειρά που αποθηκεύεται στο μέλος **all** διαμοιράζεται στα μέλη **polis**, **pavla**, και **num** του πεδίου **details**.

12.12

```
.....
int main()
{
   string onomata[100];
   float bathmoi[100][10];
   int ilikies[100];
   .....
   return 0;
}
#include <iostream>
#include <string>
using namespace std;
struct stoixeia
{
   string onoma;
   int ilikia;
   float vath[10];
};
int main()
{
   string onomata[100];
   float bathmoi[100][10];
   int ilikies[100];
   .....
   int i,j;
   stoixeia foitites[100];
   for (i=0;i<100;i++)
   {
      foitites[i].onoma=onomata[i];
      foitites[i].ilikia=ilikies[i];
      for (j=0;j<100;j++)
         foitites[i].vath[j]=bathmoi[i][j];
   }
   return 0;
}
```

Η δομή **stoixeia** περιλαμβάνει μέλη για την αποθήκευση των στοιχείων ενός φοιτητή.

Στον πίνακα δομών **foitites** θα αντιγραφούν τα περιεχόμενα των μεμονωμένων πινάκων

Τα περιεχόμενα των μεμονωμένων πινάκων αντιγράφονται στα αντίστοιχα μέλη του πίνακα δομών **foitites**.

12.13

```
struct stoixeia
{
   char eponymo[30];
   float b1,b2,b3;
   float mo()
   {
      return (b1+b2_b3)/3;
   }
   void bonus(float ar)
   {
      b1=b1+ar;
      b2=b2+ar;
      b3=b3+ar;
   }
};
```

Η συνάρτηση-μέλος **mo()** επιστρέφει ως τιμή τον μέσο όρο των βαθμών

Η συνάρτηση-μέλος **bonus()** προσθέτει σε κάθε βαθμό την τιμή της παραμέτρου της **ar**.

12.14

```
#include <iostream>
#include <string>
#include <cmath>

using namespace std;

struct circle
{
   string xroma;
   float aktina;
   float emvado()
   {
      return 3.14*pow(aktina,2);
   }
   float perimetros()
   {
      return 2*3.14*aktina;
   }
   void scale(float kl)
   {
      aktina=aktina*kl;
   }
   void stoixeia()
   {
      cout<<"Κύκλος χρώματος "<<xroma<<" και ακτίνας "<<aktina<<endl;
   }
};

int main()
{
   circle c1,c2;
   c1.xroma="Κόκκινο";
   c1.aktina=10;
   c2.xroma="Πράσινο";
   c2.aktina=20;
   c1.stoixeia();
   c2.stoixeia();
   cout << "Εμβαδό κύκλου "<<c1.xroma<<" = "<<c1.emvado()<<endl;
   cout << "Περίμετρος κύκλου "<<c1.xroma<<" = "<<c1.perimetros()<<endl;
   cout << "Εμβαδό κύκλου "<<c2.xroma<<" = "<<c2.emvado()<<endl;
   cout << "Περίμετρος κύκλου "<<c2.xroma<<" = "<<c2.perimetros()<<endl;
   c1.scale(2); // Διπλασιασμός του κύκλου c1
   cout << "Εμβαδό κύκλου "<<c1.xroma<<" = "<<c1.emvado()<<endl;
   return 0;
}
```

Δημιουργία των δύο μεταβλητών δομής **circle**, **c1** και **c2**.

Καταχώριση των στοιχείων των κύκλων στα μέλη των μεταβλητών δομής.

Εμφάνιση των στοιχείων των δύο κύκλων

12.15

```
.....
int main()
{
   rectangle rec1,rec2,*ptr;
   ptr=&rec1;
   ptr->plevra_a=5;
   ptr->plevra_b=20;
   cout << ptr->emvado()<<endl;
   cout << rec1.perimetros()<<endl;
   rec2.plevra_a=10;
   rec2.plevra_b=30;
   ptr=&rec2;
   cout << ptr->emvado()<<endl;
   cout << ptr->perimetros()<<endl;
   return 0;
}
```

```
100
50
300
80
```

Ο δείκτης **ptr** δείχνει στη μεταβλητή δομής **rec1**. Μέσω αυτού του δείκτη καταχωρίζονται οι τιμές 5 και 20 στιις μεταβλητές-μέλη της.

Μέσω του δείκτη **ptr** εμφανίζεται το εμβαδόν της **rec1**.

Καταχωρίζονται οι τιμές 10 και 30 στιις μεταβλητές-μέλη της μεταβλητής δομής **rec2**.

Τώρα ο δείκτης **ptr** δείχνει στη μεταβλητή δομής **rec2**. Μέσω του δείκτη **ptr** εμφανίζεται το εμβαδόν και η περίμετρος της **rec1**.

12.16

```
#include <iostream>
#include <string>
using namespace std;
.....
int main()
{
   stoixeia filos,*ptr;
   int i;
   ptr=&filos;
   ptr->onoma="Μπαμπόγερος Νικόλαος";
   ptr->ilikia=86;
   for (i=0;i<5;i++) ptr->vath[i]=rand()%11;
   ptr->display();
   return 0;
}
```

Ασκήσεις Κεφαλαίου 13

13.1

```
#include <iostream>
#include <string>
using namespace std;
class stoixeia
{
   int ilikia;
public:
   string onoma;
   float varos;
   void set(int a)
   {
      ilikia=a;
   }
};
int main()
{
   stoixeia ego;
   ego.onoma="Νίκος";
   ego.varos=82;
   ego.set(52);
   return 0;
}
```

Δημιουργία αντικειμένου της κλάσης **stoixeia**.

Η καταχώριση τιμής στην ιδιωτική μεταβλητή-μέλος **ilikia** γινεται μόνο μέσω της δημόσιας μεθόδου **set()**.

13.2

☞ Η κλάση δεν διαθέτει δημόσια μέλη, επομένως δεν μπορεί να προσπελαστεί κανένα μέλος της. Τα αντικείμενα που δημιουργεί είναι κυριολεκτικά άχρηστα.

13.3

```
class test
{
   int ar;
public:
   void show() {cout << ar << endl;}
   test();
   ~test();
};
test::test()
{
   ar=rand();
   cout<<"Δημιουργήθηκε το "<<ar<<endl;
}
test::~test()
{
   cout<<"Καταστράφηκε το "<<ar<<endl;
}
```

Δηλώσεις μεθόδων δόμησης και αποδόμησης.

Μέθοδος δόμησης.

Μέθοδος αποδόμησης.

☞ Στην κλάση πρέπει να προστεθούν μέθοδοι δόμησης και αποδόμησης.

13.4

- ☑ Τα ιδιωτικά μέλη μιας κλάσης είναι προσπελάσιμα μόνο από τις μεθόδους της κλάσης.
- ❑ Με τη δήλωση μιας κλάσης, δεσμεύεται χώρος μνήμης για τις μεταβλητές-μέλη της κλάσης.
- ☑ Όταν έχουμε δύο αντικείμενα της ίδιας κλάσης, μπορούμε να καταχωρίσουμε όλα τα στοιχεία του ενός στο άλλο με τον τελεστή =.
- ❑ Αν μια κλάση δεν έχει μέθοδο δόμησης, δεν μπορούν να δημιουργηθούν αντικείμενα σε αυτή την κλάση.
- ☑ Μια κλάση μπορεί να έχει πολλές μεθόδους δόμησης.
- ❑ Κάθε αντικείμενο της κλάσης έχει το δικό του αντίγραφο κώδικα για κάθε μέθοδο της κλάσης.

13.5

```
#include <iostream>
using namespace std;
class circle
{
   float aktina;
public
   circle(float r);
   float emvado();
}
float circle::emvado()
{
   return pi*aktina*aktina;
}
void circle::set_r(float r)
{
   aktina=r;
}
circle::circle(float r)
{
   aktina=r;
}
int main()
{
   circle cir1,cir2(20);
   cir1.aktina=12;
   return 0;
}
```

Το προσδιοριστικό **public** πρέπει να ακολουθείται από άνω-κάτω τελεία :.

Ο ορισμός μιας κλάσης πρέπει να τερματίζεται με ερωτηματικό ;.

Το **pi** δεν έχει οριστεί.

Η μέθοδος **set_r()**, δεν έχει δηλωθεί μέσα στο σώμα της κλάσης.

Δεν υπάρχει μέθοδος δόμησης για τη δημιουργία αντικειμένων χωρίς όρισμα.

Η μεταβλητή-μέλος **aktina** είναι ιδιωτική και μη προσπελάσιμη από κώδικα εκτός της κλάσης.

13.6

```
float kafetiera::kostos()
{
   float euro;
   euro =(1000-nero)*0.01+(100-kafes)*0.05+(300-gala)*0.2+(200-zaxari)*0.1;
   return euro/kafedes;
}
```

Υπολογίζεται το κόστος ανά καφέ.

13.7

```
#include <iostream>
using namespace std;
class triangle
{
   float basi;
   float ypsos;
public:
   triangle(){cout<<"Ναι-Κενό"<<endl;}
   triangle(float b, float y);
   ~triangle(){cout<<"Όχι"<<endl;}
   float emvado(){return basi*ypsos/2;}
   void set(float b, float y){basi=b;ypsos=y;}
   void show(){cout<<"Βάση= "<<basi<<" Ύψος="<<ypsos<<endl;}
};
triangle::triangle(float b, float y)
{
   cout<<"Ναι-Γεμάτο"<<endl;
   basi=b;
   ypsos=y;
}
int main()
{
   triangle tr1(3,5),tr2;
   tr2.set(7,8);
   tr2.show();
   {
      triangle tr3(9,8);
      tr3.show();
   }
   cout<<"Εμβαδό="<<tr1.emvado()<<endl;
   return 0;
}
```

```
Ναι-Γεμάτο
Ναι-Κενό
Βάση= 7 Ύψος=8
Ναι-Γεμάτο
Βάση= 9 Ύψος=8
Όχι
Εμβαδό=7.5
Όχι
Όχι
```

Για τη δημιουργία του αντικειμένου **tr1** καλείται η έκδοση της μεθόδου δόμησης με δύο παραμέτρους. Για τη δημιουργία του αντικειμένου **tr2** καλείται η έκδοση της μεθόδου δόμησης χωρίς παραμέτρους.

Δημιουργία του αντικειμένου **tr3**.

Καταστροφή του αντικειμένου **tr3**, καλείται η μέθοδος αποδόμησης.

Καταστροφή των αντικειμένων **tr1** & **tr2**.

13.8

Από μια κλάση δημιουργείται ένα *αντικείμενο*. Κατά τη δημιουργία του καλείται η μέθοδος *δόμησης* ενώ κατά την καταστροφή του η μέθοδος *αποδόμησης*. Οι μέθοδοι μιας κλάσης μπορεί να εφαρμοστούν μόνο επάνω σε ένα *αντικείμενο* της κλάσης. Μια κλάση μπορεί να διαθέτει πολλές *υπερφορτωμένες* μεθόδους δόμησης. Η μέθοδος δόμησης μιας κλάσης έχει ίδιο όνομα με *την κλάση*.

13.9

Τα *ιδιωτικά* μέλη μιας κλάσης δεν είναι προσπελάσιμα από κώδικα εκτός της κλάσης αλλά είναι από τις *μεθόδους* της κλάσης. Οι υπερφορτωμένες εκδόσεις μιας μεθόδου μιας κλάσης έχουν όλες το ίδιο *όνομα* αλλά διαφορετικό *πλήθος* ή τύπο *παραμέτρων*.

13.10

- Πόσες μεθόδους δόμησης διαθέτει η κλάση atm; *2*
- Αναφέρετε ένα ιδιωτικό μέλος της κλάσης atm: *poso*
- Γράψτε μία πρόταση με την οποία δημιουργούνται δύο αντικείμενα της παραπάνω κλάσης με ονόματα at1 και at2: *atm at1 at2;*
- Υποθέτουμε ότι υπάρχει ένα αντικείμενο της παραπάνω κλάσης με όνομα at1. Γράψτε μία πρόταση με την οποία θα καταχωριστεί στη μεταβλητή-μέλος trapeza του αντικειμένου at1 το κείμενο «ΠΙΣΤΕΩΣ»: *at1.trapeza="ΠΙΣΤΕΩΣ"*.
- Υποθέτουμε ότι υπάρχει ένα αντικείμενο της παραπάνω κλάσης με όνομα at1. Τι θα κάνει η πρόταση at1.poso=1000; Αιτιολογήστε την απάντησή σας. *Είναι λάθος διότι η μεταβλητή-*

*μέλος **poso** είναι ιδιωτική και επομένως δεν είναι προσπελάσιμη από κώδικα εκτός της κλάσης.*

- Υποθέτουμε ότι υπάρχει ένα αντικείμενο ATM της παραπάνω κλάσης με όνομα at1. Γράψτε μία πρόταση με την οποία θα εμφανίζεται στην οθόνη το υπόλοιπο χρημάτων που υπάρχει στο at1: *cout << at1.ypoloipo() << endl;*
- Σχολιάστε τι θα κάνει η πρόταση: atm at3("ΓΕΝΙΚΗ","ΑΘΗΝΑ"); *Θα δημιουργήσει ένα αντικείμενο της κλάσης **atm** με όνομα **at3** καταχωρίζοντας ταυτόχρονα στη μεταβλητή-μέλος **trapeza** το κείμενο «ΓΕΝΙΚΗ» και στη μεταβλητή-μέλος **address** το κείμενο «ΑΘΗΝΑ».*

13.11

```
int main()
{
   atm t1("HSBC","ATHENS"),t2;
   t2.trapeza="ALPHA";
   cout << t1.address << endl;
   cout << t2.ypoloipo() << endl;
   t1.katathesi(20000);
   t1.analipsi(5000);
   cout << t1.ypoloipo() << endl;
   t1=t2;
   cout << t1.trapeza << endl;
   return 0;
}
```

```
ATHENS
10000
65000
ALPHA
```

13.12

```
#include <iostream>
#include <string>
using namespace std;

class biblio
{
   float timi;
public:
   int selides;
   string titlos;
   biblio();
   biblio(string title);
   biblio(string title,int t);
   void info();
   void set_timi(float t);
};
void biblio::info()
{
   cout <<"----------------------"<<endl;
   cout << titlos << endl;
   cout << "Σελίδες :"<< selides << endl;
   cout << "Τιμή :"<< timi << endl;
}
biblio::biblio()
{
   titlos="";
   timi = 0;
   selides=0;
}
biblio::biblio(string title)
{
   titlos=title;
   timi = 0;
```

Ιδιωτικό μέλος της κλάσης **biblio**.

Δημόσια μέλη της κλάσης **biblio**.

Η κλάση **biblio** διαθέτει τη μέθοδο δόμησης **biblio()** υπερφορτωμένη τρεις φορές με διαφορετικό πλήθος παραμέτρων.

Η μέθοδος **info()** εμφανίζει στην οθόνη τα στοιχεία του αντικειμένου-βιβλίου.

Αυτή η έκδοση της μεθόδου δόμησης καλείται όταν δημιουργείται ένα αντικείμενο χωρίς ορίσματα. Καταχωρίζει αρχικές τιμές στις μεταβλητές-μέλη του αντικειμένου, κενό για τον τίτλο και 0 για τις δύο υπόλοιπες.

Αυτή η έκδοση της μεθόδου δόμησης καλείται όταν δημιουργείται ένα αντικείμενο με ένα όρισμα. Καταχωρίζει την τιμή της παραμέτρου στο μέλος **titlos** του αντικειμένου.

```
   selides=0;
}

biblio::biblio(string title,int t)
{
   titlos=title;
   timi=t;
   selides=0;
}
void biblio::set_timi(float t)
{
   timi=t;
}

int main()
{
   biblio b1,b2("C++ σε βάθος"),b3("Αλγόριθμοι",40);
   b1.titlos="Αντικειμενοστρεφής προγραμματισμός";
   b1.selides=100;
   b2.selides=200;
   b3.selides=300;
   b1.set_timi(20);
   b2.set_timi(30);
   b1.info();
   b2.info();
   b3.info();
   return 0;
}
```

Αυτή η έκδοση της μεθόδου δόμησης καλείται όταν δημιουργείται ένα αντικείμενο με δύο ορίσματα. Καταχωρίζει την πρώτη παράμετρο στο μέλος **titlos** και τη δεύτερη στο μέλος **timi** του αντικειμένου.

Στη μεταβλητή-μέλος **selides** των τριών αντικειμένων καταχωρίζονται οι τιμές 100, 200 και 300 αντίστοιχα.

Καταχωρίζονται οι τιμές 20 και 30 για τα βιβλία b1 και b2, αντίστοιχα.

Εμφανίζονται τα στοιχεία και των τριών βιβλίων.

☞ Στη πρώτη πρόταση της συνάρτησης main() δημιουργούνται τρία αντικείμενα της κλάσης biblio καλώντας τις αντίστοιχες μεθόδους δόμησης: για καμία, μία και δύο παραμέτρους.

☞ ΠΡΟΣΟΧΗ: η μεταβλητή μέλος timi, ως ιδιωτική, δεν είναι προσπελάσιμη παρά μόνο μέσω της δημόσιας μεθόδου set_timi().

13.13

Όταν θέλουμε να φτιάξουμε μια αντικειμενοστρεφή εφαρμογή, το πρώτο πράγμα είναι να προσδιορίσουμε τα αντικείμενα με τα χαρακτηριστικά και τις λειτουργίες που θέλουμε να χρησιμοποιήσουμε. Το επόμενο βήμα είναι να ορίσουμε τις κλάσεις από τις οποίες θα προέλθουν τα αντικείμενα αυτά.

Στο συγκεκριμένο πρόγραμμα, το μόνο αντικείμενο που προσδιορίζω είναι το έτος. Μπορεί να μην είναι ένα «χειροπιαστό» αντικείμενο του πραγματικού κόσμου, αλλά στον ΑΠ οποιαδήποτε οντότητα μπορεί να είναι αντικείμενο. Το μοναδικό χαρακτηριστικό ενός έτους είναι η τιμή του. Ο έλεγχος αν το έτος είναι δίσεκτο ή όχι θα το αναλάβει μια μέθοδος η οποία θα εφαρμόζεται στο αντικείμενο-έτος. Επομένως το πρόβλημα ανάγεται στη δημιουργία μιας κλάσης από την οποία θα δημιουργούνται τα αντικείμενα έτη. Στο πρόγραμμα που ακολουθεί, η κλάση αυτή ονομάζεται (τι άλλο!) etos!

```
#include <iostream>
using namespace std;

class etos
{
 public:
   int et;
   bool disekto();
};
bool etos::disekto()
{
```

Μεταβλητή-μέλος για την καταχώριση του έτους

Η μέθοδος **disekto()** επιστρέφει τιμή **true** αν το έτος είναι δίσεκτο και **false** αν δεν είναι.

```
   if((et%4 == 0 && et%100 != 0) || et%400 == 0)
      return true;
   else
      return false;
}
int main()
{
   etos e1;
   do
   {
      cout << "Δώσε έτος:";
      cin >> e1.et;
      if (e1.disekto())
         cout << "Δίσεκτο" << endl;
      else
         cout << "Κανονικό" << endl;
   } while (e1.et>0);
   return 0;
}
```

Δημιουργία ενός αντικειμένου της κλάσης **etos** με όνομα **e1**.

Η τιμή του έτους που δινει ο χρήστης καταχωρίζεται στη μεταβλητή-μέλος **et** του αντικειμένου **e1**.

Εφαρμόζοντας τη μέθοδο **disekto()** ελέγχουμε αν το αντικείμενο-έτος **e1** είναι δίσεκτο!

13.14

```
#include <iostream>
#include <string>
using namespace std;
class box
{
   float ypsos;
   float mikos;
   float platos;
public:
   string xroma;
   string periexomeno;
   box();
   box(float all);
   box(float y,float m,float p);
   void display();
   float ogos();
   void set(float y,float m,float p);
   void set(float kl);
};
box::box()
{
   ypsos=0;
   mikos=0;
   platos=0;
   xroma="ΑΣΠΡΟ";
   periexomeno="-";
}

box::box(float all)
{
   ypsos=mikos=platos=all;
}

box::box(float y,float m,float p)
{
   ypsos=y;
   mikos=m;
   platos=p;
}
```

Ιδιωτικά μέλη της κλάσης

Δημόσια μέλη της κλάσης

Μέθοδοι δόμησης της κλάσης

Υπόλοιπες μέθοδοι της κλάσης

Μέθοδος δόμησης η οποία εφαρμόζεται όταν δημιουργούνται αντικείμενα χωρίς ορίσματα

Μέθοδος δόμησης η οποία εφαρμόζεται όταν δημιουργούνται αντικείμενα με ένα όρισμα

Μέθοδος δόμησης η οποία εφαρμόζεται όταν δημιουργούνται αντικείμενα με τρία ορίσματα

```
void box::display()
{
   cout <<"Κουτί "<<xroma<<" Διαστάσεων: "<<ypsos<<"x"<<mikos<<"x"<<platos
        <<" με "<<periexomeno<<endl;
}
float box::ogos()
{
   return ypsos*mikos*platos;
}

void box::set(float y,float m,float p)
{
   ypsos=y;
   mikos=m;
   platos=p;
}

void box::set(float kl)
{
   ypsos=ypsos*kl;
   mikos=mikos*kl;
   platos=platos*kl;
}

int main()
{
   box k1,k2(5),k3(10,20,30);
   k2.xroma="Κόκκινο";
   k3.xroma="Πράσινο";
   k2.periexomeno="Παπούτσια";
   k3.periexomeno="Αυγά";
   cout << k3.ogos()<<endl;
   k2.set(2);
   k1.set(7,8,9);
   k1.display();
   k2.display();
   return 0;
}
```

Μέθοδος η οποία εμφανίζει τα στοιχεία των αντικειμένων

Μέθοδος η οποία επιστρέφει ως τιμή τον όγκο των αντικειμένων-κουτιών.

Μέθοδος η οποία θέτει ως διαστάσεις σε ένα αντικείμενο τις τιμές των παραμέτρων της.

Υπερφορτωμένη έκδοση της προηγούμενης μεθόδου η οποία θέτει και στις τρεις διαστάσεις ενός αντικειμένου την τιμή της παραμέτρου της.

Το αντικείμενο **k1** δημιουργείται με την πρώτη έκδοση της μεθόδου δόμησης, το **k2** με τη δεύτερη και το **k3** με την τρίτη.

Καταχωρίζεται χρώμα και περιεχόμενο για τα αντικείμενα **k2** και **k3**.

Εμφανίζεται ο όγκος του αντικειμένου **k3**.

Διπλασιάζονται οι διαστάσεις του **k2** με εφαρμογή της δεύτερης έκδοσης της **set()**.

Το **k1** λαμβάνει συγκεκριμένες διαστάσεις

Εμφανίζονται τα στοιχεία των **k1** και **k2**

13.15

13.1

stoixeia
-ilikia:int +onoma:string +varos:float
+set(int):void

13.6

kfetiera
-kafes:int -gala:int -zaxari:int -nero:int -kafedes:int
-ginetai(int,int,int):bool +kafetiera(): +status():void +sketos(int):void +glykos(int):void +metrios_me_gala(int):void +kostos():float

13.7

triangle
-basi:float -ypsos:float
+triangle(): +triangle(float,float): +~triangle(): +emvado():float +set(float,float):void +show():void

13.10

atm
-poso:int +trapeza:string +address:string
+atm(): +atm(string,string): +analipsi(int):void +katathesi(int):void +ypoloipo():int

13.16

```
car mycar;
mycar.xroma="Κόκκινο";
mycar.marka="Toyota";
mycar.fill(50);
mycar.start();
mycar.move(50);
mycar.beep(2);
mycar.turn_right();
mycar.move(100);
mycar.beep(1);
mycar.turn_right();
mycar.move(50);
mycar.stop();
cout<<mycar.used_fuel()<<endl;
```

13.17

```
#include <iostream>
#include <string>
using namespace std;

class circle
{
   float aktina;
   float center_x;
   float center_y;
public:
   string xroma;
   circle(float r, float x, float y, string xr);
   void info(int mode);
};
circle::circle(float r=10, float x=0, float y=0, string xr="ΜΑΥΡΟΣ")
{
   aktina=r;
   center_x=x;
   center_y=y;
   xroma=xr;
}
void circle::info(int mode=1)
{
   if (mode==1)
      cout<<"Ακτίνα="<<aktina<<" x="<<center_x<<" y="<<center_y<<endl;
   else if (mode==2)
      cout<<"Κύκλος "<<xroma<<" στο "<<center_x<<"-"<<center_y<<endl;
   else
      cout<<"R="<<aktina<<"@"<<center_x<<"/"<<center_y<<endl;
}

int main()
{
   circle c1,c2(20),c3(30,5,5),c4(40,15,25,"ΑΣΠΡΟΣ");
   c1.info();
   c2.info(2);
   c3.info(3);
   c4.info(2);
   return 0;
}
```

Η μέθοδος δόμησης της κλάσης διαθέτει τέσσερις παραμέτρους με προκαθορισμένες τιμές. Αυτό μας δίνει τη δυνατότητα να δημιουργήσουμε αντικείμενα της κλάσης με κανένα, ένα, δύο, τρία ή και τέσσερα ορίσματα!

Η μέθοδος **info()** διαθέτει μία παράμετρο με προκαθορισμένη τιμή. Αυτό μας δίνει τη δυνατότητα να εφαρμόσουμε τη μέθοδο με κανένα ή με ένα όρισμα.

```
Ακτίνα=10 x=0 y=0
Κύκλος ΜΑΥΡΟΣ στο 0-0
R=30@5/5
Κύκλος ΑΣΠΡΟΣ στο 15-25
```

☞ Κατά τη δημιουργία του αντικειμένου c1 καλείται η μέθοδος δόμησης χωρίς ορίσματα, οπότε οι μεταβλητές-μέλη του λαμβάνουν τις προκαθορισμένες τιμές των παραμέτρων της.

- Κατά τη δημιουργία του αντικειμένου c2 καλείται η μέθοδος δόμησης με ένα όρισμα, οπότε η τιμή του ορίσματος αποθηκεύεται στη μεταβλητή-μέλος aktina, και οι υπόλοιπες μεταβλητές μέλη του λαμβάνουν τις προκαθορισμένες τιμές των αντίστοιχων παραμέτρων της.
- Κατά τη δημιουργία του αντικειμένου c3 καλείται η μέθοδος δόμησης με τρία ορίσματα, οπότε οι τιμές των ορισμάτων αποθηκεύονται στις μεταβλητές-μέλη aktina, center_x και center_y. Ως χρώμα αποθηκεύεται η προκαθορισμένη τιμή της αντίστοιχης παραμέτρου.
- Τέλος, κατά τη δημιουργία του αντικειμένου c4 καλείται η μέθοδος δόμησης με τέσσερα ορίσματα, οι τιμές των οποίων αποθηκεύονται στις μεταβλητές-μέλη του αντικειμένου. Δεν χρησιμοποιείται καμία από τις προκαθορισμένες τιμές.
- Η πρώτη εφαρμογή της μεθόδου info() χωρίς όρισμα έχει ως αποτέλεσμα να χρησιμοποιηθεί η προκαθορισμένη τιμή της παραμέτρου mode, το 1. Οι υπόλοιπες κλήσεις της μεθόδου, με ένα όρισμα, μεταβιβάζουν την τιμή του ορίσματος στην παράμετρο mode.

Γράψτε επίσης τις κατάλληλες προτάσεις ώστε να καταχωρίσετε στο υπάρχον αντικείμενο c1 τις τιμές 50, 30 και 45 για την ακτίνα και τις συντεταγμένες center_x και center_y, αντίστοιχα.

*Αυτό δεν είναι δυνατό να υλοποιηθεί επειδή τα μέλη αυτά είναι **ιδιωτικά** και **μη προσπελάσιμα**. Δεν υπάρχει καμία δημόσια μέθοδος της κλάσης που να δίνει αυτή τη δυνατότητα.*

Ασκήσεις Κεφαλαίου 14

14.1

- ☑ Όταν ένα αντικείμενο μεταβιβάζεται σε μια συνάρτηση με αναφορά, τότε η συνάρτηση μπορεί να τροποποιήσει τα δεδομένα του αντικειμένου το οποίο χρησιμοποιήθηκε ως όρισμα κατά την κλήση της.
- ❑ Για να έχουμε πρόσβαση στα μέλη ενός αντικειμένου μέσω ενός δείκτη σε αυτό, χρησιμοποιούμε τον τελεστή τελείας (.).
- ❑ Κάθε κλάση πρέπει να έχει υποχρεωτικά μια μέθοδο δόμησης αντιγράφου.
- ❑ Η μέθοδος δόμησης αντιγράφου χρησιμοποιείται στην έμμεση δημιουργία αντικειμένων.
- ☑ Μια στατική μεταβλητή-μέλος καταλαμβάνει χώρο στη μνήμη μόνο μία φορά για ολόκληρη την κλάση.
- ❑ Μόνο οι στατικές συναρτήσεις-μέλη έχουν πρόσβαση στις στατικές μεταβλητές-μέλη της κλάσης.
- ❑ Οι φίλιες συναρτήσεις αποτελούν μεθόδους μιας κλάσης.
- ☑ Οι φίλιες συναρτήσεις μιας κλάσης έχουν πρόσβαση στα ιδιωτικά μέλη ενός αντικειμένου της κλάσης.

14.2

```
#include <iostream>
using namespace std;

class triangle
{
   float basi;
   float ypsos;
public:
   static int pl;
   triangle(){pl++;}
   triangle(float b, float y){pl++;basi=b;ypsos=y;}
   float emvado(){return basi*ypsos/2;}
   void set(float b, float y){basi=b;ypsos=y;}
   void show(){cout<<"Βάση= "<<basi<<" Ύψος="<<ypsos<<endl;}
};

int triangle::pl=0;

void do_it(triangle &tr)
{
   tr.set(10,8);
}

int main()
{
   triangle tr1(3,5),tr2,*ptr;
   ptr=&tr2;
   tr1.show();
   do_it(tr1);
   tr1.show();
   ptr->set(4,7);
   tr2.show();
   cout<<triangle::pl<<endl;
   cout<<tr1.pl<<endl;
   return 0;
}
```

```
Βάση= 3 Ύψος= 5
Βάση= 10 Ύψος= 8
Βάση= 4 Ύψος= 7
2
2
```

Οι μέθοδοι δόμησης της κλάσης **triangle** αυξάνουν την τιμή της στατικής μεταβλητής **pl** κατά 1 κάθε φορά που δημιουργείται κάποιο αντικείμενο.

Η συνάρτηση **do_it()** έχει ως παράμετρο μια αναφορά σε αντικείμενα κλάσης **triangle**. Η συνάρτηση θέτει διαστάσεις 10 και 8 στο αντικείμενο με το οποίο καλείται.

Δημιουργία δύο αντικειμένων της κλάσης **triangle**, ενός αντικειμένου **tr1** με διαστάσεις 3 και 5, και ενός **tr2** χωρίς συγκεκριμένες διαστάσεις. Επίσης δηλώνεται ένας δείκτης σε αντικείμενα **triangle**.

Στον δείκτη **ptr** ανατίθεται η διεύθυνση του **tr2**.

Η κλήση της μεθόδου **show()** εμφανίζει τις διαστάσεις του αντικειμένου **tr1**. Έπειτα, η κλήση της **do_it()** θέτει στο **tr1** διαστάσεις 10 και 8.

Καλείται η μέθοδος **set()** για το αντικείμενο στο οποίο δείχνει ο δείκτης **ptr** δηλαδή για το **tr2**.

Εμφάνιση της τιμής της στατικής μεταβλητης **pl** με αναφορά μόνο στην κλάση αλλά και μέσω ενός αντικειμένου της κλάσης.

14.3

```
float total(box bb[])
{
   int i;
   float total=0;
   for (i=0;i<100;i++) total=total+bb[i].volume();
   return total;
}

box max_box(box bb[])
{
   int i,max_i;
   float max_vol;
   max_vol=bb[0].volume();
   max_i=0;
   for (i=0;i<100;i++)
   {
      if (bb[i].volume()>max_vol)
      {
         max_vol=bb[i].volume();
         max_i=i;
      }
   }
   return bb[max_i];
}
```

Καλείται η μέθοδος **volume()** για κάθε ένα από τα αντικείμενα **box** του πίνακα. Ο συνολικός όγκος καταχωρίζεται στη μεταβλητή **total**.

Στη μεταβλητή **max_vol** καταχωρίζεται η μέγιστη τιμή του όγκου των αντικειμένων του πίνακα.

Στη μεταβλητή **max_i** καταχωρίζεται ο αριθμοδείκτης του αντικειμένου του πίνακα με τον μεγαλύτερο όγκο.

Η συνάρτηση **max_box()** επιστρέφει ως τιμή το αντικείμενο με τον μεγαλύτερο όγκο.

14.4

```
class box
{
   float x;
   float y;
   float z;
public:
   void set_xyz(float a, float b, float c)
   {
      x=a;
      y=b;
      z=c;
   }
   float volume(){return x*y*z;}
   float new_volume()
   {
      return this->x*this->y*this->z;
   }
};
.....
.....
int main()
{
   box b1,*ptr;
   ptr=&b1;
   ptr->set_xyz(10,10,10);
   cout << ptr->volume();
   cout << b1.new_volume();
   return 0;
}
```

Η μέθοδος **new_volume()** χρησιμοποιεί τον δείκτη **this** για την πρόσβαση στις μεταβλητές μέλη του αντικειμένου για το οποίο καλείται.

Στον δείκτη **ptr** καταχωρίζεται η διεύθυνση του αντικειμένου **b1**.

Οι μέθοδοι **set_xyz()** και **volume()** καλούνται μέσω του δείκτη **ptr**.

Καλείται η νέα μέθοδος **new_volume()** για το αντικείμενο **b1**.

14.5

```
#include <iostream>
using namespace std;

class box
{
   float x;
   float y;
   float z;
   static float total_vol;
public:
   box(float a, float b, float c)
   {
      x=a;
      y=b;
      z=c;
      total_vol=total_vol+volume();
   }
   ~box()
   {
      total_vol=total_vol-volume();
   }
   float volume(){return x*y*z;}
   static void show_vol()
   {
      cout << "Συνολικός όγκος:" << total_vol << endl;
   }
};

float box::total_vol=0;

int main()
{
   box::show_vol();
   box b1(10,20,5),b2(2,3,4);
   box::show_vol();
   {
      box b3(10,5,5);
      box::show_vol();
   }
   box::show_vol();
   return 0;
}
```

Η στατική μεταβλητή-μέλος **total_vol** θα χρησιμοποιηθεί για την αποθήκευση του συνολικού όγκου των αντικειμένων.

Στη μέθοδο δόμησης προστίθεται μια πρόταση η οποία προσθέτει τον όγκο του αντικειμένου που δημιουργείται στον τρέχοντα συνολικό όγκο.

Δημιουργείται μια μέθοδος αποδόμησης η οποία αφαιρεί τον όγκο του αντικειμένου που καταστρέφεται, από τον τρέχοντα συνολικό όγκο.

Η στατική μέθοδος **show_vol()** εμφανίζει τον συνολικό όγκο των αντικειμένων. Επειδή είναι στατική, μπορεί να καλείται χωρίς να υπάρχει κάποιο αντικείμενο.

Ορισμός και ανάθεση αρχικής τιμής στη στατική μεταβλητή-μέλος **total_vol**, της κλάσης **box**.

```
Συνολικός όγκος:0
Συνολικός όγκος:1024
Συνολικός όγκος:1274
Συνολικός όγκος:1024
```

Το αντικείμενο **b3** «ζει» μόνο μέσα στη σύνθετη πρόταση. Μόλις λήξει η εμβέλειά του καταστρέφεται!

14.6

```
#include <iostream>
using namespace std;
class akeraios
{
   int ar;
public:
   static int count;
   void show() {cout << ar << endl;}
   int get() {return ar;}
   void set(int x) {ar=x;}
   akeraios(){count++;cout<<"Νέος αριθμός"<<endl;}
   ~akeraios(){count--;}
};
```

Δηλώνεται η στατική μεταβλητή-μέλος **count**, στην οποία θα τηρείται το πλήθος των αντικειμένων της κλάσης.

Μέθοδοι για την εμφάνιση, ανάκτηση και καταχώριση της τιμής της μοναδικής μεταβλητής-μέλους της κλάσης.

Μέθοδοι δόμησης και αποδόμησης στις οποίες μεταβάλλεται ανάλογα η τιμή της **count**.

```
void double_it(akeraios &a)
{
   int temp;
   temp=a.get();
   a.set(2*temp);
}
akeraios conv(int a)
{
   akeraios temp;
   temp.set(a);
   return temp;
}

int akeraios::count=0;

int main()
{
   akeraios ak1,ak2;
   cout << "Πλήθος="<<akeraios::count<<endl;
   ak1.set(10);
   ak1.show();
   double_it(ak1);
   ak1.show();
   ak2=conv(23);
   ak2.show();
   cout << "Πλήθος="<<akeraios::count<<endl;
   return 0;
}
```

Η συνάρτηση **double_it()** δέχεται ως παράμετρο ένα αντικείμενο της κλάσης **akeraios** με αναφορά. Επομένως, η συνάρτηση μπορεί να αλλάξει την τιμή του αντικειμένου με το οποίο καλείται.

Η συνάρτηση **conv()** επιστρέφει ως τιμή ένα αντικείμενο της κλάσης **akeraios**, του οποίου η μεταβλητή-μέλος παίρνει ως τιμή την τιμή της παραμέτρου με την οποία καλείται η συνάρτηση.

Δέσμευση της μνήμης για τη στατική μεταβλητή-μέλος **count** και απόδοση αρχικής τιμής.

```
Νέος αριθμός
Νέος αριθμός
Πλήθος=2
10
20
Νέος αριθμός
23
Πλήθος=2
```

☞ Δημιουργούνται δύο νέα αντικείμενα της κλάσης akeraios. Η μέθοδος δόμησης της κλάσης καλείται δύο φορές και η τιμή της στατικής μεταβλητής-μέλος count αποκτά την τιμή 2.

☞ Η πρόταση ak1.set(10) θέτει την τιμή 10 στη μεταβλητή-μέλος του αντικειμένου ak1. Η πρόταση ak1.show() εμφανίζει την τιμή της μεταβλητής-μέλους του αντικειμένου ak1.

☞ Η κλήση της συνάρτησης double_it() με πραγματική παράμετρο ak1, διπλασιάζει την τιμή της μεταβλητής-μέλους του αντικειμένου ak1.

☞ Η παράσταση conv(23) επιστρέφει ως τιμή ένα προσωρινό αντικείμενο της κλάσης akeraios με τιμή μεταβλητής-μέλους το 23. Το αντικείμενο καταχωρίζεται στο ak2.

☞ Παρατηρούμε ότι το πλήθος των αντικειμένων της κλάσης akeraios παραμένει 2 διότι το προσωρινό αντικείμενο που επιστρέφει η conv() δημιουργείται και καταστρέφεται αμέσως μετά την ανάθεσή του στο ak2. Εξάλλου, η δημιουργία του προσωρινού αντικειμένου δεν καλεί τη μέθοδο δόμησης αλλα τη μέθοδο δόμησης αντιγράφου η οποία δεν έχει οριστεί.

14.7

☑ Η κλάση string μπορεί να αντικαταστήσει πλήρως τους πίνακες χαρακτήρων για την αποθήκευση συμβολοσειρών.

☑ Ένα αντικείμενο string διαχειρίζεται με δυναμικό τρόπο τη μνήμη που καταλαμβάνει.

❑ Ο χώρος που καταλαμβάνει μια μεταβλητή-μέλος string μιας κλάσης, εξαρτάται από το πλήθος χαρακτήρων οι οποίοι είναι καταχωρισμένοι σε αυτή.

❑ Οι συναρτήσεις βιβλιοθήκης που χρησιμοποιούνται για τη διαχείριση συμβολοσειρών της C (C-strings) μπορούν να χρησιμοποιηθούν και με αντικείμενα string.

14.8

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
   string s("Μυτιλήνη ");
   string s1="123456789",s2;
   char ch='#';
   int i;
   cout<<s1<<endl<<"========="<<endl;
   s2="είναι";
   s2.append("ένα");
   cout<<s2<<endl;
   s2.insert(2,s);
   cout<<s2<<endl;
   s1=s2.substr(2,9);
   cout<<s1<<endl;
   s1.erase(5);
   cout<<s1<<endl;
   cout<<s1.capacity()<<endl;
   cout<<s1.size()<<endl;
   s1.resize(20,ch);
   cout<<s1<<endl;
   i=s2.find(s);
   cout<<i<<endl;
   i=s2.find_first_not_of(s);
   cout << s2[i]<<endl;
   return 0;
}
```

```
123456789
=========
είναιένα
είΜυτιλήνη ναιένα
Μυτιλήνη
Μυτιλ
9
5
Μυτιλ###############
2
e
```

14.9

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
   string lex[5]={"Νίκος","Κατερίνα","Χρυσάνθη","Μαρία","Βασίλης"};
   int i,j,p;
   // Εμφανίζει όλα τα ονόματα αντίστροφα
   for (i=0;i<5;i++)
   {
      for (j=lex[i].size()-1;j>=0;j--)
         cout << lex[i][j];
      cout << endl;
   }
   // Εμφανίζει τα τελευταία 2 γράμματα κάθε ονόματος
   for (i=0;i<5;i++)
   {
      cout << lex[i].substr(lex[i].length()-2,2) << endl;
   }
   // Εμφανίζει το τμήμα του ονόματος από το γράμμα "ί" και μετά (αν υπάρχει)
   for (i=0;i<5;i++)
   {
      p=lex[i].find('ί');
      if (p!=string::npos)
         cout << lex[i].substr(p) << endl;
   }
```

```
ςοκίΝ
ανίρεταΚ
ηθνάσυρΧ
αίραΜ
ςηλίσαΒ
ος
να
θη
ία
ης
ίκος
ίνα
ία
ίλης
Χρυσάνθη
```

```
   // Εμφανίζει τα ονόματα που περιέχουν το αλφαριθμητικό "άν"
   for (i=0;i<5;i++)
   {
      if (lex[i].rfind("άν")!=string::npos)
         cout << lex[i] << endl;
   }
   return 0;
}
```

14.10

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string o[10];
   int i,j,k;
   bool found,koino,yparxoyn_koina=0;
   for (i=0;i<10;i++)
   {
      cout<<"Δώσε όνομα  -> ";
      getline(cin,o[i]);
   }
   for (i=0;i<o[0].size();i++)
   {
      koino=true;
      for (j=1;j<10;j++)
      {
         found=false;
         for (k=0;k<o[j].size();k++)
            if (o[0][i]==o[j][k]) found=true;
         if (found==false)
         {
            koino=false;
            break;
         }
      }
      if (koino)
      {
         cout << o[0][i] << endl;
         yparxoyn_koina=true;
      }
   }
   if (!yparxoyn_koina)
      cout << "Δεν υπάρχουν κοινοί χαρακτήρες" << endl;
   return 0;
}
```

Τα ονόματα που πληκτρολογεί ο χρήστης καταχωρίζονται στον πίνακα **o**!

Αρχικά θεωρούμε ότι ο χαρακτήρας είναι κοινός

Ελέγχουμε κάθε χαρακτήρα του πρώτου ονόματος με τους χαρακτήρες των υπόλοιπων ονομάτων.

Αν ο χαρακτήρας δεν υπάρχει τουλάχιστον σε ένα όνομα, τότε σίγουρα δεν είναι κοινός και προχωρούμε στον επόμενο.

Αν ο χαρακτήρας είναι κοινός σε όλα τα ονόματα τον εμφανίζει!

14.11

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
using namespace std;

string anagrammatismos(string lex);

int main()
{
   string str1="Προγραμματισμός";
```

Τα αρχεία κεφαλίδας **cstdlib** και **ctime** έχουν συμπεριληφθεί για να είναι δυνατή η χρήση των συναρτήσεων **rand()**, **srand()** και **time()**.

```
   cout << anagrammatismos(str1) << endl;
   return 0;
}

string anagrammatismos(string lex)
{
   int t1,t2,i;
   char ch;
   srand(time(NULL));
   for (i=1;i<=100;i++)
   {
      t1=rand()%lex.size();
      t2=rand()%lex.size();
      ch=lex[t1];
      lex[t1]=lex[t2];
      lex[t2]=ch;
   }
   return lex;
}
```

Η **srand()** χρησιμοποιείται για τη δημιουργία διαφορετικών κάθε φορά τυχαίων αριθμών από τη **rand()**.

Η διαδικασία εκτελείται 100 φορές για να γίνει καλό ανακάτεμα! Το πλήθος επαναλήψεων μπορεί να είναι και διαφορετικό.

Οι τυχαίοι αριθμοί **t1** και **t2** υποδεικνύουν θέσεις χαρακτήρων μέσα στην παράμετρο **lex**. Έχουν τιμές από 0 έως τον τελευταίο χαρακτήρα της παραμέτρου.

Γίνεται αντιμετάθεση των χαρακτήρων των θέσεων **t1** και **t2**.

14.12

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string o[10];
   int i,space_pos;
   for (i=0;i<10;i++)
   {
      cout<<"Δώσε όνοματεπώνυμο  -> ";
      getline(cin,o[i]);
   }
   for (i=0;i<10;i++)
   {
      space_pos=o[i].find(' ');
      cout << o[i].substr(space_pos+1)<<endl;
   }
   for (i=0;i<10;i++)
   {
      space_pos=o[i].find(' ');
      cout << o[i].substr(0,space_pos)<<endl;
   }
   return 0;
}
```

Εντοπίζει τον χαρακτήρα του διαστήματος και καταχωρίζει τη θέση του στη μεταβλητή **space_pos**.

Εμφανίζει τους χαρακτήρες αμέσως μετά τη θέση του διαστήματος μέχρι τέλους.

Εμφανίζει τους χαρακτήρες από την αρχή μέχρι αμέσως πριν τη θέση του διαστήματος.

14.13

```
#include <iostream>
#include <string>
using namespace std;
class my_date
{
   int hmera;
   int minas;
   int etos;
public:
   bool set_hmer(int h, int m, int e);
   void show();
};
```

```
bool my_date::set_hmer(int h, int m, int e)
{
   bool disekto;
   int hmeres_minon[12]={31,28,31,30,31,30,31,31,30,31,30,31};
   if (e<=0)
      return false;
   disekto=(e%4==0 && e%100!=0) || e%400==0;
   if (disekto)
      hmeres_minon[1]=29;
   if (!(m>=1 && m<=12))
      return false;
   if (!(h>=1 && h<=hmeres_minon[m-1]))
      return false;
   hmera=h;
   minas=m;
   etos=e;
}
void my_date::show()
{
   cout<<hmera<<"/"<<minas<<"/"<<etos<<endl;
}

int main()
{
   my_date e1;
   if (e1.set_hmer(29,2,2016))
      e1.show();
   else
      cout << "Λάθος ημερομηνία" << endl;
   if (e1.set_hmer(31,4,2013))
      e1.show();
   else
      cout << "Λάθος ημερομηνία" << endl;
   return 0;
}
```

Ελέγχεται αν το έτος είναι αποδεκτό.

Στην περίπτωση που το έτος είναι δίσεκτο, ο Φεβρουάριος έχει 29 ημέρες!

Ελέγχεται αν ο μήνας είναι εντός των σωστών ορίων.

Ελέγχεται αν η ημέρα είναι εντός των ορίων του αντίστοιχου μήνα.

Καταχώριση των αποδεκτών πλέον τιμών στις μεταβλητές-μέλη του αντικειμένου

```
29/2/2016
Λάθος ημερομηνία
```

14.14

- ☑ Μια κλάση μπορεί να διαθέτει μεταβλητές-μέλη που να είναι αντικείμενα άλλης κλάσης. Αυτή η κλάση λέγεται σύνθετη κλάση.
- ❑ Πρώτα δημιουργείται ένα αντικείμενο μιας σύνθετης κλάσης και μετά τα αντικείμενα-μέλη που περιέχει.
- ☑ Οι λίστες αρχικοποίησης αφορούν μόνο τις μεθόδους δόμησης.
- ❑ Οι λίστες αρχικοποίησης είναι υποχρεωτικές σε περιπτώσεις σύνθετων κλάσεων.

14.15

Στην περίπτωση αυτή δεν θα μπορούσαν να δημιουργηθούν αντικείμενα της κλάσης ergasia με χρήση της προκαθορισμένης μεθόδου δόμησης της κλάσης (δηλαδή χωρίς ορίσματα), διότι η δημιουργία ενός αντικειμένου με αυτόν τον τρόπο δημιουργεί ταυτόχρονα και ένα αντικείμενο της κλάσης my_date, επίσης με χρήση της προκαθορισμένης μεθόδου δόμησης αυτής της κλάσης. Αν αυτή δεν υπάρχει, τότε δεν είναι δυνατή ούτε η δημιουργία αντικειμένων της κλάσης ergasia με αυτόν τον τρόπο. Επομένως, η πρόταση:

```
ergasia e1,e2("Λίστες αρχικοποίησης",5,2,2015);
```

δεν θα μεταγλωττιστεί γιατί δεν μπορεί να δημιουργηθεί το αντικείμενο e1!

14.16

```
#include <iostream>
#include <string>
using namespace std;
class box
{
   float x;
   float y;
   float z;
public:
   box(float a, float b, float c):x(a),y(b),z(c) {}
   box():x(0),y(0),z(0) {}
   box(int a):x(a),y(a),z(a) {}
   float volume(){return x*y*z;}
};

int main()
{
   box b1,b2(10),b3(3,5,8);
   cout<<b1.volume()<<endl;
   cout<<b2.volume()<<endl;
   cout<<b3.volume()<<endl;
   return 0;
}
```

14.17

```
my_date d1,d2(15,6,2014),d3(4,5);
d1.show();
d2.show();
d3.show();
```

1/1/1
15/6/2014
4/5/2015

14.18

```
gamos mine,yours("Πουλχερία"),g3("Rihanna",5,6,2016);

mine.show();
yours.show();
g3.show();
```

Κανένας 1/1/1
Πουλχερία 1/1/2015
Rihanna 5/6/2016

14.19

```
// Οι μέθοδοι display() των κλάσεων δεν ζητούνται αλλά έχουν προστεθεί
// για να γίνει κατανοητή και να δοκιμαστεί η λειτουργία του προγράμματος
// ΠΡΟΣΟΣΧΗ!!! κάποιες λίστες αρχικοποίησης αναδιπλώνονται σε δύο γραμμές
#include <iostream>
#include <string>
using namespace std;
class point
{

   float x_pos;
   float y_pos;
public:
   point(float x, float y):x_pos(x),y_pos(y) {}
   void display()
   {
      cout << " Σημείο "<<x_pos<<","<<y_pos<<endl;
   }
};
```

Μέθοδος δόμησης με λίστα αρχικοποίησης η οποία καταχωρίζει τις παραμέτρους της μεθόδου ως θέση του σημείου (x_pos, y_pos).

```
class rectangle
{
   float plevra_x;
   float plevra_y;
   point thesi ;
public:
   rectangle(float pl, float yps, float x, float y):plevra_x(pl),
             plevra_y(yps),thesi(x,y) {}
   void display()
   {
      cout << "Παραλληλόγραμμο "<<plevra_x<<"x"<<plevra_y;
      thesi.display();
   }
};

class circle
{
public:
   float aktina;
   point center;
   circle(float r,float kentro_x,float kentro_y):aktina(r),center(kentro_x,kentro_y) {}
   void display()
   {
      cout << "Κύκλος ακτίνας "<<aktina;
      center.display();
   }
};
class arabas
{
public:
   rectangle karotsa;
   circle roda1;
   circle roda2;
   arabas(float x, float y, float platos, float ypsos, float akt):
   karotsa(platos,ypsos,x,y),roda1(akt,x,y),roda2(akt,x+platos,y) {}
   void display()
   {
      cout << "Καρότσα: ";
      karotsa.display();
      cout << "Πρώτη ρόδα: ";
      roda1.display();
      cout << "Δεύτερη ρόδα: ";
   roda2.display();
   }
};
int main()
{
   arabas karo(10,10,10,20,5);
   karo.display();
   return 0;
}
```

Μέθοδος δόμησης με τέσσερις παραμέτρους και λίστα αρχικοποίησης. Το μέλος **thesi** αρχικοποιείται χρησιμοποιώντας τη μέθοδο δόμησης της κλάσης **point**!

Μέθοδος δόμησης με τρεις παραμέτρους και λίστα αρχικοποίησης. Το μέλος **center** αρχικοποιείται με τη μέθοδο δόμησης της κλάσης **point**!

Μέθοδος δόμησης της κλάσης **arabas** με πέντε παραμέτρους και λίστα αρχικοποίησης. Η πρώτες δύο παράμετροι ορίζουν τη θέση του (X,Y), οι δύο επόμενες το πλάτος και το ύψος της καρότσας του, και η τελευταία την ακτίνα που θα έχουν οι δύο ρόδες του. Το μέλος **karotsa** αρχικοποιείται με τη μέθοδο δόμησης της κλάσης **rectangle**, και τα μέλη **roda1** και **roda2** αρχικοποιούνται με τη μέθοδο δόμησης της κλάσης **circle**.

Η μέθοδος **display()** εμφανίζει τα στοιχεία ενός αντικειμένου αραμπά, εφαρμόζοντας με τη σειρά της τις μεθόδους **display()** των επιμέρους μελών του.

Δημιουργείται ένα αντικείμενο της κλάσης **arabas**, με χρήση της μεθόδου δόμησης της κλάσης.

```
Καρότσα: Παραλληλόγραμμο 10x20 Σημείο 10,10
Πρώτη ρόδα: Κύκλος ακτίνας 5 Σημείο 10,10
Δεύτερη ρόδα: Κύκλος ακτίνας 5 Σημείο 20,10
```

14.20

```
class lastixo
{
public:
   string eidos;
   float diametros;
   float platos
   string marka;
};

class mpoyzi
{
public:
   string typos;
   string marka;
};

class roda
{
public:
   float diametros;
   int mpoylonia;
   lastixo las;
};

class mixani
{
public:
   int kivika;
   float aloga;
   mpoyzi mp[6];
};

class aytokinito
{
public:
   string marka;
   string xroma;
   mixani kinitiras;
   roda troxoi[4];
};
```

Το μέλος **las** της κλάσης **roda**, είναι αντικείμενο της κλάσης **lastixo**. Μία ρόδα έχει ένα λάστιχο. Αυτό υλοποιεί τη συσχέτιση σύνθεσης των δύο κλάσεων όπως φαίνεται στο διάγραμμα κλάσεων UML της εκφώνησης.

Η κλάση **mixani** περιέχει ως μέλος έναν πίνακα έξι αντικειμένων της κλάσης **mpoyzi**. Μία μηχανή επομένως διαθέτει έξι μπουζιά. Αυτό υλοποιεί τη συσχέτιση σύνθεσης των δύο κλάσεων όπως φαίνεται στο διάγραμμα κλάσεων UML της εκφώνησης.

Η κλάση **aytokinito** περιέχει ένα μέλος της κλάσης **kinitiras** και έναν πίνακα τεσσάρων αντικειμένων της κλάσης **roda**. Μία μηχανή επομένως διαθέτει έναν κινητήρα και τέσσερις ρόδες. Αυτό υλοποιεί τη συσχέτιση σύνθεσης των τριών κλάσεων όπως φαίνεται στο διάγραμμα κλάσεων UML της εκφώνησης.

14.21

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <iomanip>
using namespace std;

class trapoula
{
   string xartia[52];
public:
   trapoula();
   void suffle(int n);
   void display();
};

trapoula::trapoula()
{
   int i,j;
```

Στον πίνακα **xartia** ενός αντικειμένου της κλάσης θα καταχωριστούν τα 52 χαρτιά (φιγούρα και είδος) της τράπουλας.

Η μέθοδος **suffle()** ανακατεύει τα χαρτιά του πίνακα **xartia** πολλές φορές, ανάλογα με την τιμή της παραμέτρου **n**.

Η μέθοδος **display()** εμφανίζει τα χαρτιά της τράπουλας με τη σειρά που βρίσκονται μέσα στον πίνακα **xartia**.

Μέθοδος δόμησης της κλάσης.

```
   string eidos[4]={"Καρό","Κούπα","Μπαστούνι","Σπαθί"};
   string figoura[13]={"A","2","3","4","5","6","7","8","9","10","J","Q","K"};
   for (i=0;i<13;i++)
   {
      for (j=0;j<4;j++)
         xartia[i*4+j]=figoura[i]+" "+eidos[j];
   }
}

void trapoula::suffle(int n=5)
{
   int i,val;
   string temp;
   int k1,k2;
   srand(time(NULL));
   if (n<5) n=5;
   if (n>50) n=50;
   for (i=0;i<n*100;i++)
   {
      k1=rand()%52;
      k2=rand()%52;
      temp=xartia[k1];
      xartia[k1]=xartia[k2];
      xartia[k2]=temp;
   }
   cout << "***** Η ΤΡΑΠΟΥΛΑ ΑΝΑΚΑΤΕΥΤΗΚΕ "<<n<<" ΦΟΡΕΣ *****"<<endl;
}

void trapoula::display()
{
   int i;
   cout <<"*************************************************"<<endl;
   for (i=0;i<52;i++)
   {
      cout <<  xartia[i] << endl;
   }
   cout <<"*************************************************"<<endl;
}

int main()
{
   trapoula tr1;
   tr1.display();
   tr1.suffle();
   tr1.display();
   return 0;
}
```

Στον πίνακα **xartia** καταχωρίζονται όλοι οι συνδυασμοί φιγούρας και είδους.

Η μέθοδος **suffle()** ανακατεύει τα χαρτιά του πίνακα **xartia** πολλές φορές, ανάλογα με την τιμή της παραμέτρου **n**. Η προκαθορισμένη τιμή της παραμέτρου είναι 5.

Οι μεταβλητές **k1** και **k2** προσδιορίζουν δύο τυχαίες θέσεις του πίνακα. Έπειτα, αντιμεταθέτει τα περιεχόμενα αυτών των δύο θέσεων. Αυτό γίνεται 100*n φορες!

Η μέθοδος **display()** εμφανίζει τα χαρτιά της τράπουλας με τη σειρά που βρίσκονται μέσα στον πίνακα **xartia**.

Δημιουργεί ένα αντικείμενο της κλάσης **trapoula**. Μετά εμφανίζει τα χαρτιά της τράπουλας, την ανακατεύει και τα ξαναεμφανίζει.

Ασκήσεις Κεφαλαίου 15

15.1

- ☑ Η υπερφόρτωση τελεστών μπορεί να αλλάξει ριζικά την αρχική σημασία του τελεστή για μια κλάση. Για παράδειγμα, ο τελεστής + θα μπορούσε να κάνει αφαίρεση!
- ❑ Η υπερφόρτωση του τελεστή () αλλάζει τον τρόπο κλήσης των συναρτήσεων στη C++.
- ☑ Η υπερφόρτωση τελεστών με τη χρήση συναρτήσεων που δεν είναι μέλη μιας κλάσης γίνεται συνήθως όταν το αριστερό μέλος του τελεστή ανήκει σε κάποιο βασικό τύπο δεδομένων.
- ❑ Σε μια κλάση, αν δεν υπερφορτωθεί ο τελεστής ανάθεσης τιμής (=), δεν μπορεί να χρησιμοποιηθεί.
- ☑ Σε έναν υπερφορτωμένο διμελή τελεστή, το αντικείμενο που καλεί τη μέθοδο υπερφόρτωσης τελεστή είναι το αριστερό μέλος της διμελούς πράξης.

15.2

```
#include <iostream>
using namespace std;
class triangle
{
   float basi;
   float ypsos;
public:
   triangle(float b, float y){basi=b;ypsos=y;}
   triangle(){basi=0;ypsos=0;};
   void show()
   {
      cout<<"Βάση= "<<basi<<" Ύψος="<<ypsos<<endl;
   }
   triangle operator!();
   triangle operator-(triangle op2);
   triangle operator=(triangle op2);
};

triangle triangle::operator!()
{
   float temp;
   temp = ypsos;
   ypsos = basi;
   basi=temp;
   return *this;
}
triangle triangle::operator-(triangle op2)
{
   triangle temp;
   temp.basi=basi+op2.basi;
   temp.ypsos=ypsos+op2.ypsos;
   return temp;
}
triangle triangle::operator=(triangle op2)
{
   basi=op2.basi/2;
   ypsos=op2.ypsos/2;
   return *this;
}
```

Μέθοδος υπερφόρτωσης του τελεστή !. Όταν εφαρμοστεί σε αντικείμενα της κλάσης **triangle** αντιμεταθέτει τις τιμές του ύψους και της βάσης. Επιστρέφει ως τιμή το ίδιο το αντικείμενο!

Μέθοδος υπερφόρτωσης του τελεστή -. Όταν αφαιρούνται αντικείμενα της κλάσης **triangle** επιστρέφει ως τιμή ένα αντικείμενο rectangle με βάση το άθροισμα των βάσεων των δύο μελών και ύψος το άθροισμα των υψών των δύο μελών του τελεστή.

Μέθοδος υπερφόρτωσης του τελεστή ανάθεσης τιμής =. Όταν ανατίθεται ένα αντικείμενο **triangle** σε ένα άλλο, τότε στο αριστερό μέλος καταχωρίζονται οι μισές τιμές των διαστάσεων του δεξιού. Επιστρέφει ως τιμή το ίδιο το αριστερό αντικείμενο.

```
int main()
{
   triangle tr1(3,5),tr2,tr3(1,1);
   tr2.show();
   tr2=tr1-tr3;
   tr2.show();
   tr1.show();
   tr2=!tr1;
   tr1.show();
   tr2.show();
   return 0;
}
```

```
Βάση= 0 Ύψος=0
Βάση= 2 Ύψος=3
Βάση= 3 Ύψος=5
Βάση= 5 Ύψος=3
Βάση= 2.5 Ύψος=1.5
```

☞ Η παράσταση tr1-tr3 επιστρέφει ένα αντικείμενο με διαστάσεις 4,6 επειδή ο υπερφορτωμένος τελεστής - προσθέτει τις αντίστοιχες διαστάσεις των δύο αντικειμένων. Όμως, ο υπερφορτωμένος τελεστής ανάθεσης τιμής = θα εκχωρήσει στο αριστερό του μέλος, δηλαδή στο αντικείμενο tr2, τις μισές διαστάσεις του αντικειμένου που επιστρέφει η παράσταση. Επομένως θα αποκτήσει διάσταση βάσης 2 και ύψους 3.

☞ Η παράσταση !tr1 θα αντιμεταθέσει τη διάσταση της βάσης με τη διάσταση του ύψους. Επομένως το tr1 θα αποκτήσει διάσταση βάσης 5 και ύψους 3. Και πάλι, όμως, ο υπερφορτωμένος τελεστής ανάθεσης τιμής = θα εκχωρήσει στο αριστερό του μέλος, δηλαδή στο αντικείμενο tr2, τις μισές διαστάσεις του αντικειμένου tr1. Επομένως το αντικείμενο tr2 θα αποκτήσει διάσταση βάσης 2.5 και ύψους 1.5.

15.3

```
kafetiera kafetiera::operator()(int k, int g, int z, int n)
{
   kafes=k;
   gala=g;
   zaxari=z;
   nero=n;
   return *this;
}
```

☞ Η μέθοδος υπερφόρτωσης του τελεστή () έχει ως αποτέλεσμα την ανάθεση τιμών στις μεταβλητές-μέλη του αντικειμένου. Για παράδειγμα, η πρόταση: kaf1(10,20,100,200) θα έχει ως αποτέλεσμα την καταχώριση των τιμών 10,20,100 και 200 στις μεταβλητές-μέλη kafes, gala, zaxari και nero ενός αντικειμένου kaf1 της κλάσης kafetiera.

☞ Ο δείκτης this δείχνει στο αντικείμενο που καλεί τη μέθοδο υπερφόρτωσης. Η μέθοδος επιστρέφει ως τιμή το ίδιο το αντικείμενο (*this).

15.4

```
bool kafetiera::operator==(kafetiera op2)
{
   if (kafes==op2.kafes && gala==op2.gala && zaxari==op2.zaxari && nero==op2.nero)
      return true;
   else
      return false;
}
```

☞ Η μέθοδος υπερφόρτωσης του τελεστή == επιστρέφει τιμή bool. Η μέθοδος ελέγχει αν όλες οι τιμές των μεταβλητών-μελών του αριστερού μέλους του τελεστή ισούνται με τις αντίστοιχες τιμές των μελών του δεξιού μέλους του τελεστή.

15.5

```
kafetiera kafetiera::operator*(int value)
{
   kafetiera temp;
   temp.kafes=kafes*value;
   temp.gala=gala*value;
   temp.zaxari=zaxari*value;
   temp.nero=nero*value;
   return temp;
}
kafetiera operator*(int value, kafetiera op2)
{
   kafetiera temp;
   temp.kafes=op2.kafes*value;
   temp.gala=op2.gala*value;
   temp.zaxari=op2.zaxari*value;
   temp.nero=op2.nero*value;
   return temp;
}
```

☞ Η πρώτη μέθοδος υπερφόρτωσης του τελεστή * ορίζεται μέσα στην κλάση kafetiera. Η έκδοση αυτή καλείται όταν το αριστερό μέλος του τελεστή * είναι αντικείμενο της κλάσης kafetiera και το δεξιό μέλος είναι ακέραιος αριθμός. Π.χ. kaf1*10.

☞ Η δεύτερη συνάρτηση υπερφόρτωσης του τελεστή * ορίζεται εκτός της κλάσης kafetiera. Ο ορισμός της συνάρτησης εκτός της κλάσης είναι απαραίτητος όταν κατά την κλήση του διμελούς τελεστή * το αριστερό του μέλος δεν είναι αντικείμενο της κλάσης kafetiera αλλά ένας ακέραιος αριθμός. Η έκδοση αυτή καλείται όταν το αριστερό μέλος είναι ακέραιος αριθμός και το δεξιό μέλος είναι αντικείμενο της κλάσης kafetiera. Π.χ. 10*kaf1. Η συνάρτηση αυτή πρέπει να δηλωθεί ως φίλια της κλάσης kafetiera για να μπορεί να έχει πρόσβαση στα ιδιωτικά μέλη της κλάσης

15.6

```
#include <iostream>
using namespace std;
class triangle
{
   float basi;
   float ypsos;
public:
   triangle(float b, float y){basi=b;ypsos=y;}
   triangle(){};
   void show(){cout<<"Βάση= "<<basi<<" Ύψος="<<ypsos<<endl;}
   triangle operator++();
   triangle operator++(int);
};
triangle triangle::operator++()
{
   basi++;
   return *this;
}
triangle triangle::operator++(int)
{
   ypsos++;
   return *this;
}
int main()
{
   triangle tr1(3,5);
   tr1.show();
```

Μέθοδος υπερφόρτωσης του επιθηματικού τελεστή ++. Η εφαρμογή του τελεστή θα έχει ως αποτέλεσμα την αύξηση της τιμής του ύψους του αντικειμένου κατά 1. Η μέθοδος επιστρέφει το ίδιο το αντικείμενο.

```
   ++tr1;
   tr1.show();
   tr1++;
   tr1.show();
   return 0;
}
```

☞ Οι μέθοδοι υπερφόρτωσης του προθεματικού και του επιθηματικού τελεστή ξεχωρίζουν μεταξύ τους από τη ψευδοπαράμετρο τύπου int που διαθέτει η μέθοδος υπερφόρτωσης του επιθηματικού τελεστή.

15.7

```
class ergasia
{
   .....
   string &operator[](int i);
};

string &ergasia::operator[](int i)
{
   if (i>=1 && i<=plithos_foititon)
      return onomata[i-1];
}
int main()
{
   .....
   e1[2]="Νίκος";
   e1.display();
   return 0;
}
```

Η μόνη αλλαγή που πρέπει να γίνει είναι να οριστεί η μέθοδος υπερφόρτωσης του τελεστή **[]** ως αναφορική με χρήση του προθέματος **&**. Τώρα η μέθοδος επιστρέφει μια αναφορά στη θέση μνήμης του πίνακα **onomata**.

Η πρόταση αυτή θα καταχωρίσει στη δεύτερη θέση του πίνακα **onomata** του αντικειμένου **e1** το «Νίκος».

15.8

```
#include <iostream>
#include <string>
using namespace std;
class my_date
{
   int hmera;
   int minas;
   int etos;
public:
   void set_hmer(int h, int m, int e);
   void show();
   my_date operator++();
   my_date operator--();
   int operator-(my_date op2);
};
void my_date::set_hmer(int h, int m, int e)
{
   hmera=h;
   minas=m;
   etos=e;
}
void my_date::show()
{
   cout<<hmera<<"/"<<minas<<"/"<<etos<<endl;
}

my_date my_date::operator++()
{
   hmera++;
   if (hmera>30)
```

```
    {
       hmera=1;
       minas++;
       if (minas>12)
       {
          minas=1;
          etos++;
       }
    }
    return *this;
}

my_date my_date::operator--()
{
    hmera--;
    if (hmera==0)
    {
       hmera=30;
       minas--;
       if (minas==0)
       {
          minas=12;
          etos--;
       }
    }
    return *this;
}

int my_date::operator-(my_date op2)
{
    int hmeres1,hmeres2;
    hmeres1=etos*360+minas*30+hmera;
    hmeres2=op2.etos*360+op2.minas*30+op2.hmera;
    if (hmeres1>hmeres2)
       return hmeres1-hmeres2;
    else
       return hmeres2-hmeres1;
}

int main()
{
    my_date d1,d2,d3;
    d1.set_hmer(1,1,2016);
    d2.set_hmer(3,7,1980);
    d3.set_hmer(8,3,1960);
    d1.show();
    --d1;                  // μειώνει την ημερομηνία d1 κατά μία ημέρα
    d1.show();
    cout<<d1-d2<<endl; // εμφανίζει τη διαφορά σε ημέρες των ημερομηνιών d1 και d2
    cout<<d3-d2<<endl; // εμφανίζει τη διαφορά σε ημέρες των ημερομηνιών d3 και d2
    return 0;
}
```

Μέθοδος υπερφόρτωσης του προθεματικού τελεστή **--**. Η εφαρμογή του τελεστή θα έχει ως αποτέλεσμα τη μείωση της ημερομηνίας κατά 1 ημέρα. Αν είναι η 1η του μήνα, η μείωση επηρεάζει και τον μήνα και το έτος. Η μέθοδος επιστρέφει ως τιμή το ίδιο το αντικείμενο.

Μέθοδος υπερφόρτωσης του τελεστή **-**. Υπολογίζονται τα σύνολα των ημερών των ημερομηνιών του αριστερού και του δεξιού μέλους του τελεστή. Ως τιμή επιστρέφεται η απόλυτη τιμή της διαφοράς τους. Κάθε μήνας υπολογίζεται με 30 ημέρες.

```
1/1/2016
30/12/2015
12777
7315
```

15.9

```
#include <iostream>
#include <string>
using namespace std;

class arithmos
{
    int a;
```

```
   string dek(int ar);
   string ek(int ar);
public:
   int operator=(int op2);
   string operator!();
};
string arithmos::ek(int ar)
{
   string ekatontades[10]={"","εκατό","διακόσια","τριακόσια",
             "τετρακόσια","πεντακόσια","εξακόσια","επτακόσια",
             "οκτακόσια","εννιακόσια"};
   int e;
   e=ar/100;
   return ekatontades[e]+" "+dek(ar);
}
string arithmos::dek(int ar)
{
   string monades[11]={"","ένα","δύο","τρία","τέσσερα",
             "πέντε","έξι","επτά","οκτώ","εννιά","δέκα"};
   string dekades[10]={"","δέκα","είκοσι","τριάντα","σαράντα",
             "πενήντα","εξήντα","εβδομήντα","ογδόντα","εννενήντα"};
   int m,d;
   m=ar%10;
   d=(ar%100)/10;
   if (d==1 && m==1)
      return "έντεκα";
   else if (d==1 && m==2)
      return "δώδεκα";
   return dekades[d]+" "+monades[m];
}
int arithmos::operator=(int op2)
{
   a=op2;
   return a;
}
string arithmos::operator!()
{
   string res="";
   if (a==0)
      res="μηδέν";
   res=res+ek(a);
   return res;
}
int main()
{
   arithmos arith;
   arith=524;
   cout<<!arith;
   return 0;
}
```

Η μέθοδος **ek()** επιστρέφει ως τιμή τον αριθμό της παραμέτρου της ολογράφως.

Στη μεταβλητή **e** καταχωρίζεται το πλήθος των εκατοντάδων του αριθμού.

Η μέθοδος **dek()** επιστρέφει ως τιμή τον αριθμό των δύο τελευταίων ψηφίων της παραμέτρου **ar** ολογράφως.

Στη μεταβλητή **m** καταχωρίζεται το πλήθος των μονάδων του αριθμού της παραμέτρου **ar** και στη μεταβλητή **d** το πλήθος των δεκάδων.

Μέθοδος υπερφόρτωσης του τελεστή ανάθεσης τιμής =. Καταχωρίζει τον ακέραιο του δεξιού μέλους στη μεταβλητή-μέλος **a** του αντικειμένου του αριστερού του μέλους.

Μέθοδος υπερφόρτωσης του τελεστή !. Χρησιμοποιώντας τη μέθοδο **ek()** επιστρέφει ως τιμή τον αριθμό της μεταβλητής-μέλους **a** ολογράφως.

```
πεντακόσια είκοσι τέσσερα
```

☞ Το πρόγραμμα θα εμφανίσει τον αριθμό 524 ολογράφως!

☞ Θα χρησιμοποιήσει τη μέθοδο υπερφόρτωσης του τελεστή ανάθεσης τιμής = και θα καταχωρίσει στη μεταβλητή-μέλος a του αντικειμένου arith τον ακέραιο 524.

☞ Η παράσταση !arith εφαρμόζει τη μέθοδο υπερφόρτωσης του τελεστή ! στο αντικείμενο arith. Επιστρέφει ως τιμή τον αριθμό της μεταβλητής-μέλος a του αντικειμένου ολογράφως.

Ασκήσεις Κεφαλαίου 16

16.1

- ☑ Η παράγωγη κλάση δεν έχει πρόσβαση στα ιδιωτικά μέλη της βασικής κλάσης.
- ❑ Η παράγωγη κλάση δεν έχει πρόσβαση στα προστατευμένα μέλη της βασικής κλάσης.
- ❑ Όταν υπάρχει μια μέθοδος με το ίδιο όνομα σε μια παράγωγη και σε μια βασική κλάση, η κλήση αυτής της μεθόδου για ένα αντικείμενο της παράγωγης κλάσης έχει ως αποτέλεσμα την κλήση της αντίστοιχης μεθόδου της βασικής κλάσης.
- ❑ Όταν σε μια βασική κλάση υπάρχει μια μέθοδος με όνομα func(), τότε τα αντικείμενα της παράγωγης κλάσης μπορούν *πάντα* να προσπελάζουν τη μέθοδο func() της βασικής κλάσης.
- ❑ Όταν δημιουργείται ένα αντικείμενο μιας παράγωγης κλάσης, πρώτα εκτελείται η μέθοδος δόμησης της παράγωγης κλάσης και έπειτα της βασικής.

16.2

Στην κληρονομικότητα, από μια κλάση μπορεί να δημιουργηθεί μια νέα *κλάση*. Η αρχική κλάση λέγεται *βασική* ενώ η νέα *παράγωγη*. Η νέα κλάση έχει πρόσβαση σε όλα τα μέλη της *βασικής* κλάσης εκτός από τα *ιδιωτικά*.

16.3

Στην πρόταση class A:public B, βασική κλάση είναι η κλάση *B* και παράγωγη η κλάση *A*. Το πρόθεμα *public* λέγεται προσδιοριστικό πρόσβασης κληρονομικότητας και ορίζει τον τρόπο με τον οποίο η παράγωγη κλάση κληρονομεί τα μέλη της βασικής. Κληρονομικότητα πολλών επιπέδων έχουμε όταν μια *παράγωγη* κλάση αποτελεί τη *βασική* κλάση για τη δημιουργία μιας άλλης κλάσης.

16.4

```
#include <iostream>
#include <cmath>
#define PI 3.141593
using namespace std;

class circle_based_shape
{
   float aktina;
public:
   float emvado(){return pow(get_r(),2)*PI;}
   float perimetros(){return 2*get_r()*PI;}
   void set_r(float r){aktina=r;}
   float get_r() {return aktina;}
};

class konos:public circle_based_shape
{
public:
   float ypsos;
   float ogos() {return  emvado()*ypsos/3;}
};

int main()
{
   konos k1;
   k1.ypsos=10;
   k1.set_r(5);
```

Η κλάση **konos** παράγεται από την κλάση **circle_based_shape** με δημόσια πρόσβαση.

Για τον υπολογισμό του όγκου χρησιμοποιείται η μέθοδος **emvado()** η οποία επιστρέφει το εμβαδό της βάσης του κώνου.

Θέτει το ύψος και την ακτίνα της βάσης του κώνου.

```
   cout<<"Όγκος κώνου: "<<k1.ogos()<<" m3"<<endl;
   return 0;
}
```

16.5

```
#include <iostream>
#include <string>
using namespace std;

class boats
{
   float mikos;
   string onoma;
public:
   void display(){cout<<onoma<<" "<<mikos<<endl;}
   void set(string onom, float mk);
};
void boats::set(string onom, float mk)
{
   onoma=onom;
   mikos=mk;
}
class foyskoto:public boats
{
   int aerothalamoi;
public:
   void set_all(string onom, float mk, int aer);
   void display_all();
};
void foyskoto::set_all(string onom, float mk, int aer)
{
   set(onom,mk);
   aerothalamoi=aer;
}
void foyskoto::display_all()
{
   display();
   cout<<aerothalamoi<<endl;
}

int main()
{
   foyskoto f1,f2;
   f1.set_all("Skipper",7.80,7);
   f2.set_all("Planatech",5.70,5);
   f1.display_all();
   f2.display_all();
   return 0;
}
```

Ιδιωτικά μέλη της κλάσης **boats**.

Η κλάση **boats** πρέπει οπωσδήποτε να διαθέτει μια δημόσια μέθοδο μέσω της οποίας θα καταχωρίζονται τιμές στις ιδιωτικές μεταβλητές-μέλη της.

Ιδιωτικά μέλη της κλάσης **foyskoto**. Η κλάση παράγεται από την **boats** με δημόσια πρόσβαση.

Η μέθοδος **set_all()** καταχωρίζει δεδομένα στις μεταβλητές-μέλη των αντικειμένων της κλάσης **foyskoto**. Η μέθοδος χρησιμοποιεί τη δημόσια μέθοδο **set()** για την καταχώριση των στοιχείων του υπο-αντικειμένου της κλάσης **boats**.

Η μέθοδος **display_all()** εμφανίζει τα στοιχεία των αντικειμένων της κλάσης **foyskoto**. Η μέθοδος χρησιμοποιεί τη δημόσια μέθοδο **display()** για την εμφάνιση των στοιχείων του υπο-αντικειμένου της κλάσης **boats**.

```
Skipper 7.8
7
Planatech 5.7
5
```

16.6

Ναι, υπάρχει. Οι μέθοδοι της κλάσης sphere δεν έχουν πρόσβαση στη μεταβλητή-μέλος aktina της βασικής κλάσης, διότι είναι ιδιωτική.

Με το προσδιοριστικό **protected:**, η μεταβλητή-μέλος aktina γίνεται προστατευμένη για την κλάση circle_based_shapes, αλλά τώρα είναι προσπελάσιμη από την παράγωγη κλάση sphere:

```
class circle_based_shapes
{
protected:
   float aktina;
public:
   float emvado(){return pow(aktina,2)*PI;}
   void set_r(float r){aktina=r;}
};
```

16.7

Μέλος	Κλάση B	Κλάση C	Κλάση D	Κλάση E	Κλάση F
Ιδιωτικό μέλος κλάσης **A**	❹	❹	❹	❹	❹
Προστατευμένο μέλος κλάσης **A**	❸	❷	❸	❷	❹
Δημόσιο μέλος κλάσης A	❸	❷	❸	❷	❹
Δημόσιο μέλος κλάσης **B**		❺	❺	❺	❶
Προστατευμένο μέλος κλάσης **C**	❺		❸	❷	❺

16.8

```
class A
{
public:
   int A1;
};

class B:private A
{
public:
   int B1;
};

class F:public B
{
public:
   int F1;
};

class C:protected A
{
public:
   int C1;
};

class D:private C
{
public:
   int D1;
};

class E:public C
{
public:
   int E1;
};
```

16.9

```
#include <iostream>
#include <cstdlib>
using namespace std;

class A
{
public:
   int A1;
   A() {A1=rand();}
   void show() {cout<<"A1="<<A1<<endl;}
};

class B:private A
{
public:
   int B1;
   B() {B1=rand();}
   void show()
   {
      cout<<"B1="<<B1<<endl;
      A::show();
   }
};

class F:public B
{
public:
   int F1;
   F() {F1=rand();}
   void show()
   {
      cout<<"F1="<<F1<<endl;
      B::show();
   }
};

class C:protected A
{
public:
   int C1;
   C() {C1=rand();}
   void show()
   {
      cout<<"C1="<<C1<<endl;
      A::show();
   }
};

class D:private C
{
public:
   int D1;
   D() {D1=rand();}
   void show()
   {
      cout<<"D1="<<D1<<endl;
      C::show();
   }
};
```

Η μέθοδος δόμησης καταχωρίζει έναν τυχαίο αριθμό στη μεταβλητή-μέλος της κλάσης.

Καλείται η μέθοδος **show()** της κλάσης A.

Καλείται η μέθοδος **show()** της κλάσης B.

Καλείται η μέθοδος **show()** της κλάσης A.

Καλείται η μέθοδος **show()** της κλάσης C.

```
class E:public C
{
public:
   int E1;
   E() {E1=rand();}
   void show()
   {
      cout<<"E1="<<E1<<endl;
      C::show();
   }
};

int main()
{
   A obj_A;
   B obj_B;
   C obj_C;
   D obj_D;
   E obj_E;
   F obj_F;
   obj_F.show();
   obj_E.show();
   return 0;
}
```

Δημιουργείται ένα αντικείμενο από κάθε κλάση.

```
F1=16827
B1=23281
A1=28145
E1=5705
C1=24464
A1=26962
```

16.10

```
#include <iostream>
#include <cmath>
#define PI 3.141593
using namespace std;
class circle_based
{
   float aktina;
public:
   string color;
   float emvado(){return aktina*aktina*PI;}
   void display() {cout <<aktina<<" "<<emvado()<<endl;}
   circle_based(int r, string c):aktina(r),color(c)
   {
      cout<<"Κύκλος"<<endl;
   }
};

class cylinder:public circle_based
{
   float ypsos;
public:
   void set_y(float y){ypsos=y;}
   void display() {cout <<ypsos<<" "<<color<<endl;}
   cylinder(int r, float yps, string col): circle_based (r,col)
   {
      cout<<"Κύλινδρος"<<endl;
      ypsos=yps;
   }
};

class sphere:public circle_based
{
public:
   sphere(int r, string col):circle_based(r,col)
   {
      cout<<"Σφαίρα"<<endl;
```

Μέθοδος δόμησης της κλάσης με δύο παραμέτρους. Καταχωρίζει τις τιμές των παραμέτρων ως τιμή ακτίνας και χρώματος, αντίστοιχα. Εμφανίζει επίσης τη λέξη «Κύκλος».

Μέθοδος δόμησης της κλάσης με τρεις παραμέτρους. Δημιουργεί έναν κύκλο βάσης με ακτίνα **r** και χρώμα **col**. Καταχωρίζει το **yps** ως τιμή της μεταβλητής-μέλους ypsos. Εμφανίζει επίσης τη λέξη «Κύλινδρος».

Μέθοδος δόμησης της κλάσης με δύο παραμέτρους. Δημιουργεί τον κύκλο βάσης της σφαίρας με ακτίνα **r** και χρώμα **col**. Εμφανίζει επίσης τη λέξη «Σφαίρα».

```
   }
};

int main()
{
   cylinder solinas(10,5,"Κόκκινο");
   sphere mpala(10,"Μπλε");
   solinas.display();
   mpala.display();
   return 0;
}
```

```
Κύκλος
Κύλινδρος
Κύκλος
Σφαίρα
5 Κόκκινο
10 314.159
```

☞ Η πρόταση **cylinder** solinas(10,5,"Κόκκινο"); δημιουργεί ένα αντικείμενο κλάσης cylinder χρησιμοποιώντας τη μέθοδο δόμησης της κλάσης με τρεις παραμέτρους. Επομένως, δημιουργεί έναν κύκλο βάσης ακτίνας 10 με «Κόκκινο» χρώμα, και έναν κύλινδρο ύψους 5. Ας μην ξεχνάμε ότι κάθε αντικείμενο κλάσης cylinder περιέχει ένα αντικείμενο κλάσης circle το οποίο δημιουργείται πρώτο. Οι μέθοδοι δόμησης των δύο κλάσεων εμφανίζουν επίσης με τη σειρά τις λέξεις «Κύκλος» και «Κύλινδρος».

☞ Η πρόταση **sphere** mpala(10,"Μπλε"); δημιουργεί ένα αντικείμενο κλάσης sphere χρησιμοποιώντας τη μέθοδο δόμησης της κλάσης με δύο παραμέτρους. Επομένως, δημιουργεί έναν κύκλο ακτίνας 10 και χρώματος "Μπλε". Ας μην ξεχνάμε ότι κάθε αντικείμενο κλάσης sphere περιέχει ένα αντικείμενο κλάσης circle το οποίο δημιουργείται πρώτο. Οι μέθοδοι δόμησης των δύο κλάσεων εμφανίζουν επίσης με τη σειρά τις λέξεις «Κύκλος» και «Σφαίρα».

☞ Η πρόταση solinas.display(); εφαρμόζει τη μέθοδο display() της κλάσης cylinder στο αντικείμενο solinas και εμφανίζει το ύψος και το χρώμα του κυλίνδρου.

☞ Η πρόταση mpala.display(); εφαρμόζει τη μέθοδο display() της κλάσης circle_based στο αντικείμενο mpala (διότι η κλάση sphere δεν διαθέτει τέτοια μέθοδο που να την υποσκελίζει), και εμφανίζει την ακτίνα και το εμβαδό της βάσης του κύκλου της σφαίρας.

Σχολιάστε τι αποτέλεσμα θα είχαν οι ακόλουθες προτάσεις:

```
cylinder c1;
sphere s1;
```

☞ Και οι δύο προτάσεις είναι λάθος διότι προϋποθέτουν την ύπαρξη μεθόδων δόμησης χωρίς παραμέτρους. Καμία από τις δύο κλάσεις δεν διαθέτει τέτοια μέθοδο δόμησης.

16.11

✘	b1.mb1=3	*Η μεταβλητή-μέλος **mb1** είναι ιδιωτική. Τα αντικείμενα της κλάσης **b** δεν έχουν πρόσβαση.*
✘	b1.mb2=3	*Η μεταβλητή-μέλος **mb2** είναι προστατευμένη. Τα αντικείμενα της κλάσης **b** δεν έχουν πρόσβαση.*
✓	b1.b_func()	*Καλείται η μέθοδος **b_func()** της κλάσης **b**.*
✓	c1.c_func()	*Καλείται η μέθοδος **c_func()** της κλάσης **c**.*
✓	c1.b_func()	*Καλείται η μέθοδος **b_func()** της κλάσης **b**.*
✓	d1.d_func()	*Καλείται η μέθοδος **d_func()** της κλάσης **d**.*
✘	d1.b_func()	*Η κλάση **d** παράγεται από την **b** με ιδιωτική πρόσβαση. Τα δημόσια μέλη της **b** κληρονομούνται ως ιδιωτικά. Τα αντικείμενα της κλάσης **d** δεν έχουν πρόσβαση στα δημόσια μέλη της **b**.*
✓	c1.show()	*Καλείται η μέθοδος **show()** της κλάσης **c**.*
✓	c1.b::show()	*Καλείται η μέθοδος **show()** της κλάσης **b**.*
✓	c1.test()	*Επειδή η κλάση c δεν διαθέτει μέθοδος **test()**, καλείται η μέθοδος **test()** της βασικής κλάσης **b**.*
✘	d1.test()	*Όπως αναφέρθηκε και προηγούμενα, η κλάση **d** παράγεται από την **b** με ιδιωτική πρόσβαση. Τα αντικείμενα της κλάσης **d** δεν έχουν πρόσβαση στα δημόσια μέλη της **b**.*
✓	c2=c1	*Αναθέτει τα δεδομένα του αντικειμένου c1 στο c2.*
✘	d1=c1	*Τα αντικείμενα είναι διαφορετικού τύπου.*
✘	if (c1==c2) cout<<"NAI"	*Ο τελεστής == δεν μπορεί να εφαρμοστεί σε αντικείμενα της κλάσης c διότι δεν έχει υπερφορτωθεί για αυτή την κλάση.*

16.12

```
.....
.....
class c_box:private box
{
protected:
   string color;
public:
   c_box():box(),color("Λευκό"){}
   c_box(float a,float b,float c,float v,string col):box(a,b,c,v),color(col){}
   void set_color(string col){color=col;}
   string get_color(){return color;}
   void show()
   {
      rec3D::show();
      cout<<"Χρώμα:"<<color<<endl;
   }
};
int main()
{
   c_box cb1(5,10,20,25.5,"Κόκκινο");
   cb1.show();
   return 0;
}
```

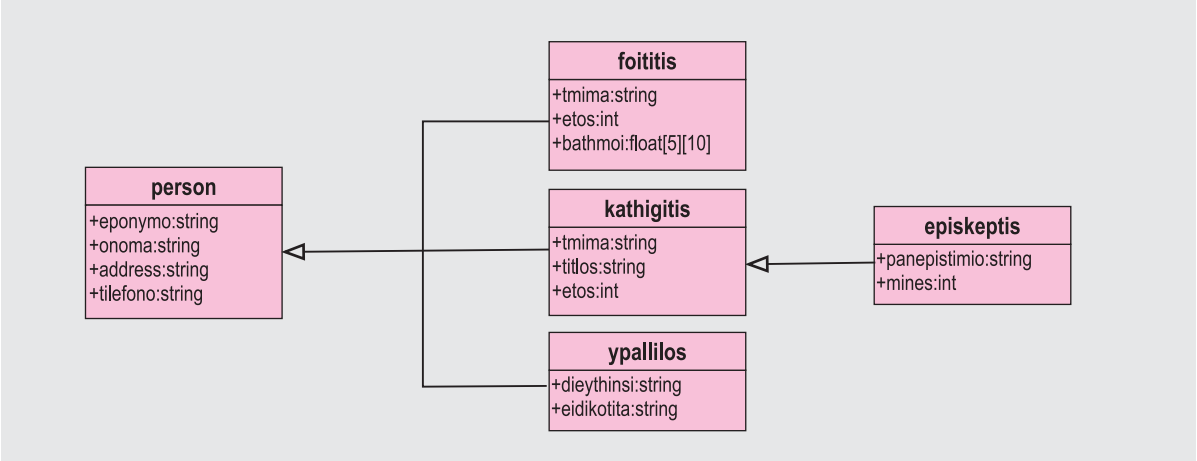
Η πρώτη μέθοδος δόμησης χωρίς παραμέτρους χρησιμοποιεί τη μέθοδο δόμησης της κλάσης **box**, επίσης χωρίς παραμέτρους, για τη δημιουργία του υπο-αντικειμένου **box** με μηδενικές διαστάσεις. Η μέθοδος δόμησης με πέντε παραμέτρους χρησιμοποιεί τη μέθοδο δόμησης της κλάσης **box** με τέσσερις παραμέτρους για τη δημιουργία του υπο-αντικειμένου **box** με τις συγκεκριμένες διαστάσεις.

Η μέθοδος **show()** της κλάσης **c_box** καλεί τη μέθοδο **show()** της κλάσης **rec3D** για να εμφανίσει τις διαστάσεις.

Δημιουργεί ένα αντικείμενο **cb1** κλάσης **c_box** χρησιμοποιώντας τη μέθοδο δόμησης της κλάσης με πέντε παραμέτρους.

Εμφανίζει τα στοιχεία του αντικειμένου **cb1** χρησιμοποιώντας τη μέθοδο **show()** της κλάσης.

16.13



16.14

```
#include <iostream>
#include <string>
using namespace std;

class account
{
   float poso;
public:
   string onoma;
   account(float p):poso(p){}
   account():poso(0){}
   float katathesi(float p){poso=poso+p;}
   float analipsi(float p){if (p<=ypoloipo()) poso=poso-p;}
   float ypoloipo(){return poso;}
};

class pistotiki:public account
{
   float orio;
public:
   pistotiki(float o):account(1000),orio(o){cout<<orio<<endl;}
   pistotiki():account(),orio(100){}
   float analipsi(float p)
   {
      if (p<=orio)
         account::analipsi(p);
      else
         account::analipsi(orio);
   }
};

int main()
{
   pistotiki p1(200),p2;
   cout << p1.ypoloipo()<<endl;
   p1.katathesi(1000);
   p1.katathesi(500);
   cout << p1.ypoloipo()<<endl;
   p1.analipsi(100);
   p1.analipsi(300);
   cout << p1.ypoloipo()<<endl;
   p2.katathesi(1000);
   p2.analipsi(400);
   cout << p2.ypoloipo()<<endl;
   return 0;
}
```

Η κλάση διαθέτει δύο μεθόδους δόμησης. Μία με μία παράμετρο η οποία καταχωρίζει την τιμή της παραμέτρου της ως ποσό του λογαριασμού, και μία με καμία παράμετρο η οποία ορίζει στο ποσό μηδενική τιμή.

Η μέθοδος **katathesi()** προσθέτει την τιμή της παραμέτρου της στο ποσό του λογαριασμού. Η μέθοδος **analipsi()** αφαιρεί την τιμή της παραμέτρου της από το ποσό του λογαριασμού αλλά μόνο στην περίπτωση που το υπόλοιπο επαρκεί!

Η κλάση **pistotiki** είναι παράγωγη της κλάσης **account**

Η μέθοδος δόμησης με μία παράμετρο δημιουργεί λογαριασμό με ποσό 1000 και καταχωρίζει την τιμή της παραμέτρου της ως όριο της πιστωτικής κάρτας. Η μέθοδος δόμησης χωρίς παράμετρο δημιουργεί λογαριασμό με ποσό 0 και θέτει το 100 ως όριο της πιστωτικής κάρτας.

Η μέθοδος **analipsi** της κλάσης **pistotiki** κάνει ανάληψη του ποσού χρησιμοποιώντας τη μέθοδο **analipsi** της κλάσης **account**. Αν όμως έχουμε υπερβεί το όριο, κάνει ανάληψη του ορίου και όχι όλου του ποσού.

```
200
1000
2500
2200
900
```

☞ Η πρόταση **pistotiki** p1(200),p2; δημιουργεί δύο αντικείμενα της κλάσης pistotiki χρησιμοποιώντας την πρώτη μέθοδο δόμησης για το p1 και τη δεύτερη για το p2. Επομένως, για το p1 υπάρχει αρχικό ποσό 100 και όριο 200 ενώ για το p2 υπάρχει αρχικό ποσό 0 και όριο 100.

☞ Προσοχή: Στην πρόταση p1.analipsi(300); θα γίνει ανάληψη μόνο 200 διότι τα 300 είναι πάνω από το όριο της πιστωτικής p1, και στην πρόταση p2.analipsi(400); θα γίνει ανάληψη μόνο 100 διότι τα 400 είναι πάνω από το όριο της πιστωτικής p2!

16.15

```
#include <iostream>
#include <string>
using namespace std;

class account
{
   float poso;
public:
   string onoma;
   account(float p):poso(p){}
   account():poso(0){}
   float katathesi(float p){poso=poso+p;}
   float analipsi(float p){if (p<=ypoloipo()) poso=poso-p;}
   float ypoloipo(){return poso;}
};
class misthodosias:protected account
{
   float foros;
public:
   misthodosias() {foros=0;}
   float katathesi(float p)
   {
      float temp;
      temp=p*20/100;
      foros=foros+temp;
      account::katathesi(p-temp);
   }
   void show()
   {
      cout<<"Διαθέσιμο ποσό:"<<ypoloipo()<<endl;
      cout<<"Συνολικός φόρος:"<<foros<<endl;
   }
};

int main()
{
   misthodosias m1;
   m1.katathesi(1000);
   m1.show();
   m1.katathesi(9000);
   m1.show();
   m1.katathesi(50);
   m1.show();
   return 0;
}
```

Μέθοδος δόμησης της κλάσης **misthodosias** χωρίς παράμετρο η οποία καταχωρίζει μηδενική τιμή στον φόρο.

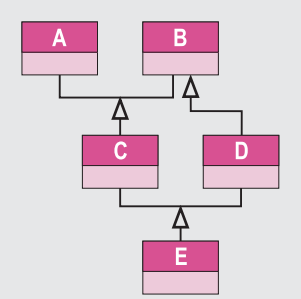
Η μέθοδος **katathesi()** παρακρατεί το 20% του ποσού το οποίο το προσθέτει στη μεταβλητή-μέλος και το υπόλοιπο το καταθέτει στον λογαριασμό χρησιμοποιώντας τη μέθοδο **katathesi()** της κλάσης **account**.

Η μέθοδος **show()** εμφανίζει το διαθέσιμο ποσό και το συνολικό ποσό του φόρου.

```
Διαθέσιμο ποσό:800
Συνολικός φόρος:200
Διαθέσιμο ποσό:8000
Συνολικός φόρος:2000
Διαθέσιμο ποσό:8040
Συνολικός φόρος:2010
```

Ασκήσεις Κεφαλαίου 17

17.1




```
#include <iostream>
using namespace std;

class sxima
{
   string color;
public:
   void set_col(string col) {color=col;}
   void show() {cout<<"Σχήμα χρώματος "<<color<<endl;}
};

class kyklos: public sxima
{
   float aktina;
public:
   void set_r(float r) {aktina=r;}
   void show() {cout<<"Κύκλος ακτίνας "<<aktina<<endl;}
};

class tetragono: public sxima
{
   float plevra;
public:
   void set_a(float a) {plevra=a;}
   void show() {cout<<"Τετράγωνο πλευράς "<<plevra<<endl;}
};

int main()
{
   kyklos k1;
   tetragono t1;
   sxima *ptr1 = &t1;
   t1.set_a(10);
   t1.set_col("Κόκκινο");
   k1.set_col("Μπλε");
   k1.set_r(5);
   ptr1->show();
   ptr1=&k1;
   ptr1->show();
   return 0;
}
```

```
Σχήμα χρώματος Κόκκινο
Σχήμα χρώματος Μπλε
```

Καλείται η μέθοδος **set_col()** της βασικής κλάσης **sxima**.

Καλείται η μέθοδος **show()** της βασικής κλάσης **sxima**, παρά το γεγονός ότι ο δείκτης **ptr1**, και στις δύο περιπτώσεις, δείχνει σε αντικείμενα παράγωγων κλάσεων. Αυτό συμβαίνει διότι ο δείκτης **ptr1** ανήκει στη βασική κλάση **sxima**.

17.2

*Η μόνη αλλαγή που πρέπει να γίνει είναι να δηλωθεί η μέθοδος show() της βασικής κλάσης sxima ως εικονική με το πρόθεμα **virtual**:*

```
class sxima
{
   char color[10];
public:
   void set_col(string col) {color=col;}
   virtual void show() {cout<<"Σχήμα χρώματος "<<color<<endl;}
};
```

17.3

```
#include <iostream>
using namespace std;

class A
{
public:
   void show_a() {cout<<"Κλάση-A"<<endl;}
};

class B
{
public:
   void show_b() {cout<<"Κλάση-B"<<endl;}
};

class C:public A, virtual public B
{
public:
   void show_c() {cout<<"Κλάση-C"<<endl;}
};

class D:virtual public B
{
public:
   void show_d() {cout<<"Κλάση-D"<<endl;}
};

class E:public C, public D
{
public:
   void show_e() {cout<<"Κλάση-E"<<endl;}
};

int main()
{
   E obj;
   obj.show_a();
   obj.show_b();
   return 0;
}
```

Η κλάση **C** παράγεται από τις κλάσεις **A** και **B**

Καλείται η μέθοδος **show_a()** της κλάσης **A**.

Καλείται η μέθοδος **show_b()** της κλάσης **B**.

☞ Για να περιέχουν τα αντικείμενα της κλάσης E ένα μόνο στιγμιότυπο της κλάσης B, στον ορισμό των παράγωγων κλάσεων C και D πρέπει να δηλωθεί ως **εικονική** (virtual) η βασική κλάση B .

☞ Αν η κλάση B δεν είχε δηλωθεί ως εικονική, στον ορισμό των κλάσεων C και D το αντικείμενο obj θα περιείχε δύο στιγμιότυπα της κλάσης B. Σε αυτή τη περίπτωση, η πρόταση obj.show_b() θα δημιουργούσε ασάφεια διότι ο μεταγλωττιστής δεν θα γνώριζε ποια από τις δύο μεθόδους show_b() των δύο υποαντικειμένων της κλάσης B να καλέσει.

☞ Στην περίπτωση που δημιουργηθεί ένα αντικείμενο της παράγωγης κλάσης C, η σειρά δημιουργίας των στιγμιοτύπων των κλάσεων είναι A→B→C.

17.4

Στην περίπτωση που το στιγμιότυπο της κλάσης B υπάρχει δύο φορές.

A→B→C→B→D→E

Στην περίπτωση που το στιγμιότυπο της κλάσης B υπάρχει μία μόνο φορά.

B→A→C→D→E

17.5

- ☑ Η κλάση A παράγεται από τη B. Ένας δείκτης της κλάσης B μπορεί να δείχνει σε αντικείμενα της κλάσης A.
- ❑ Η κλάση A παράγεται από τη B. Ένας δείκτης της κλάσης A μπορεί να δείχνει σε αντικείμενα της κλάσης B.
- ☑ Μια εικονική μέθοδος δηλώνεται στη βασική κλάση.
- ☑ Μια γνήσια εικονική μέθοδος δηλώνεται στη βασική κλάση αλλά συνήθως δεν ορίζεται σε αυτήν. Σε κάθε περίπτωση, θα πρέπει να ορίζεται στις παράγωγες κλάσεις.
- ❑ Όταν η κλάση A παράγεται από τη B, και η κλάση C από την A, τότε έχουμε πολλαπλή κληρονομικότητα.
- ☑ Όταν η κλάση A παράγεται από τη B και την C, τότε έχουμε πολλαπλή κληρονομικότητα.
- ❑ Αφηρημένη κλάση λέγεται η κλάση που περιέχει έστω και μία εικονική μέθοδο.
- ❑ Κατά τη δημιουργία αντικειμένων της κλάσης που ορίζεται με την πρόταση **class** A: **public** B, **public** C, τα αντικείμενα δημιουργούνται με τη σειρά A→B→C.

17.6

Όταν μία μέθοδος δηλωθεί ως *εικονική (virtual)* στη βασική κλάση, μπορεί να οριστεί πάλι σε μία ή περισσότερες παράγωγες κλάσεις. Η κλήση μιας τέτοιας μεθόδου γίνεται μέσω *δεικτών* της βασικής κλάσης οι οποίοι όμως δείχνουν σε αντικείμενα των *παράγωγων* κλάσεων. Τέτοιες μέθοδοι μπορεί να μην διαθέτουν καθόλου σώμα στη βασική κλάση και τότε λέγονται *γνήσιες εικονικές (pure virtual)* μέθοδοι. Μια τέτοια μέθοδος ξεχωρίζει από τις υπόλοιπες επειδή περιέχει ένα *=0* στο τέλος της δήλωσής της. Μια κλάση που περιέχει έστω και μια τέτοια μέθοδο λέγεται *αφηρημένη (abstract)* κλάση. Από αυτή την κλάση δεν μπορούν να δημιουργηθούν *αντικείμενα* παρά μόνο άλλες *παράγωγες* κλάσεις.

17.7

Πολλαπλή κληρονομικότητα έχουμε όταν μία κλάση παράγεται από *περισσότερες από μία βασικές κλάσεις*. Στην περίπτωση αυτή, κάθε αντικείμενο της παράγωγης κλάσης διαθέτει από ένα *στιγμιότυπο* των βασικών κλάσεων. Υπάρχουν περιπτώσεις όπου μια *παράγωγη* κλάση περιέχει περισσότερα από ένα στιγμιότυπα μιας βασικής κλάσης. Αυτό λύνεται με τη δήλωση της συγκεκριμένης βασικής κλάσης ως *εικονικής (virtual)*.

17.8

```
#include <iostream>
using namespace std;

class A
{
public:
   virtual void display()=0;
};
```

Η μέθοδος **display()** δηλώνεται ως γνήσια εικονική.

```
class B: public A
{
public:
   void display() {cout<<"class B"<<endl;}
};
class C: public A
{
public:
   void show() {cout<<"class C"<<endl;}
};
int main()
{
   C c1;
   B b1;
   A a1;
   b1.display();
   c1.show();
   return 0;
}
```

Στη βασική κλάση **A**, η μέθοδος **display()** δηλώνεται ως γνήσια εικονική. Αυτό σημαίνει ότι κάθε παράγωγη κλάση θα πρέπει να διαθέτει έναν ορισμό αυτής της μεθόδου. Όμως η παράγωγη κλάση **C** δεν διαθέτει κάποια έκδοση της μεθόδου **display()**!

Η βασική κλάση **A**, από τη στιγμή που διαθέτει μια γνήσια εικονική μέθοδο, είναι μια αφηρημένη κλάση. Δεν είναι δυνατόν να δημιουργηθούν αντικείμενα μιας αφηρημένης κλάσης, οπότε η πρόταση αυτή είναι λανθασμένη.

17.9

```
#include <iostream>
using namespace std;
class test
{
public:
   virtual void display()=0;
};
class A: public test
{
public:
   void display() {cout<<"class A"<<endl;}
};
class B: public test
{
public:
   void display() {cout<<"class B"<<endl;}
};
int main()
{
   A a1;
   B b1;
   a1.display();
   b1.display();
   return 0;
}
```

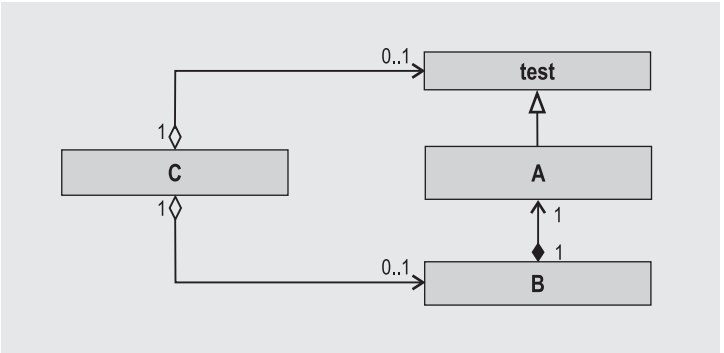
Η μέθοδος **display()** δηλώνεται ως γνήσια εικονική. Η κλάση **test**, από τη στιγμή που διαθέτει μια γνήσια εικονική μέθοδο, είναι μια αφηρημένη κλάση. Η μέθοδος **display()** πρέπει να οριστεί σε κάθε παράγωγη κλάση της κλάσης **test**.

Στις κλάσεις **A** και **B** ορίζεται από μια έκδοση της γνήσιας εικονικής μεθόδου **display()** της βασικής κλάσης **test**.

Εφαρμόζονται οι μέθοδοι **display()** των παράγωγων κλάσεων **A** και **B** αντίστοιχα.

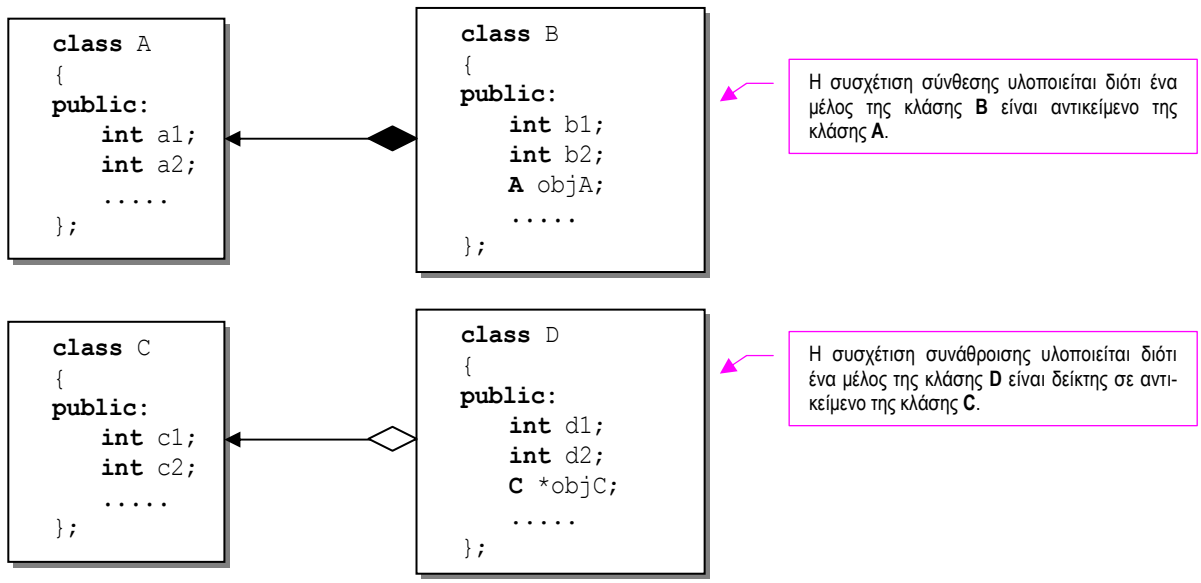
17.10

```
class test
{
public:
   int t1;
};
class A:public test
{
public:
   int a1;
   int a2;
};
```



```
class B
{
public:
   int b1;
   int b2;
   A ob1;
   void display() {cout<<ob1.a1<<endl<<ob1.a2<<endl;}
};
class C
{
public:
   test *p1;
   B *p2;
   void display() {cout<<p1->t1<<endl; p2->display();}
};
```

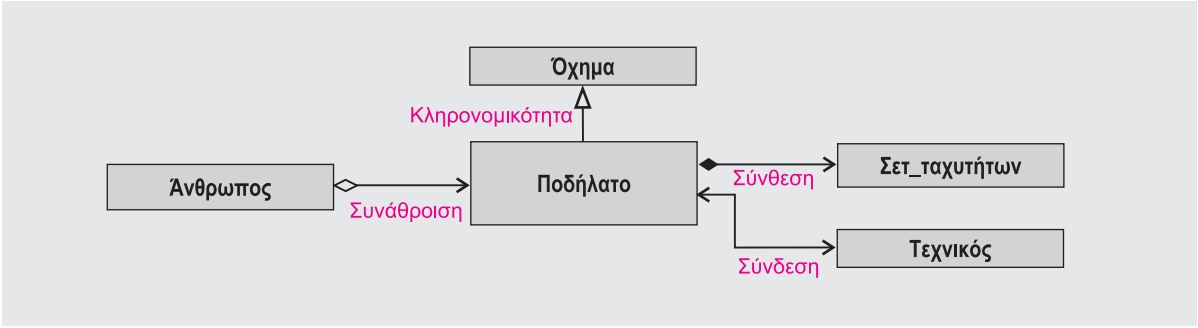
17.11



17.12

Αεροσκάφος - Μηχανή:	*Καμία*
Μηχανή - Τουρμπίνα:	*Κληρονομικότητας*
Αεροσκάφος - Τουρμπίνα:	*Σύνθεσης*
Αεροσκάφος - Επιβάτης:	*Συνάθροισης*
Επιβάτης - Άνθρωπος:	*Κληρονομικότητας*
Πιλότος - Αεροσκάφος:	*Συνάθροισης*
Πιλότος - Άνθρωπος:	*Κληρονομικότητας*
Αεροσκάφος - Δρομολόγιο:	*Σύνδεσης*
Πιλότος - Επιβάτης:	*Καμία*

17.13



17.14

```
class A
{
public:
   int a1;
   virtual void test()=0;
};
class C:
{
public:
   int c1;
   void show() {cout<<"class C"<<endl;}
};
class B: public A
{
public:
   int b1;
   C objC[2];
   void test() {cout<<"class B"<<endl;}
};

class D:
{
public:
   int d1;
   B *objB[5];
   void show() {cout<<"class D"<<endl;}
};
```

Από το διάγραμμα κλάσεων παρατηρούμε ότι η κλάση **A** έχει πλάγια γραφή, που σημαίνει ότι είναι μια αφηρημένη κλάση και πρέπει να περιέχει τουλάχιστον μία γνήσια εικονική μέθοδο, την **test()**.

Από το διάγραμμα κλάσεων παρατηρούμε ότι η κλάση **B** πρέπει να παράγεται από την **A**.

Από τη συσχέτιση σύνθεσης του διαγράμματος προκύπτει ότι η κλάση **B** πρέπει να περιέχει δύο μέλη της κλάσης **C**.
Η κλάση **B** πρέπει επίσης να περιέχει μια έκδοση της γνήσιας εικονικής μεθόδου **test()** η οποία δηλώνεται στη βασική κλάση **A**.

Η κλάση **D** πρέπει να έχει τη δυνατότητα αποθήκευσης μέχρι πέντε δεικτών σε αντικείμενα της κλάσης **B**. Αυτό υλοποιεί τη συσχέτιση συνάθροισης μεταξύ των κλάσεων **D** και **B** του διαγράμματος.

17.15

☞ Η συσχέτιση των κλάσεων toyristas και polis γίνεται μέσω της κλάσης σύνδεσης episkepsi.

☞ Κάθε αντικείμενο της κλάσης σύνδεσης episkepsi περιέχει έναν δείκτη σε ένα αντικείμενο της κλάσης toyristas και έναν δείκτη σε ένα αντικείμενο της κλάσης polis. Επίσης, η κλάση σύνδεσης episkepsi περιέχει δύο μέλη κλάσης imerominia.

☞ Στο πρόγραμμα που ακολουθεί οι κλάσεις διαθέτουν μόνο τις απαραίτητες μεταβλητές-μέλη, μεθόδους δόμησης, καθώς και κάποιες άλλες μεθόδους για καθαρά χρηστικούς και εκπαιδευτικούς λόγους.

```
#include <iostream>
using namespace std;

class toyristas
{
public:
   string onoma;
   toyristas(string o) {onoma=o;}
};

class polis
{
public:
   string onomasia;
   polis(string o) {onomasia=o;}
};

class imerominia
{
   int imera;
   int minas;
   int etos;
public:
   imerominia(int i,int m, int e) {imera=i;minas=m;etos=e;}
   imerominia() {imera=1;minas=1;etos=2000;}
   void show() {cout<<imera<<"/"<<minas<<"/"<<etos<<endl;}
};

class episkepsi
{
   toyristas *toyr;
   polis *pol;
   imerominia apo;
   imerominia eos;
public:
   episkepsi(toyristas *t, polis *p, imerominia a, imerominia e)
   {toyr=t;pol=p;apo=a;eos=e;}
   void display()
   {
      cout<<toyr->onoma<<" Επίσκεψη στην πόλη: "<<pol->onomasia<<endl;
      cout<<"Από:";
      apo.show();
      cout<<"Εως:";
      eos.show();
   }
};

int main()
{
   toyristas t1("Νίκος"),t2("Μάκης"),t3("Μαρία");
   polis p1("Αθήνα"),p2("Μυτιλήνη"),p3("Λονδίνο"),p4("Παρίσι");
   imerominia i1(10,11,2015),i2(15,11,2015),i3(20,8,2015),i4(30,8,2015);
   episkepsi e1(&t1,&p4,i1,i2),e2(&t3,&p2,i3,i4);
```

Μέθοδος δόμησης της κλάσης **toyristas**.

Μέθοδος δόμησης της κλάσης **polis**.

Μέθοδοι δόμησης της κλάσης **imerominia**.

Κάθε αντικείμενο της κλάσης σύνδεσης **episkepsi** περιέχει έναν δείκτη σε ένα αντικείμενο της κλάσης **toyristas** και έναν δείκτη σε ένα αντικείμενο της κλάσης **polis**. Επίσης η κλάση περιέχει δύο μέλη κλάσης **imerominia**.

Μέθοδος δόμησης της κλάσης **episkepsi**.

Εμφάνιση των στοιχείων της επίσκεψης.

```
    e1.display();
    e2.display();
    return 0;
}
```

- Στο πρόγραμμα αρχικά δημιουργούνται τρία αντικείμενα-τουρίστες και τέσσερα αντικείμενα-πόλεις με αντίστοιχα ονόματα.
- Έπειτα δημιουργούνται τέσσερα αντικείμενα-ημερομηνίες με συγκεριμένες αρχικές τιμές.
- Τέλος, δημιουργούνται δύο αντικείμενα-επισκέψεις των οποίων οι αρχικές τιμές προσδιορίζουν τον τουρίστα, την πόλη, και τις ημερομηνίες άφιξης και αναχώρισης.
- Εμφανίζονται τα στοιχεία των επισκέψεων!

```
Νίκος Επίσκεψη στην πόλη: Παρίσι
Από:10/11/2015
Έως:15/11/2015
Μαρία Επίσκεψη στην πόλη: Μυτιλήνη
Από:20/8/2015
Έως:30/8/2015
```

Ασκήσεις Κεφαλαίου 18

18.1

Η επικοινωνία μεταξύ του προγράμματος και των περιφερειακών συσκευών του Η/Υ γίνεται μέσω αντικειμένων *ρευμάτων*. Κάποια από αυτά είναι ήδη συνδεδεμένα με συγκεκριμένες περιφερειακές συσκευές και λέγονται *προκαθορισμένα ρεύματα*. Τα cin και cout δεν είναι τίποτε άλλο παρά *αντικείμενα* ρευμάτων. Η C++ υποστηρίζει δύο κατηγορίες ρευμάτων τα *δυαδικά* ρεύματα και τα ρεύματα *χαρακτήρων* τα οποία αποκαλούνται αντίστοιχα χαμηλού και υψηλού επιπέδου.

18.2

Η βασικότερη κλάση από την οποία παράγονται όλες οι κλάσεις διαχείρισης ρευμάτων είναι η κλάση *ios_base*. Το αντικείμενο cout ανήκει στην κλάση *ostream* ενώ το cin ανήκει στην κλάση *istream*. Ένα αρχείο εισόδου πρέπει να το συνδέσουμε με ένα αντικείμενο ρεύματος κλάσης *ifstream* ενώ ένα αρχείο εξόδου με ένα αντικείμενο ρεύματος κλάσης *ofstream*. Με ένα αντικείμενο ρεύματος κλάσης fstream θα συνδέσουμε ένα αρχείο *εισόδου/εξόδου*.

18.3

- ☑ Ο χειρισμός ενός αρχείου στη C++ γίνεται μέσω των αντικειμένων ρευμάτων.
- ❑ Για τη C++, ένα αρχείο είναι μια σειρά από bit.
- ☑ Όταν ζητήσουμε να ανοίξουμε ένα αρχείο για εγγραφή και το αρχείο δεν υπάρχει, τότε δημιουργείται.
- ❑ Δεν μπορούμε να έχουμε περισσότερα από δύο αρχεία ταυτόχρονα ανοιχτά.
- ☑ Η ενδιάμεση μνήμη (buffer) επιταχύνει τις διαδικασίες ανάγνωσης/εγγραφής.

18.4

```
cout << setw(6) << setfill('@') << 123 << endl;
cout << setw(6) << hex << uppercase << 255 << endl;
cout << setw(6) << fixed << setprecision(3) << 1.23456 << endl;
cout << setw(8) << showpos << dec <<setfill('#') << 24 <<endl;
```

```
@@@123
@@@@FF
@1.235
#####+24
```

☞ Να θυμόμαστε ότι κάποιοι χειριστές ενεργοποιούν λειτουργίες οι οποίες ισχύουν όχι μόνο για το επόμενο στοιχείο που εισάγεται σε ένα ρεύμα εξόδου αλλά μέχρι να απενεργοποιηθούν ή να αλλάξουν.

18.5

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
   cout.width(6);      // Ορίζει ως πλάτος εμφάνισης τους 6 χαρακτήρες
   cout.fill('@');     // Ορίζει ως χαρακτήρα πλήρωσης κενών θέσων τον χαρακτήρα @
   cout<<123<<endl;    // Εμφανίζει το 123 σύμφωνα με τα παραπάνω
   cout.setf(ios::hex,ios::basefield);     // Ορίζει εμφάνιση τιμών σε δεκαεξαδική μορφή
   cout.setf(ios::uppercase);              // Ορίζει εμφάνιση τιμών με κεφαλαίους χαρακτήρες
   cout.width(6);                          // Ορίζει ως πλάτος εμφάνισης τους 6 χαρακτήρες
   cout << 255 << endl;                    // Εμφανίζει το 255 σύμφωνα με τα παραπάνω
   cout.width(6);                          // Ορίζει ως πλάτος εμφάνισης τους 6 χαρακτήρες
   cout.setf(ios::dec,ios::basefield);     // Ορίζει εμφάνιση τιμών σε δεκαδική μορφή
   cout.setf(ios::fixed);                  // Ορίζει εμφάνιση τιμών σε κανονική (όχι εκθετική) μορφή
   cout.precision(3);                      // Ορίζει εμφάνιση με ακρίβεια τριών δεκαδικών ψηφίων
   cout << 1.23456 << endl;                // Εμφανίζει το 1.23456 σύμφωνα με τα παραπάνω
```

```
   cout.width(8);                    // Ορίζει ως πλάτος εμφάνισης τους 8 χαρακτήρες
   cout.fill('#');                   // Ορίζει ως χαρακτήρα πλήρωσης κενών θέσων τον χαρακτήρα #
   cout.setf(ios::showpos);          // Ορίζει εμφάνιση του προσήμου και στις θετικές τιμές
   cout << 24 <<endl;                // Εμφανίζει το 24 σύμφωνα με τα παραπάνω
   return 0;
}
```

18.6

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   cout.precision(1);
   cout.width(5);
   cout<<fixed<<12.456<<endl;
   cout.fill('=');
   cout.width(7);
   cout.precision(2);
   cout<<3.4<<endl;
   cout.width(5);
   cout<<0.33<<endl;
   return 0;
}
```

```
 12.5
==3.40
=0.33
```

18.7

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
   ofstream myfile;
   int i;
   myfile.open("arithmoi");
   if (!myfile)
   {
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
      exit(2);
   }
   for(i=1;i<=100;i++) myfile<<i<<endl;
   myfile.close();
   return 0;
}
```

Ανοίγει το αρχείο **arithmoi** και ελέγχει αν υπήρξε πρόβλημα στο άνοιγμα του αρχείου.

Εγγράφει στο αρχείο τους αριθμούς από το 1 μέχρι το 100, Οι αριθμοί χωρίζονται με χαρακτήρα αλλαγής γραμμής.

Κλείνει το αρχείο **arithmoi**.

18.8

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
   ifstream myfile;
   int i,arist=0,apotyx=0,synolo=0;
   float bathmos,par,pap;
   myfile.open("bathmoi.txt");
   if(!myfile)
   {
```

```
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
      exit(2);
   }
   while (myfile>>bathmos)
   {
      if (bathmos>=18) arist++;
      if (bathmos<10) apotyx++;
      synolo++;
   }
   myfile.close();
   par=100.0*arist/synolo;
   pap=100.0*apotyx/synolo;
   cout <<"Αριστούχοι "<<arist<<" ποσοστό "
      <<fixed<<setprecision(1)<<par<<"%"<<endl;
   cout <<"Αποτυχόντες "<<apotyx<<" ποσοστό "
      <<fixed<<setprecision(1)<<pap<<"%"<<endl;
   return 0;
}
```

Η μεταβλητή **arist** χρησιμοποιείται για την καταμέτρηση του πλήθους των αριστούχων, η **apotyx** για το πλήθος των αποτυχόντων και η **synolo** για το συνολικό πλήθος των βαθμών στο αρχείο.

Υπολογισμός των ποσοστών

18.9

```
bool append(int pin[])
{
   ofstream myfile;
   int i;
   myfile.open("output",ios::app);
   if(!myfile)
   {
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
      return false;
   }
   myfile.close();
   for(i=0;i<100;i++)
      myfile<<pin[i]<<endl;
   myfile.close();
   return true;
}
```

Ανοίγει το αρχείο **output** για έξοδο και προσθήκη στο τέλος του αρχείου.

Γράφει στο αρχείο τους αριθμούς του πίνακα. Οι αριθμοί χωρίζονται με χαρακτήρα αλλαγής γραμμής.

Κλείσιμο του αρχείου.

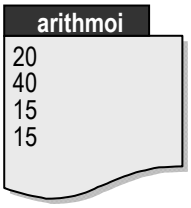
☞ Η παραπάνω συνάρτηση επιστρέφει τιμή **false** αν υπήρξε πρόβλημα στο άνοιγμα του αρχείου και τιμή **true** αν δεν υπήρξε κανένα πρόβλημα.

18.10

```
#include <iostream>
#include <iomanip>
#include <fstream>

using namespace std;

int main()
{
   float ar,s=0,m;
   int n=0;
   ifstream myfile;
   myfile.open("arithmoi");
   while (myfile>>ar)
   {
      s=s+ar;
      n++;
   }
   m=s/n;
   myfile.close();
   cout<<m<<endl;
```



Διαβάζει από το αρχείο τον επόμενο αριθμό και τον καταχωρίζει στη μεταβλητή **ar**.

Στη μεταβλητή **s** καταχωρίζεται το άθροισμα όλων των αριθμών, ενώ στη μεταβλητή **n** το πλήθος τους.

```
   return 0;
}
```

Τι θα εμφανίσει στην οθόνη στην περίπτωση που τα περιεχόμενα του αρχείου arithmoi είναι όπως εμφανίζονται στην εικόνα δεξιά;

☞ Το πρόγραμμα υπολογίζει τον μέσο όρο των αριθμών που υπάρχουν στο αρχείο. Επομένως στη συγκεκριμένη περίπτωση θα εμφανίσει τον αριθμό 20 (100/5).

18.11

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
int main()
{
   float ar,s=0,m,maximum,minimum;
   int n=0;
   string name;
   ifstream myfile;
   cout << "Δώσε όνομα αρχείου:";
   cin>>name;
   myfile.open(name);
   if(!myfile)
   {
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου!"<<endl;
      return 1;
   }
   myfile>>ar;
   maximum=minimum=ar;
   while (myfile>>ar)
   {
      if (ar>maximum)
         maximum=ar;
      else if (ar<minimum)
         maximum=ar;
   }
   myfile.close();
   cout<<"Μέγιστος:"<<maximum<<endl;
   cout<<"Ελάχιστος:"<<minimum<<endl;
   return 0;
}
```

Ανοίγει το αρχείο το όνομα του οποίου πληκτρολόγησε ο χρήστης (αρκεί να υπάρχει).

Διαβάζει τον πρώτο αριθμό του αρχείου και τον καταχωρίζει ταυτόχρονα ως μέγιστο και ελάχιστο αριθμό.

Διαβάζει τον επόμενο αριθμό και τον καταχωρίζει στη μεταβλητή **ar**.

Στην περίπτωση που ο αριθμός είναι μεγαλύτερος από τον τρέχοντα μεγαλύτερο τον καταχωρίζει στη θέση **maximum** και στην περίπτωση που είναι μικρότερος από τον τρέχοντα μικρότερο τον καταχωρίζει στη θέση **minimum**. Οι μεταβλητές **maximum** και **minimum** περιέχουν στο τέλος τον μεγαλύτερο και τον μικρότερο αριθμό του αρχείου αντίστοιχα.

18.12

```
#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
using namespace std;

int main()
{
   float b1,b2,b3;
   string onoma;
   ifstream myfile;
   myfile.open("onomata.txt");
   while (myfile)
   {
      myfile>>onoma>>b1>>b2>>b3;
      if(myfile)
      {
```

Στις μεταβλητές **b1** ,**b2** και **b3** θα καταχωριστούν οι τρεις βαθμοί κάθε μαθητή, ενώ στο αντικείμενο **onoma** το όνομά του.

Εξάγει από το ρεύμα εισόδου ένα σύνολο χαρακτήρων και τρεις αριθμούς. Καταχωρίζονται με τη σειρά στο αντικείμενο **onoma** και στις μεταβλητές **b1**, **b2** και **b3**.

```
            cout<<setw(15)<<left<<onoma<<fixed<<setprecision(1)<<(b1+b2+b3)/3<<endl;
         }
      }
      myfile.close();
      return 0;
}
```

Σε ένα εύρος πεδίου 15 θέσεων εμφανίζει το όνομα με αριστερή στοίχιση. Εμφανίζει επίσης τον μέσο όρο των βαθμών με ένα δεκαδικό.

18.13

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

int main()
{
   int ar=0,ep=0;
   string onom,bathmoi,epityxontes,apotyxontes;
   float b;
   ifstream arxeio_bath;
   ofstream arxeio_ap,arxeio_ep;
   cout<<"Δώσε όνομα αρχείου:";
   cin>>bathmoi;
   epityxontes=bathmoi+".ep";
   apotyxontes=bathmoi+".ap";
   arxeio_bath.open(bathmoi);
   arxeio_ap.open(apotyxontes);
   arxeio_ep.open(epityxontes);
   if (!arxeio_bath)
   {
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου εισόδου"<<endl;
      return 1;
   }

   while (!arxeio_bath.eof())
   {
      arxeio_bath>>onom>>b;
      if (b>=5)
      {
         arxeio_ep<<onom<<endl;
         ep++;
      }
      else
         arxeio_ap<<onom<<endl;
      ar++;
   }
   arxeio_bath.close();
   arxeio_ap.close();
   arxeio_ep.close();
   cout << "Πλήθος φοιτητών: "<<ar<<endl;
   cout << "Ποσοστό επιτυχόντων: "<<100*(float)ep/ar<<"%"<<endl;
   return 0;
}
```

Ανοίγει τα τρία αρχεία με βάση το όνομα που πληκτρολόγησε ο χρήστης (αρκεί να υπάρχει το αρχείο).

Η επαναληπτική διαδικασία σταματάει μόλις φτάσουμε στο τέλος του αρχείου.

Διαβάζει ένα όνομα και έναν βαθμό.

Στην περίπτωση που ο βαθμός είναι >=5 γράφει το όνομα στο αρχείο των επιτυχόντων (arxeio_ep), διαφορετικά το γράφει στο αρχείο των αποτυχόντων (arxeio_ap). Μετράει επίσης το πλήθος των επιτυχόντων (ep) καθώς και το συνολικό πλήθος (ar).

Κλείνει και τα τρία αρχεία.

18.14

- ☑ Η τυχαία προσπέλαση σε ένα αρχείο επιτυγχάνεται με τις συναρτήσεις-μέλη seekp() και seekg().
- ☑ Οι συναρτήσεις-μέλη write() και read() γράφουν και διαβάζουν στοιχεία σε δυαδική μορφή.
- ☑ Η μόνη διαφορά μεταξύ των ρευμάτων κειμένου και των δυαδικών ρευμάτων είναι ότι γίνεται μετατροπή κάποιων ειδικών χαρακτήρων στα αρχεία κειμένου κατά την ανάγνωση και εγγραφή.
- ☑ Η τιμή ενός αντικειμένου ρεύματος είναι αληθής όταν δεν έχουν παρουσιαστεί σφάλματα κατά τη διαδικασία ανάγνωσης/εγγραφής.
- ❑ Ένα αρχείο που έχει γραφεί με μορφοποιημένη έξοδο είναι κατάλληλο για τυχαία προσπέλαση.
- ☑ Η τυχαία προσπέλαση προϋποθέτει εγγραφές σταθερού μήκους.

18.15

```
#include <iostream>
#include <fstream>
using namespace std;
class stoixeia
{
public:
   char eponymo[30];
   char taxi[5];
   float mesos_oros;
   int ilikia;
};
int main()
{
   stoixeia eggr;
   char max_onoma[30];
   float max_mo=0;
   fstream file;
   file.open("sxoleio",ios::in|ios::binary);
   file.seekg(14*sizeof(stoixeia));
   file.read(reinterpret_cast<char*>(&eggr),sizeof(eggr));
   if (file)
   {
      cout<<eggr.eponymo<<endl;
      cout<<eggr.taxi<<endl;
      cout<<eggr.mesos_oros<<endl;
      cout<<eggr.ilikia<<endl;
      cout<<"----------------------"<<endl;
   }
   file.close();
   return 0;
}
```

Στο αντικείμενο **eggr** θα καταχωριστούν τα στοιχεία της εγγραφής που θα διαβάσουμε από το αρχείο.

Τοποθετεί τον δείκτη ανάγνωσης του αρχείου στην αρχή της 15ης εγγραφής.

Διαβάζει τα στοιχεία της 15ης εγγραφής και τα καταχωρίζει στο αντικείμενο **eggr**.

Εμφανίζει τα στοιχεία της εγγραφής που διαβάστηκε.

18.16

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
using namespace std;

class stoixeia
{
public:
   char eponymo[30];
   char taxi[5];
   float mesos_oros;
   int ilikia;
};

int main()
{
   stoixeia eggr;
   char max_onoma[30];
   float max_mo=0;
   fstream file;
   file.open("sxoleio",ios::in|ios::out|ios::binary);
   file.seekp(14*sizeof(stoixeia));
   cout<<"Επώνυμο :";
   cin>>eggr.eponymo;
   cout<<"Τάξη :";
   cin>>eggr.taxi;
   cout<<"Μέσος όρος :";
   cin>>eggr.mesos_oros;
   cout<<"Ηλικία :";
   cin>>eggr.ilikia;
   file.write(reinterpret_cast<char*>(&eggr),sizeof(eggr));
   file.close();
   return 0;
}
```

Στο αντικείμενο **eggr** θα καταχωρίζονται τα στοιχεία της εγγραφής που θα γράψουμε στο αρχείο.

Ανοίγει το αρχείο για ανάγνωση και εγγραφή. Προσοχή: αν το ανοίξουμε μόνο για εγγραφή, θα διαγράψουμε όλες τις υπάρχουσες εγγραφές.

Τοποθετεί τον δείκτη θέσης του αρχείου στην αρχή της 15ης εγγραφής.

Ζητάει από τον χρήστη να πληκτρολογήσει τα στοιχεία του μαθητή και τα καταχωρίζει στα αντίστοιχα μέλη του αντικειμένου **eggr**.

Καταχωρίζει τα στοιχεία του αντικειμένου **eggr** στη 15η εγγραφή του αρχείου.

18.17

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstring>

using namespace std;

class stoixeia
{
public:
   char eponymo[30];
   char taxi[5];
   float mesos_oros;
   int ilikia;
};

int main()
{
   stoixeia eggr;
   char max_onoma[30];
   float max_mo=0;
   fstream file;
   file.open("sxoleio",ios::in|ios::binary);
```

Στο αντικείμενο **eggr** θα καταχωρίζονται τα στοιχεία κάθε εγγραφής που διαβάζουμε από το αρχείο.

Στον πίνακα **max_onoma[]** θα καταχωριστούν τα στοιχεία του μαθητή με τον μεγαλύτερο μέσο όρο. Στη μεταβλητή **max_mo** θα καταχωρίζεται ο τρέχων μεγαλύτερος μέσος όρος. Θέτουμε αρχική τιμή 0.

```
   // Τοποθετούμε τον δείκτη στην αρχή του αρχείου
   file.seekg(0);
   while (file)
   {
      file.seekg(i*sizeof(stoixeia));
      file.read(reinterpret_cast<char*>(&eggr),sizeof(eggr));
      if (file)
      {
         if (eggr.mesos_oros>max_mo)
         {
            max_mo=eggr.mesos_oros;
            strcpy(max_onoma,eggr.eponymo);
         }
      }
   }
   cout<<max_onoma<<" "<<max_mo<<endl;
   file.close();
   return 0;
}
```

Διαβάζει μία έγγραφή από το αρχείο και την καταχωρίζει στο αντικείμενο **eggr** (βλέπε παράδειγμα Π18.5 του βιβλίου).

☞ Διαβάζουμε μία προς μία τις εγγραφές και αποθηκεύουμε τα στοιχεία κάθε εγγραφής στο αντικείμενο eggr. Η επαναλαμβανόμενη διαδικασία σταματάει όταν διαβάσουμε και την τελευταία εγγραφή του αρχείου, όποτε στην επόμενη προσπάθεια ανάγνωσης η τιμή του ρεύματος file γίνεται ψευδής και σταματάει η επαναληπτική διαδικασία.

☞ Μετά το τέλος της επαναληπτικής διαδικασίας, ο πίνακας onoma[] θα περιέχει το όνομα του μαθητή με τον μεγαλύτερο μέσο όρο και η μεταβλητή max_mo θα περιέχει τον μέσο όρο του.

18.18

```
#include <iostream>
#include <iomanip>
#include <fstream>

using namespace std;

class stoixeia
{
public:
   char eponymo[30];
   char taxi[5];
   float mesos_oros;
   int ilikia;
};

int main()
{
   stoixeia eggr;
   char arx;
   int i;
   fstream file;
   file.open("sxoleio",ios::in|ios::binary);
   cout<<"Dose Xaraktira:";
   cin>>arx;
   file.seekg(0);
   while (file)
   {
      file.read(reinterpret_cast<char*>(&eggr),sizeof(eggr));
      if (file)
      {
         if (eggr.eponymo[0]==arx)
         {
            cout<<eggr.eponymo<<" "<<eggr.ilikia<<endl;
```

Στο αντικείμενο **eggr** θα καταχωρίζονται τα στοιχεία της κάθε εγγραφής που διαβάζουμε από το αρχείο.

Διαβάζει μία-μία τις εγγραφές από το αρχείο και τις καταχωρίζει στο αντικείμενο **eggr**.

Στην περίπτωση που ο πρώτος χαρακτήρας του επωνύμου είναι αυτός που δώσαμε (arx), τότε εμφανίζει το επώνυμο και την ηλικία του μαθητή.

```
            }
         }
      }
      file.close();
      return 0;
}
```

☞ Διαβάζουμε μία προς μία τις εγγραφές και αποθηκεύουμε τα στοιχεία της κάθε εγγραφής στο αντικείμενο eggr. Η επαναλαμβανόμενη διαδικασία σταματάει όταν διαβάσουμε και την τελευταία εγγραφή του αρχείου, όποτε στην επόμενη προσπάθεια ανάγνωσης η τιμή του ρεύματος file γίνεται ψευδής και σταματάει η επαναληπτική διαδικασία.

18.19

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
   char ch,file_in[20];
   ifstream fin;
   cout<<"Δώσε όνομα αρχείο εισόδου:";
   cin>>file_in;
   fin.open(file_in,ios::binary);
   if (!fin)
   {
      cout<<"Πρόβλημα στο αρχείο εισόδου\n";
      exit(2);
   }
   while (fin.get(ch))
   {
      ch--;
      cout<<ch;
   }
   fin.close();
   return 0;
}
```

Άνοιγμα του αρχείου, σε δυαδική μορφή, για ανάγνωση δεδομένων.

Διαβάζει έναν-έναν τους χαρακτήρες από το αρχείο και τους καταχωρίζει στη μεταβλητή **ch**.

Εμφανίζει στην οθόνη τον προηγούμενο χαρακτήρα από αυτόν που διάβασε.

18.20

```
class stoixeia
{
public:
   char eponymo[30];
   char taxi[5];
   float mesos_oros;
   int ilikia;
   friend ostream &operator<<(ostream &file_out, stoixeia &obj);
   friend istream &operator>>(istream &file_in, stoixeia &obj);
};

ostream &operator<<(ostream &file_out, stoixeia &obj)
{
   file_out << obj.eponymo<<endl;
   file_out << obj.taxi<<endl;
   file_out << obj.mesos_oros<<endl;
   file_out << obj.ilikia<<endl;
   return file_out;
}
```

Συνήθως οι συναρτήσεις υπερφόρτωσης των τελεστών εισαγωγής και εξαγωγής δηλώνονται ως φίλιες για να έχουν πρόσβαση στα ιδιωτικά της μέλη της κλάσης . Βέβαια, στη συγκεκριμένη περίπτωση δεν είναι απαραίτητο διότι όλα τα μέλη της κλάσης **stoixeia** είναι δημόσια.

Η συνάρτηση υπερφόρτωσης του τελεστή εισαγωγής (<<) γράφει τα στοιχεία ενός αντικειμένου της κλάσης **stoixeia** στο ρεύμα εξόδου.

```
istream &operator>>(istream &file_in, stoixeia &obj)
{
   file_in >> obj.eponymo ;
   file_in >> obj.taxi;
   file_in >> obj.mesos_oros;
   file_in >> obj.ilikia;
   return file_in;
}
```

Η συνάρτηση υπερφόρτωσης του τελεστή εξαγωγής (>>) εξάγει τέσσερα στοιχεία από το ρεύμα εισόδου και τα καταχωρίζει στα μέλη **eponymo**, **taxi**, **mesos_oros** και **ilikia** του αντικειμένου **obj**.

18.21

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

int main()
{
   stringstream ss;
   char ch=' ';
   int a;
   float f;
   string s;
   ss << "25";
   ss <<ch<<35;
   ss <<".";
   ss <<15+20<<" Τέλος";
   ss>>a>>f>>s;
   cout << a << endl << f << endl << s << endl;
   return 0;
}
```

```
25
35.75
Τέλος
```

18.22

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main(int argc,char *argv[])
{
   char ch1,ch2;
   ifstream fin1,fin2;
   bool isa=true;
   int ar=0;
   if (argc!=3)
   {
      cout<<"Λάθος πλήθος παραμέτρων"<<endl;
      return 1;
   }
   fin1.open(argv[1],ios::binary);
   if (!fin1)
   {
      cout<<"Πρόβλημα στο αρχείο 1"<<endl;
      return 2;
   }
   fin2.open(argv[2],ios::binary);
   if (!fin2)
   {
      cout<<"Πρόβλημα στο αρχείο 2"<<endl;
      return 3;
   }
```

Άνοιγμα του πρώτου αρχείου, σε δυαδική μορφή, για ανάγνωση δεδομένων.

```
while (fin1)
   {
      fin1.get(ch1);
      fin2.get(ch2);
      ar++;
      if (ch1!=ch2)
      {
         isa=false;
         break;
      }
   }
   fin1.close();
   fin2.close();
   if (isa)
      cout<<"Τα αρχεία είναι ίδια"<<endl;
   else
      cout<<"Τα αρχεία διαφέρουν στο byte "<<ar<<endl;
   return 0;
}
```

Ανάγνωση ενός χαρακτήρα από κάθε αρχείο.

Η μεταβλητή **ar** μετράει το σημείο (byte) στο οποίο βρισκόματε.

Στην περίπτωση που οι δύο χαρακτήρες δεν είναι ίσοι, διακόπτεται η ανάγνωση.

18.23

```
#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>

using namespace std;

int main()
{
   char ch;
   string onoma;
   int codes[256],i;
   ifstream infile;
   cout << "Όνομα αρχείου:";
   cin>>onoma;
   infile.open(onoma);
   if (!infile)
   {
      cout << "Πρόβλημα στο άνοιγμα του αρχείου" << endl;
      return 1;
   }
   for (i=0;i<256;i++) codes[i]=0;
   while (!infile.eof())
   {
      infile.get(ch);
      codes[ch]++;
   }
   infile.close();
   for (i=32;i<256;i++)
      if (codes[i]!=0) printf("%c->%d\n",i,codes[i]);
   return 0;
}
```

Ζητάει να πληκτρολογηθεί το όνομα του αρχείου, το οποίο καταχωρίζεται στο αντικείμενο **onoma**. Έπειτα ανοίγει το αρχείο με το συγκεκριμένο όνομα.

Μηδενίζει τον πίνακα μετρητών **codes[]**.

Διαβάζει έναν-έναν τους χαρακτήρες του αρχείου.

Αυξάνει τη θέση μνήμης του πίνακα **codes[]**, ανάλογα με τον κωδικό του χαρακτήρα **ch**.

☞ Χρησιμοποιούμε τον πίνακα codes[] ως πίνακα μετρητών για κάθε χαρακτήρα, ανάλογα με τον κωδικό του ASCII. Για τον λόγο αυτό, ο πίνακας αρχικά μηδενίζεται. Για παράδειγμα, η θέση codes[65] μετράει τους χαρακτήρες 'A' με κωδικό 65.

☞ Τέλος εμφανίζει τη μέτρηση των εκτυπώσιμων χαρακτήρων με κωδικούς από 32 μέχρι 255.

18.24

```
class student
{
public:
   string onoma;
   string code;
   int etos;
   float batmoi[5][10];
   .....
   void grapse();
   void diabase();
   .....
};

void student::grapse()
{
   int i,j;
   ofstream outfile;
   outfile.open(code);
   outfile<<onoma<<endl;
   outfile<<code<<endl;
   outfile<<etos<<endl;
   for (i=0;i<5;i++)
      for (j=0;j<10;j++)
         outfile<<batmoi[i][j]<<endl;
   outfile.close();
}
void student::diabase()
{
   int i,j;
   ifstream infile;
   infile.open(code);
   getline(infile,onoma);
   getline(infile,code);
   infile>>etos;
   for (i=0;i<5;i++)
      for (j=0;j<10;j++)
         infile >> batmoi[i][j];
   infile.close();
}
```

Η μέθοδος **grapse()** γράφει τα στοιχεία ενός φοιτητή σε ένα αρχείο με όνομα την τιμή της μεταβλητής-μέλους **code** του αντικειμένου.

Η μέθοδος **diabase()** διαβάζει τα στοιχεία ενός φοιτητή από ένα αρχείο με όνομα την τιμή της μεταβλητής-μέλους **code** του αντικειμένου. Τα στοιχεία που διαβάζει τα καταχωρίζει στις μεταβλητές-μέλη του αντικειμένου.

Ασκήσεις Κεφαλαίου 19

19.1

```
float par(int n)
{
   float p;
   if (n==1) return 1.0;
   p=1.0/n+par(n-1);
   return p;
}
```

Μη αναδρομική περίπτωση.

Η παράσταση 1.0/n έχει αποτέλεσμα **double** (με δεκαδικά). Αν ήταν 1/n το αποτέλεσμα θα ήταν τύπου **int**.

19.2

```
int par(int n)
{
   int p;
   p=n+par(n-1);
   return p;
}
```

Η συνάρτηση δεν διαθέτει μη-αναδρομική περίπτωση.

19.3

☞ Η συνάρτηση επιστρέφει ως τιμή το άθροισμα της σειράς n + n/2 + n/4 .... μέχρι το πηλίκο να είναι διαφορετικό από το 1.

☞ Η κλήση της par(20) θα επιστρέψει το 37 (20 + 10 + 5 + 2 + 0).

19.4

```
#include <iostream>
#include <cmath>

using namespace std;

double par(int n)
{
   double p;
   if(n==0) return 0;
   p=sqrt(2+par(n-1));
   return p;
}


double all(int k)
{
   double p;
   if(k==0) return 2;
   p=2/par(k)*all(k-1);
   return p;
}

int main()
{
   cout << all(1000);
   return 0;
}
```

Μη αναδρομική περίπτωση,

Αναδρομική κλήση συνάρτησης.

Μη αναδρομική περίπτωση

Αναδρομική κλήση συνάρτησης.

☞ Η συνάρτηση par() χρησιμοποιεί αναδρομική διαδικασία για τον υπολογισμό του παρονομαστή κάθε μέλους της ακολουθίας.

☞ Η συνάρτηση all() χρησιμοποιεί αναδρομική διαδικασία για τον υπολογισμό του γινομένου όλων των μελών της ακολουθίας.

19.5

- ☑ Κάθε αναδρομική συνάρτηση πρέπει να έχει μία τουλάχιστον μη αναδρομική περίπτωση.
- ❑ Όλες οι συναρτήσεις μπορούν να γραφούν με αναδρομική μορφή.
- ☑ Μια αναδρομική συνάρτηση μπορεί να προκαλέσει εξάντληση της μνήμης του Η/Υ.
- ❑ Όλες οι γλώσσες προγραμματισμού υποστηρίζουν αναδρομικές συναρτήσεις.
- ☑ Μια αναδρομική συνάρτηση πρέπει να έχει τουλάχιστον μία παράμετρο.

19.6

```
#include <iostream>
using namespace std;

int mkd(int m, int n);

int main()
{
   int a,b;
   cout<<"Δώσε δύο αριθμούς :";
   cin >> a >> b;
   cout << "Ο ΜΚΔ του "<<a<<" και του "<<b<<" είναι "<<mkd(a,b)<<endl;
   return 0;
}

int mkd(int m, int n)
{
   int temp;
   if (n>m)
   {
      temp=m;
      m=n;
      n=temp;
   }
   if (n==0)
      return m;
   else
      return mkd(n, m % n);
}
```

Αντιμετάθεση των **m** και **n** ώστε η μεταβλητή **m** να περιέχει πάντα τον μεγαλύτερο αριθμό.

Μη αναδρομική περίπτωση

Αναδρομική κλήση της συνάρτησης.

19.7

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
   int i,k,n;
   char ch;
   string lex;
   cout<<"Δώσε λέξη :";
   cin>>lex;
   n=lex.size();
   for(i=1;i<n;i++)
   {
      for(k=n-1;k>=i;k--)
      {
         if(lex[k]<lex[k-1])
         {
            ch=lex[k];
            lex[k]=lex[k-1];
            lex[k-1]=ch;
```

Στη μεταβλητή **n** καταχωρίζεται το πλήθος των χαρακτήρων του αντικειμένου **lex**.

Χρησιμοποιείται ο αλγόριθμος *bubble sort* για την ταξινόμηση των χαρακτήρων του αντικειμένου **lex**.

```
            }
         }
      }
      cout<<lex;
      return 0;
}
```

19.8

```
#include <iostream>
using namespace std;

class lexi
{
   string data;
public:
   void set(string d);
   void show();
   void sort();
};

void lexi::set(string d)
{
   data=d;
}

void lexi::show()
{
   cout<<data<<endl;
}

void lexi::sort()
{
   char ch;
   int i,n,k;
   n=data.size();
   for(i=1;i<n;i++)
   {
      for(k=n-1;k>=i;k--)
      {
         if(data[k]<data[k-1])
         {
            ch=data[k];
            data[k]=data[k-1];
            data[k-1]=ch;
         }
      }
   }
}

int main()
{
   lexi l1,l2;
   l1.set("656752613");
   l2.set("Αποτελέσματα");
   l1.show(); // εμφανίζεται η λέξη του αντικειμένου l1
   l2.show(); // εμφανίζεται η λέξη του αντικειμένου l2
   l1.sort(); // γίνεται ταξινόμηση της λέξης του αντικειμένου l1
   l2.sort(); // γίνεται ταξινόμηση της λέξης του αντικειμένου l2
   l1.show(); // εμφανίζεται η λέξη του αντικειμένου l1
   l2.show(); // εμφανίζεται η λέξη του αντικειμένου l2
   return 0;
}
```

Ορισμός της κλάσης **lexi**. Η κλάση διαθέτει το μέλος **data** για την καταχώριση μιας λέξης.

Η μέθοδος **set()** καταχωρίζει τους χαρακτήρες της παραμέτρου της στο μέλος **data**.

Η μέθοδος **show()** εμφανίζει τους χαρακτήρες της λέξης.

Η μέθοδος **sort()** ταξινομεί τους χαρακτήρες λέξης.

Δημιουργούνται δύο αντικείμενα της κλάσης **lexi**.

Δίνονται τιμές στο μέλος **data** των αντικειμένων **l1** και **l2**.

```
656752613
Αποτελέσματα
123556667
Αέααελμοπσττ
```

19.9

```
void order(string x[100])
{
   int i,k;
   string temp;
   for(i=1;i<100;i++)
   {
      for(k=99;k>=i;k--)
      {
         if(x[k][1]<x[k-1][1])
         {
            temp=x[k];
            x[k]=x[k-1];
            x[k-1]=temp;
         }
      }
   }
}
```

Χρησιμοποιείται ο αλγόριθμος *bubble sort*.

Συγκρίνεται ο δεύτερος χαρακτήρας των συμβολοσειρών.

Αντιμετάθεση των συμβολοσειρών.

19.10

```
int i=0,p1,p2;
char ch1,ch2,temp;
ch1=ch2=lex[0];
p1=p2=0;
while (i<lex.size())
{
   if (ch1>lex[i])
   {
      ch1=lex[i];
      p1=i;
   }
   if (ch2<lex[i])
   {
      ch2=lex[i];
      p2=i;
   }
   i++;
}
temp=lex[0];
lex[0]=lex[p2];
lex[p2]=temp;
temp=lex[lex.size()-1];
lex[lex.size()-1]=lex[p1];
lex[p1]=temp;
.....
```

Εντοπίζει τον μεγαλύτερο (σε κωδικό) χαρακτήρα του **lex** και καταχωρίζει τη θέση της πρώτης εμφάνισής του στη μεταβλητή **p1**.

Εντοπίζει τον πρώτο μικρότερο (σε κωδικό) χαρακτήρα του **lex** και καταχωρίζει τη θέση του στη μεταβλητή **p2**.

Αντιμεταθέτει τον μεγαλύτερο χαρακτήρα (στη θέση p1) με τον πρώτο χαρακτήρα (θέση 0).

Αντιμεταθέτει τον μικρότερο χαρακτήρα (στη θέση p2) με τον τελευταίο χαρακτήρα (θέση lex.size()-1).

☞ Ο παραπάνω κώδικας εντοπίζει τον μεγαλύτερο (σε κωδικό ASCII) χαρακτήρα του lex και τον αντιμεταθέτει με τον χαρακτήρα της πρώτης θέσης. Επίσης, εντοπίζει τον μικρότερο (σε κωδικό) χαρακτήρα και τον αντιμεταθέτει με τον χαρακτήρα της τελευταίας θέσης.

☞ Ο μεγαλύτερος χαρακτήρας της λέξης είναι ο «Φ» ο οποίος αντιμετατίθεται με τον πρώτο χαρακτήρα, ενώ ο μικρότερος, που είναι το πρώτο «Α», αντιμετατίθεται με τον τελευταίο χαρακτήρα. Επομένως, μετά από την εκτέλεση του παραπάνω κώδικα το lex θα περιέχει τους χαρακτήρες: "ΦΠΣΖΟΚΕΣΑΛΑΚΗΑ".

19.11

- ☑ Η δυαδική αναζήτηση προϋποθέτει ταξινομημένα δεδομένα.
- ☑ Οι διαδικασίες ταξινόμησης είναι γενικά χρονοβόρες σε μεγάλους πίνακες.
- ❑ Σε έναν μεγάλο πίνακα με τυχαίους αριθμούς η καλύτερη μέθοδος ταξινόμησης είναι η μέθοδος φυσαλίδας.
- ❑ Σε έναν μισοταξινομημένο πίνακα η καλύτερη μέθοδος ταξινόμησης είναι η *quick sort*.
- ☑ Σε έναν πίνακα με τυχαίους αριθμούς, η μόνη μέθοδος αναζήτησης είναι η σειριακή.
- ☑ Η ταξινόμηση ενός πίνακα αντικειμένων προϋποθέτει την υπερφόρτωση των τελεστών σύγκρισης για την κλάση των αντικειμένων.

19.12

```
.....
bool tmima::find(string onom)
{
   int i;
   for(i=0;i<plithos;i++)
   {
      if (foit[i]==onom)
      {
         return true;
      }
   }
   return false;
}

int main()
{
   tmima t1("Πολιτισμική Τεχνολογία");
   // Προσθήκη τεσσάρων φοιτητών στο τμήμα t1
   t1.add_foititi("Οκνηρός Νικόλαος");
   t1.add_foititi("Ζαμανφουτίδου Αναστασία");
   t1.add_foititi("Ξαπλώστρας Τηλέμαχος");
   t1.add_foititi("Αραχτός Αναστάσιος");
   t1.display();
   if (t1.find("Ζαμανφουτίδου Αναστασία"))
      cout<<"NAI1"<<endl;
   else
      cout<<"OXI1"<<endl;
   if (t1.find("Τεμπελχανάς Νικόλαος"))
      cout<<"NAI2"<<endl;
   else
      cout<<"OXI2"<<endl;
   return 0;
}
```

Στην παράμετρο **onom** μεταβιβάζεται το όνομα του φοιτητή που αναζητούμε.

Αν εντοπιστεί το όνομα τερματίζεται η αναζήτηση και επιστρέφει τιμή **true**, διαφορετικά επιστρέφει τιμή **false**.

Εντοπίζει τον συγκεκριμένο φοιτητή μέσα στο τμήμα **t1** και εμφανίζει «NAI1».

Δεν εντοπίζει τον φοιτητή και εμφανίζει «OXI2».

19.13

```
class person
{
   string onoma;
   string eponymo;
   int ilikia;
public:
   void set(string o, string e, int i);
   void display();
   .....
   bool operator<(person op2);
};
```

Μέθοδος υπερφόρτωσης του τελεστή <. Είναι απαραίτητη ώστε να μπορεί να χρησιμοποιηθεί ο τελεστής για τη σύγκριση αντικειμένων της κλάσης. Ορίζεται μετά!

```
bool person::operator<(person op2)
{
   if (ilikia<op2.ilikia)
      return true;
   else
      return false;
}
void order(person x[],int n)
{
   int i,k;
   person temp;
   for(i=1;i<n;i++)
   {
      for(k=n-1;k>=i;k--)
      {
         if (x[k]<x[k-1])
         {
            temp=x[k];
            x[k]=x[k-1];
            x[k-1]=temp;
         }
      }
   }
}
```

Στη μέθοδο υπερφόρτωσης του τελεστή < η σύγκριση των δύο αντικειμένων γίνεται με βάση την ηλικία τους.

Χρησιμοποιείται ο αλγόριθμος *bubble sort* για την ταξινόμηση ενός πίνακα της κλάσης **person**.

Τα αντικείμενα συγκρίνονται με χρήση του υπερφορτωμένου τελεστή <.

19.14

```
int tmima::check_sort()
{
   int ayxousa=1,fthinousa=1,i;
   for (i=0;i<plithos-1;i++)
   {
      if (foit[i]>foit[i+1]) ayxousa=0;
      if (foit[i]<foit[i+1]) fthinousa=0;
   }
   if (ayxousa)
      return 1;
   else if (fthinousa)
      return -1;
   else
      return 0;
}
```

Ελέγχονται οι διαδοχικές θέσεις του πίνακα. Αν βρεθούν τιμές με αντίθετη σειρά, τότε μηδενίζονται οι αντίστοιχες μεταβλητές.

Αν η μεταβλητή **ayxousa** παραμείνει με τιμή 1, τότε ο πίνακας έχει αύξουσα ταξινόμηση. Αν η μεταβλητή **fthinousa** παραμείνει με τιμή 1, τότε ο πίνακας έχει φθίνουσα ταξινόμηση. Αν και οι δύο μεταβλητές έχουν τιμή 0, τότε ο πίνακας δεν είναι ταξινομημένος. Στην περίπτωση που και οι δύο μεταβλητές έχουν τιμή 1, τότε ο πίνακας περιέχει τον ίδιο αριθμό σε όλες τις θέσεις του.

19.15

```
#include <iostream>
#include <string>
using namespace std;

#define size 50 // Καθορίζει το πλήθος των χωρών

int main()
{
   int i,j,k,bathmoi[size][10],min,max,ath;
   string xores[size],temp;
   float mo[size],tm;
   for(i=0;i<size;i++)              // Διαβάζει τα ονόματα των χωρών
   {
      cout << "Δώσε όνομα χώρας "<<i+1<<":";
      cin>>xores[i];
```

```
   }
   for(i=0;i<size;i++)              // Διαβάζει τους βαθμούς κάθε χώρας
   {
      cout<<"Δώσε βαθμούς χώρας "<<xores[i]<<endl;
      for (j=0;j<10;j++)
      {
         cout << "Βαθμός "<<j+1<<":";
         cin >> bathmoi[i][j];
      }
   }
   for(i=0;i<size;i++)              // Υπολογίζει τον μέσο όρο των βαθμών κάθε χώρας
   {
      min=max=bathmoi[i][0];
      ath=0;
      for (j=0;j<10;j++)
      {
         ath=ath+bathmoi[i][j];
         if (bathmoi[i][j]>max) max=bathmoi[i][j]; // Βρίσκει τον μέγιστο βαθμό κάθε χώρας
         if (bathmoi[i][j]<min) min=bathmoi[i][j]; // Βρίσκει τον ελάχιστο βαθμό κάθε χώρας
      }
      mo[i]=(ath-min-max)/8.0;                     // Υπολογίζει τον μέσο όρο των βαθμών μιας χώρας
   }
   for(i=1;i<size;i++)
   {
      for(k=size-1;k>=i;k--)
      {
         if(mo[k]>mo[k-1])
         {
            tm=mo[k];
            mo[k]=mo[k-1];
            mo[k-1]=tm;
            temp=xores[k];
            xores[k]=xores[k-1];
            xores[k-1]=temp;
         }
      }
   }
   for(i=0;i<size;i++)
   {
      cout <<xores[i]<<" "<<mo[i]<<endl;
   }
   return 0;
}
```

Από το άθροισμα αφαιρούνται ο ελάχιστος και ο μέγιστος βαθμός.

Ταξινόμηση του πίνακα των μέσων όρων με ταυτόχρονη αντιμετάθεση του ονόματος κάθε χώρας.

Εμφάνιση της βαθμολογίας των χωρών

19.16

```
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

#define size 50     // Καθορίζει το πλήθος των αθλητών

class agonas
{
   string onomata[size];
   float ripseis[size][6];
public:
   void diabase_athlites();
   void diabase_ripseis();
   void sort_ripseis();
```

Η κλάση **agonas** χρησιμοποιείται για τη διαχείριση αγώνων. Τα ονόματα των αθλητών και οι ρίψεις τους καταχωρίζονται στους πίνακες-μέλη της κλάσης.

Οι μέθοδοι τις κλάσεις υλοποιούν τις ζητούμενες λειτουργίες του προγράμματος.

```
   void sort_onomata();
   void display();
};

void agonas::diabase_athlites()
{
   int i;
   for (i=0;i<size;i++)
   {
      cout << "Δώσε όνομα αθλητή "<<i+1<<":";
      getline(cin,onomata[i]);
   }
}
void agonas::diabase_ripseis()
{
   int i,j;
   for (i=0;i<size;i++)
   {
      cout << "Δώσε ριψεις αθλητη "<< onomata[i]<<endl;
      for (j=0;j<6;j++)
      {
         cout<<"Ρίψη "<<j+1<<":";
         cin>>ripseis[i][j];
      }
   }
}
void agonas::sort_ripseis()
{
   int i,g,k;
   float tm;
   for (g=0;g<size;g++)
   {
      for (i=1;i<6;i++)
      {
         for (k=5;k>=i;k--)
         {
            if (ripseis[g][k]<ripseis[g][k-1])
            {
               tm=ripseis[g][k];
               ripseis[g][k]=ripseis[g][k-1];
               ripseis[g][k-1]=tm;
            }
         }
      }
   }
}

void agonas::sort_onomata()
{
   int i,k,r;
   string tm;
   float tr;
   for(i=1;i<size;i++)
   {
      for(k=size-1;k>=i;k--)
      {
         if(onomata[k]<onomata[k-1])
         {
            tm=onomata[k];
            onomata[k]=onomata[k-1];
            onomata[k-1]=tm;
```

Η μέθοδος **diabase_athlites()** διαβάζει τα ονόματα των αθλητών και τα καταχωρίζει στον αντίστοιχο πίνακα-μέλος της κλάσης.

Η μέθοδος **diabase_ripseis()** διαβάζει τις ρίψεις των αθλητών και τις καταχωρίζει στον αντίστοιχο πίνακα-μέλος της κλάσης.

Η μέθοδος **sort_ripseis()** ταξινομεί τις ρίψεις κάθε αθλητή με αύξουσα σειρά.

Η μέθοδος **sort_onomata()** ταξινομεί τα ονόματα των αθλητών με αύξουσα αλφαβητική σειρά.

```
            for (r=0;r<6;r++)
            {
               tr=ripseis[k][r];
               ripseis[k][r]= ripseis[k-1][r];
               ripseis[k-1][r]=tr;
            }
         }
      }
   }
}
void agonas::display()
{
   int i,j;
   for (i=0;i<size;i++)
   {
      cout<<setw(15)<<left<<onomata[i];
      for (j=0;j<6;j++)
         cout<<setw(4)<<right<<ripseis[i][j];
      cout<<endl;
   }
}
int main()
{
   agonas a1;
   a1.diabase_athlites();
   a1.diabase_ripseis();
   a1.sort_ripseis();
   a1.sort_onomata();
   a1.display();
   return 0;
}
```

Όταν αντιμετατίθενται τα ονόματα αντιμετατίθενται και οι αντίστοιχες ρίψεις των αθλητών.

Η μέθοδος **display()** εμφανίζει τους αθλητές και τις ρίψεις τους.

Δημιουργεί ένα αντικείμενο της κλάσης **agonas** με όνομα **a1**.

Διαβάζει τα ονόματα των αθλητών και τις ρίψεις τους για τον αγώνα **a1**.

Ταξινομεί τις ρίψεις των αθλητών του αγώνα **a1**, έπειτα ταξινομεί τα ονόματα, και μετά εμφανίζει τα ονόματα και τις ρίψεις τους.

☞ Οι μέθοδοι sort_ripseis() και sort_onomata() θα μπορούσαν να καλούνται από τη μέθοδο display() και όχι στη main(). Σε αυτή τη περίπτωση θα μπορούσαν να δηλωθούν ως ιδιωτικές.

19.17

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

class rouleta
{
   int numbers[1000];
   int fores;
public:
   rouleta() {fores=0;}
   int rikse();
   void display_syxna();
};

int rouleta::rikse()
{
   int arithmos;
   arithmos=rand()%37;
   if (fores==1000)
   {
      cout << "Γέμισε το ιστορικό ρίψεων";
      cout << " δεν επιτρέπονται άλλες ρίψεις"<<endl;
      return -1;
   }
```

Η κλάση **rouleta** χρησιμοποιείται για τη διαχείριση των ρουλετών του καζίνου. Το ιστορικό ρίψεων της ρουλέτας καταχωρίζεται στον πίνακα **numbers** και το πλήθος ρίψεων στη μεταβλητή-μέλος **fores**.

Μέθοδος δόμησης της κλάσης. Δίνει αρχική τιμή 0 στο πλήθος ρίψεων.

Οι μέθοδοι της κλάσης υλοποιούν τις ζητούμενες λειτουργίες του προγράμματος.

Η μέθοδος **rikse()** προσομοιώνει μια ρίψη της ρουλέτας. Επιστρέφει έναν τυχαίο αριθμό από 0 έως 36 και τον καταχωρίζει στο ιστορικό ρίψεων. Αν έχουμε ήδη φτάσει στις 1000 ρίψεις δεν επιτρέπει άλλες και εμφανίζει σχετικό μήνυμα.

```
   numbers[fores]=arithmos;
   fores++;
   return arithmos;
}

void rouleta::display_syxna()
{
   int num[37],i,max;
   for (i=0;i<36;i++) num[i]=0;     // Μηδενίζει τον πίνακα num[]
   for (i=0;i<fores;i++)
      num[numbers[i]]++;            // Ανάλογα με το νούμερο αυξάνει την αντίστοιχη θέση του πίνακα num[]
   max=num[0];
   for (i=0;i<36;i++)
      if (num[i]>max) max=num[i];   // Βρίσκει τον μεγαλύτερο του πίνακα num[]
         cout << "Τα παρακάτω νούμερα είναι τα πιο συχνά! Βγήκαν "
              <<max<<" φορές το καθένα"<<endl;
   for (i=0;i<36;i++)
      if (num[i]==max) cout << i << endl;
}

int main()
{
   rouleta r1,r2;
   int i;
   srand(time(NULL));
   for (i=1;i<=800;i++) r1.rikse();
   for (i=1;i<=800;i++) r2.rikse();
   r1.display_syxna();
   r2.display_syxna();
   return 0;
}
```

Η μέθοδος **display_syxna()** εμφανίζει τους αριθμούς με τη συχνότερη εμφάνιση. Χρησιμοποιεί τον πίνακα **num** ως μετρητή των αντίστοιχων τιμών. Π.χ η θέση mum[4] μετράει τις εμφανίσεις του 4, η num[32] τις εμφανίσεις του 32, κ.ο.κ.

Δημιουργεί δύο αντικείμενα της κλάσης **rouleta** με ονόματα **r1** και **r2**.

Προσομοιώνει 800 ρίψεις και στις δύο ρουλέτες.

Εμφανίζει τα νούμερα με τη συχνότερη εμφάνιση σε κάθε μία από τις ρουλέτες.

19.18

```
#include <iostream>
#include <string>
using namespace std;

#define size 50 // Καθορίζει το πλήθος των χωρών

class diagonismos
{
   string xores[size];
   int bathmoi[size][10];
   float mo[size];
public:
   void diavase_xores();
   void diavase_bathmoys();
   void ypologismos_mesoy();
   void taxinomisi();
   void display();
};

void diagonismos::diavase_xores()
{
   int i;
   for(i=0;i<size;i++)
   {
      cout << "Δώσε όνομα χώρας "<<i+1<<":";
      cin>>xores[i];
   }
}
```

Η κλάση **diagonismos** χρησιμοποιείται για τη διαχείριση διαγωνισμών τραγουδιού. Τα ονόματα των χωρών, οι βαθμολογίες τους και ο μέσος όρος που υπολογίζεται, καταχωρίζονται στους αντίστοιχους πίνακες-μέλη της κλάσης.

Οι μέθοδοι της κλάσης υλοποιούν τις ζητούμενες λειτουργίες του προγράμματος.

Η μέθοδος **diavase_xores()** διαβάζει τα ονόματα των χωρών και τα καταχωρίζει στον αντίστοιχο πίνακα-μέλος.

```
void diagonismos::diavase_bathmoys()
{
   int i,j;
   for(i=0;i<size;i++)
   {
      cout<<"Δώσε βαθμούς χώρας "<<xores[i]<<endl;
      for (j=0;j<10;j++)
      {
         cout << "Βαθμός "<<j+1<<":";
         cin >> bathmoi[i][j];
      }
   }
}
```

Η μέθοδος **diavase_bathmoys()** διαβάζει τους βαθμούς των χωρών και τους καταχωρίζει στον αντίστοιχο πίνακα-μέλος.

```
void diagonismos::ypologismos_mesoy()
{
   int i,j,min,max,ath;
   for(i=0;i<size;i++)
   {
      min=max=bathmoi[i][0];
      ath=0;
      for (j=0;j<10;j++)
      {
         ath=ath+bathmoi[i][j];
         if (bathmoi[i][j]>max) max=bathmoi[i][j]; // Βρίσκει τον μέγιστο βαθμό κάθε χώρας
         if (bathmoi[i][j]<min) min=bathmoi[i][j]; // Βρίσκει τον ελάχιστο βαθμό κάθε χώρας
      }
      mo[i]=(ath-min-max)/8.0;                    // Υπολογίζει τον μέσο όρο των βαθμών κάθε χώρας
   }
}
```

Η μέθοδος **ypologismos_mesoy()** υπολογίζει τους μέσους όρους των βαθμών των χωρών και τους καταχωρίζει στον αντίστοιχο πίνακα-μέλος.

```
void diagonismos::taxinomisi()
{
   int i,k;
   float tm;
   string temp;
   for(i=1;i<size;i++)
   {
      for(k=size-1;k>=i;k--)
      {
         if(mo[k]>mo[k-1])
         {
            tm=mo[k];
            mo[k]=mo[k-1];
            mo[k-1]=tm;
            temp=xores[k];
            xores[k]=xores[k-1];
            xores[k-1]=temp;
         }
      }
   }
}
```

Η μέθοδος **taxinomisi()** ταξινομεί τις χώρες σύμφωνα με τον μέσο όρο της βαθμολογίας τους με φθίνουσα σειρά.

```
void diagonismos::display()
{
   int i;
   for(i=0;i<size;i++)
   {
      cout <<xores[i]<<" "<<mo[i]<<endl;
   }
}
```

Η μέθοδος **display()** εμφανίζει τις χώρες και τον μέσο όρο της βαθμολογίας τους.

```
int main()
{
   diagonismos d1;
   d1.diavase_xores();
   d1.diavase_bathmoys();
   d1.ypologismos_mesoy();
   d1.taxinomisi();
   d1.display();
   return 0;
}
```

Δημιουργεί ένα αντικείμενο της κλάσης **diagonismos** με όνομα **d1**.

Διαβάζει τις χώρες και τους βαθμούς τους για τον διαγωνισμό **d1**.

Υπολογίζει τον μέσο όρο των βαθμών για τις χώρες του διαγωνισμού **d1**.

Ταξινομεί τις χώρες σύμφωνα με τον μέσο όρο των βαθμών τους και ακολούθως τις εμφανίζει.

Ασκήσεις Κεφαλαίου 20

20.1

- ❑ Ο πίνακας αποτελεί μια δυναμική κατανομή μνήμης.
- ☑ Στη δυναμική κατανομή μνήμης, υπάρχει περίπτωση να μην είναι δυνατή η δέσμευση της ποσότητας μνήμης που ζητάμε.
- ☑ Η πρόταση new int[100] δεσμεύει 400 byte μνήμης.
- ❑ Ο τελεστής delete διαγράφει τα περιεχόμενα ενός τμήματος δυναμικά κατανεμημένης μνήμης η οποία, μετά την εφαρμογή του, παραμένει δεσμευμένη.
- ❑ Η C++ διαθέτει αυτόματο τρόπο αύξησης ή μείωσης ενός τμήματος μνήμης το οποίο έχει δεσμευτεί δυναμικά.

20.2

```
#include <iostream>
using namespace std;

int main()
{
   int i,ar,*pin;
   cout<<"Δώσε πλήθος:";
   cin>>ar;
   pin=new int[ar];
   for(i=0;i<ar;i++)
      cin>>pin[i];
   return 0;
}
```

Δεσμεύει τόσες θέσεις μνήμης τύπου **int** όσες ο αριθμός **ar** που δώσαμε.

Καταχωρίζει στις θέσεις μνήμης αριθμούς που ζητάει από το πληκτρολόγιο.

☞ Στον δείκτη pin καταχωρίζεται η διεύθυνση που επιστρέφει ο τελεστής new. Επομένως, ο δείκτης pin δείχνει στην αρχή του τμήματος μνήμης που δέσμευσε η new.

☞ Ο pin χρησιμοποιείται κανονικά ως πίνακας με βάση τη σχέση δεικτών και πινάκων.

20.3

Θα πρέπει να προστεθεί στον κώδικα της προηγούμενης άσκησης η παρακάτω πρόταση:

```
pin=resize(pin,ar*sizeof(int),2*ar*sizeof(int));
```

Η resize() είναι η παρακάτω συνάρτηση της οποίας η λειτουργία εξηγείται στο Κεφάλαιο 20 του βιβλίου.

```
int *resize(int *old, unsigned oldsize, unsigned newsize)
{
   int *tmp=new int[newsize];
   if (newsize<oldsize) oldsize=newsize;
   for (int i=0;i<oldsize;++i)
      tmp[i]=old[i];
   delete[] old;
   return tmp;
}
```

☞ Εδώ, η resize() δεσμεύει με δυναμικό τρόπο νέο διπλάσιο χώρο στη μνήμη, αντιγράφει τα δεδομένα από τον αρχικό χώρο στο νέο και τέλος αποδεσμεύει τον αρχικό χώρο μνήμης. Ο δείκτης που επιστρέφει (και δείχνει στο νέο μπλοκ μνήμης) καταχωρίζεται στην ίδια μεταβλητή δείκτη pin.

☞ Είναι πιθανόν (σχεδόν βέβαιο) το νέο μπλοκ μνήμης να βρίσκεται σε διαφορετική θέση από το αρχικό.

20.4

```
#include <iostream>
#include <string>
using namespace std;
class stoixeia
{
public:
   string onoma;
   string address;
   string thl;
   int ilikia;
};
int main()
{
   stoixeia *p;
   p=new stoixeia;          // δημιουργεί δυναμικά ένα αντικείμενο της κλάσης stoixeia
   p->onoma="Νίκος";        // καταχωρίζει στο αντικείμενο αυτό τα δικά μου στοιχεία
   p->address="Μυτιλήνη";
   p->thl="2251041688";
   p->ilikia=54;
   cout<<p->onoma<<endl;    // εμφανίζει τα στοιχεία που καταχωρίστηκαν
   cout<<p->address<<endl;
   cout<<p->thl<<endl;
   cout<<p->ilikia<<endl;
   delete p;                // αποδεσμεύει τον δυναμικά κατανεμημένο χώρο
   return 0;
}
```

20.5

```
#include <iostream>
#include <string>
using namespace std;
class stoixeia
{
public:
   string onoma;
   string address;
   string thl;
   int ilikia;
};
int main()
{
   int ar;
   stoixeia *pin;
   cout<<"Δώσε πλήθος μαθητών:";
   cin>>ar;
   pin = new stoixeia[ar];
   if (ar>=10)
   {
      cout<<"Δώσε όνομα:";
      cin>>pin[9].onoma;
      cout<<"Δώσε διεύθυνση:";
      cin>>pin[9].address;
      cout<<"Δώσε τηλέφωνο:";
      cin>>pin[9].thl;
      cout<<"Δώσε ηλικία:";
      cin>>pin[9].ilikia;
   }
   return 0;
}
```

Δεσμεύει τόσες θέσεις μνήμης για αντικείμενα **stoixeia** όσες ο αριθμός **ar** που δώσαμε.

Ζητάει και καταχωρίζει τα στοιχεία του μαθητή στη δέκατη θέση του πίνακα (η δέκατη θέση είναι η pin[9]).

20.6

```
#include <iostream>
#include <cstdlib>
using namespace std;
class arithmoi
{
   int *ar;
public:
   arithmoi();
   ~arithmoi();
   arithmoi(const arithmoi &a);
   arithmoi operator=(arithmoi op2);
   void fill_random();
   void display();
   void sort();
};
arithmoi::arithmoi()
{
   ar=new int[100];
}
arithmoi::~arithmoi()
{
   delete[] ar;
}
arithmoi::arithmoi(const arithmoi &a)
{
   int i;
   ar=new int[100];
   for (i=0;i<100;i++)
   ar[i]=a.ar[i];
}
arithmoi arithmoi::operator=(arithmoi op2)
{
   int i;
   for (i=0;i<100;i++)
      ar[i]=op2.ar[i];
   return *this;
}
void arithmoi::fill_random()
{
   int i;
   for (i=0;i<100;i++) ar[i]=rand();
}
void arithmoi::display()
{
   int i;
   cout<<"========"<<endl;
   for (i=0;i<100;i++) cout<<ar[i]<<'\t';
   cout<<endl;
}
void arithmoi::sort()
{
   int i,k,temp;
   for (i=1;i<100;i++)
   {
      for (k=99;k>=i;k--)
      {
         if (ar[k]<ar[k-1])
         {
            temp=ar[k];
            ar[k]=ar[k-1];
```

Ο δείκτης **ar** θα χρησιμοποιηθεί για να δείχνει στον δυναμικά δεσμευμένο χώρο μνήμης.

Η μέθοδος δόμησης της κλάσης δεσμεύει έναν χώρο για την αποθήκευση 100 ακέραιων. Στον δείκτη **ar** αποθηκεύεται η διεύθυνση του χώρου μνήμης.

Η μέθοδος αποδόμησης αποδεσμεύει τον χώρο που δεσμεύει δυναμικά ένα αντικείμενο της κλάσης.

Η μέθοδος δόμησης αντιγράφου της κλάσης αντιγράφει τους αριθμούς του πρωτότυπου αντικειμένου στο αντίγραφο.

Η μέθοδος υπερφόρτωσης του τελεστή ανάθεσης τιμής = αντιγράφει τους αριθμούς του δεξιού μέλους στο αριστερό.

```
            ar[k-1]=temp;
         }
      }
   }
}

int main()
{
   arithmoi a1;          // δημιουργεί ένα αντικείμενο a1 της κλάσης arithmoi
   a1.fill_random();     // συμπληρώνει το αντικείμενο a1 με 100 τυχαίους αριθμούς
   a1.display();         // εμφανίζει τους αριθμούς του αντικειμένου a1
   arithmoi a2=a1,a3;    // δημιουργεί ένα αντικείμενο a2 αντίγραφο του a1 και ένα άλλο a3 της κλάσης arithmoi
   a1.sort();            // ταξινομεί τους αριθμούς του αντικειμένου a1
   a2.display();         // εμφανίζει τους αριθμούς του αντικειμένου a3
   a3=a1;                // καταχωρίζει τα περιεχόμενα του αντικειμένου a1 στο αντικείμενο a3
   a3.display();         // εμφανίζει τους αριθμούς του αντικειμένου a3
   return 0;
}
```

20.7

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

int main()
{
   ifstream file;
   float *pin,sum=0,mo,temp;
   int i,k,plithos;
   file.open("times.txt");
   if (!file)
   {
      cout<<"Πρόβλημα στο αρχείο"<<endl;
      return 1;
   }
   file>>plithos;
   pin=new float[plithos];
   for (i=0;i<plithos;i++)
      file>>pin[i];
   file.close();
   for (i=0;i<plithos;i++)
      sum=sum+pin[i];
   mo=sum/plithos;
   for (i=1;i<plithos;i++)
      for (k=plithos-1;k>=i;k--)
      {
         if (pin[k]>pin[k-1])
         {
            temp=pin[k];
            pin[k]=pin[k-1];
            pin[k-1]=temp;
         }
      }
   cout<<"Ο μέσος όρος των τιμών είναι "<<mo<<endl;
   for (i=0;i<plithos;i++)
      cout<<pin[i]<<endl;
   delete[] pin;
   return 0;
}
```

Άνοιγμα του αρχείου

Δυναμική κατανομή μνήμης για το απαιτούμενο πλήθος δεδομένων. Ανάγνωση των αριθμών και καταχώρισή τους στον πίνακα **pin[]**.

Υπολογισμός αθροίσματος και μέσου όρου.

Φθίνουσα ταξινόμηση του πίνακα **pin[]** με τον αλγόριθμο *bubble sort*.

Εμφάνιση του μέσου όρου και των τιμών του πίνακα.

Αποδέσμευση του δυναμικά δεσμευμένου χώρου.

20.8

```
#include <iostream>
using namespace std;
int main()
{
   int i=0,ar=50,k,n,*pin,*pin2,sum=0;
   pin=new int[ar];
   if (pin==NULL)
   {
      cout <<"Δεν υπάρχει αρκετή μνήμη για δέσμευση"<<endl;
      return 1;
   }
   while (1)
   {
      cin>>pin[i];
      i++;
      if (i==ar)
      {
         n=ar*1.2;
         pin2=new int[n];
         if (pin2==NULL) break;
         for (k=0;k<ar;k++) pin2[k]=pin[k];
         delete[] pin;
         ar=ar*1.2;
         pin=pin2;
      }
   }
   for (k=0;k<ar;k++) sum=sum+pin[k];
   cout<<sum/ar<<endl;
   return 0;
}
```

Δεσμεύεται χώρος για 50 ακέραιους αριθμούς.

Ελέγχεται αν γέμισε ο χώρος των **ar** θέσεων.

Δεσμεύουμε χώρο 20% μεγαλύτερο από τον υπάρχοντα χώρο **ar** θέσεων. Στην περίπτωση που δεν υπάρχει ελεύθερη μνήμη, διακόπτεται η επαναληπτική διαδικασία.

Αντιγράφει τους αριθμούς σε αυτόν το νέο χώρο και αποδεσμεύει τον παλιό.

Αλλάζει την τιμή του μεγέθους στο νέο μέγεθος (αυξημένο κατά 20%). Ο δείκτης **pin** τώρα δείχνει στο νέο χώρο που δεσμεύτηκε.

Υπολογίζει το άθροισμα των αριθμών και τον μέσο όρο τους.

☞ Το πρόγραμμα αρχικά δεσμεύει χώρο για 20 ακέραιους αριθμούς. Κατόπιν ζητάει από τον χρήστη να πληκτρολογεί συνέχεια αριθμούς και τους καταχωρίζει στον χώρο που δέσμευσε. Τον χώρο τον αντιμετωπίζει ως έναν μονοδιάστατο πίνακα με όνομα pin.

☞ Μόλις γεμίσει ο αρχικός χώρος των 50 θέσεων, δεσμεύει δυναμικά χώρο 20% μεγαλύτερο (60 θέσεων), αντιγράφει τους αριθμούς σε αυτόν τον χώρο, αποδεσμεύει τον παλιό, και ορίζει τον δείκτη pin να δείχνει στο νέο χώρο που δεσμεύτηκε. Μόλις γεμίσει πάλι ο νέος χώρος, τον αυξάνει κατά 20% κ.ο.κ.

☞ Αυτό επαναλαμβάνεται μέχρι να μην υπάρχει ελεύθερη μνήμη για δέσμευση χώρου μεγαλύτερου κατά 20%.

☞ Τέλος, υπολογίζει και εμφανίζει τον μέσο όρο όλων των αριθμών που δόθηκαν.

Ασκήσεις Κεφαλαίου 21

21.1

- ❑ Οι λίστες και τα δυαδικά δένδρα δεσμεύουν συγκεκριμένο μέγεθος μνήμης.
- ☑ Σε μια δομή ουράς το πρώτο στοιχείο που προστίθεται στην ουρά είναι το πρώτο που φεύγει από την ουρά.
- ❑ Σε μια δομή στοίβας το πρώτο στοιχείο που προστίθεται στη στοίβα είναι το πρώτο που φεύγει από τη στοίβα.
- ☑ Σε μια απλά συνδεδεμένη λίστα δεν μπορούμε να εμφανίσουμε τα στοιχεία της λίστας με τη σειρά από το τελευταίο προς το πρώτο.
- ☑ Ένα δυαδικό δένδρο αναζήτησης διατηρεί τα δεδομένα διατεταγμένα ως προς το πεδίο-κλειδί του δυαδικού δένδρου.

21.2

```
#include <iostream>
#include <cstdlib>
using namespace std;

class node
{
public:
   int data;
   node *next;
}*list_head,*neos;

void add_node_to_list(int ar);

int main()
{
   int i,a[100];
   for (i=0;i<100;i++) a[i]=rand();
   list_head=NULL;
   for (i=0;i<100;i++)
   {
      add_node_to_list(a[i]);
   }
   return 0;
}
void add_node_to_list(int ar)
{
   neos = new node;
   neos->data=ar;
   neos->next = list_head;
   list_head=neos;
}
```

Βλέπε παράδειγμα δημιουργίας μιας απλά συνδεδεμένης λίστας στο Κεφάλαιο 21 του βιβλίου.

Συμπλήρωση του πίνακα **a** με τυχαίους αριθμούς.

Η αρχική τιμή του χειριστή της λίστας **list_head** ορίζεται ίση με NULL.

Προσθήκη των αριθμών του πίνακα στη συνδεδεμένη λίστα.

Δέσμευση μνήμης για ένα νέο κόμβο.

Καταχώριση του αριθμού στο πεδίο **data** του νέου κόμβου. Στο μέλος **next** καταχωρίζεται η διεύθυνση του τρέχοντα κόμβου κεφαλής της λίστας, ενώ στον χειριστή της λίστας **list_head** καταχωρίζεται η διεύθυνση του νέου κόμβου, ο οποίος είναι τώρα στην κορυφή της λίστας.

21.3

```
#include <iostream>
#include <cstdlib>
using namespace std;

class node
{
public:
   int data;
   node *next;
   node *previous;

}*list_head,*list_tail,*neos;

node *find_thesi(int d)
{
   node *p;
   p=list_head;
   while (p!=NULL)
   {
      if (d<=p->data)
         return p->previous;
      p=p->next;
   }
   return list_tail;
}
void add_node_to_list(node *thesi,int d)
{
   node *neos=new node;
   neos->data=d;
   if (list_head==NULL)
   {
      neos->next=NULL;
      neos->previous=NULL;
      list_head=neos;
      list_tail=neos;
   }
   else if (thesi==NULL)
   {
      neos->next=list_head;
      neos->previous=NULL;
      list_head->previous=neos;
      list_head=neos;
   }
   else if (thesi==list_tail)
   {
      neos->next=NULL;
      neos->previous=list_tail;
      list_tail->next=neos;
      list_tail=neos;
   }
   else
   {
      neos->next=thesi->next;
      neos->previous=thesi;
      thesi->next->previous=neos;
      thesi->next=neos;
   }
}
```

Το τμήμα δεδομένων αποτελείται από το μέλος **data**, στο οποίο καταχωρίζεται ο αριθμός. Για την υλοποίηση της ταξινομημένης λίστας, χρησιμοποιείται μια διπλά συνδεδεμένη λίστα.

Ελέγχονται τα δεδομένα κάθε κόμβου με τη σειρά. Μόλις εντοπιστεί η θέση στην οποία πρέπει να παρεμβληθεί ο νέος κόμβος, επιστρέφει ως τιμή τη διεύθυνση του κόμβου μετά από τον οποίο πρέπει να γίνει η παρεμβολή. Επιστρέφει τιμή NULL όταν ο κόμβος πρέπει να παρεμβληθεί στην αρχή, πριν από τον πρώτο κόμβο της λίστας.

Η συνάρτηση προσθέτει ένα νέο κόμβο με τιμή **d** μετά από τη θέση που δείχνει ο δείκτης **thesi**.

Αν η λίστα είναι κενή, ο νέος κόμβος είναι ο πρώτος και μοναδικός κόμβος της λίστας. Ενημερώνονται τόσο οι δείκτες του νέου κόμβου όσο και οι χειριστές της λίστας.

Αν ο νέος κόμβος πρέπει να τοποθετηθεί στην κορυφή της λίστας (thesi==NULL), ενημερώνονται οι δείκτες του νέου κόμβου, ο υπάρχων πρώτος κόμβος ώστε ο προηγούμενος να είναι πλέον ο νέος κόμβος, καθώς και ο χειριστής της κεφαλής της λίστας.

Αν ο νέος κόμβος πρέπει να τοποθετηθεί στο τέλος της λίστας (thesi==list_tail), ενημερώνονται οι δείκτες του νέου κόμβου, ο υπάρχων τελευταίος κόμβος ώστε ο επόμενος να είναι πλέον ο νέος κόμβος, καθώς και ο χειριστής της ουράς της λίστας.

Αν ο νέος κόμβος πρέπει να τοποθετηθεί ενδιάμεσα, μετά από τον κόμβο στον οποίο δείχνει ο δείκτης **thesi**, ενημερώνονται οι δείκτες του νέου κόμβου, ο κόμβος πριν από το σημείο παρεμβολής ώστε ο επόμενος να είναι πλέον ο νέος κόμβος, καθώς και ο κόμβος μετά το σημείο παρεμβολής ώστε ο προηγούμενος να είναι ο νέος κόμβος.

```
void display_all()
{
   node *p;
   p=list_head;
   while (p!=NULL)
   {
      cout<<p->data<<endl;
      p=p->next;
   }
}
int main()
{
   int a[100],i;
   node *pos;
   list_head=list_tail=NULL;
   for (i=0;i<100;i++) a[i]=rand();
   for (i=0;i<100;i++)
   {
      pos=find_thesi(a[i]);
      add_node_to_list(pos,a[i]);
   }
   display_all();
   return 0;
}
```

Εμφανίζει όλα τα δεδομένα της ταξινομημένης λίστας.

Συμπληρώνει τον πίνακα **a** με τυχαίους αριθμούς.

Εντοπίζει τη θέση στη λίστα στην οποία πρέπει να προστεθεί ο αριθμός που βρίσκεται στη θέση **a[i]** του πίνακα.

Προσθέτει, στη σωστή θέση στη λίστα, έναν κόμβο με περιεχόμενο τον αριθμό της θέσης **a[i]** του πίνακα.

21.4

```
float calculate_mo(node *p)
{
   float sum=0,mo;
   int plithos=0;
   while (p!=NULL)
   {
      sum=sum+p->varos;
      p=p->next;
      plithos++;
   }
   mo=sum/plithos;
   return mo;
}
```

Η μεταβλητή **sum** θα χρησιμοποιηθεί για την αποθήκευση του αθροίσματος των βαρών. Η μεταβλητή **plithos** θα χρησιμοποιηθεί για την καταμέτρηση των κόμβων της λίστας.

Επισκεπτόμαστε έναν προς έναν τους κόμβους της λίστας και προσθέτουμε στη μεταβλητή **sum** το εκάστοτε βάρος (p->varos). Κάθε φορά η μεταβλητή **plithos** αυξάνεται κατά 1.

Η συνάρτηση επιστρέφει ως τιμή τον μέσο όρο (sum/plithos) των βαρών.

21.5

```
#include <iostream>
#include <conio.h>
using namespace std;
class mystack
{
   class node
   {
      public:
      string onoma;
      int  ilikia;
      node *next;
   };
   node *list_head;
public:
   void pop();
   bool push(string o, int il);
   bool isempty();
   void gettop();
   void display();
```

```
   mystack();
   ~mystack();
};

mystack::mystack()
{
   list_head=NULL;
}

mystack::~mystack()
{
   node *temp;
   while (list_head!=NULL)
   {
      temp = list_head->next;
      delete list_head;
      list_head=temp;
   }
   cout<<"Καταστροφή στοίβας"<<endl;
}

bool mystack::push(string o, int il)
{
   node *neos;
   neos=new(nothrow) node;
   if (neos==NULL) return false;
   neos->onoma=o;
   neos->ilikia=il;
   neos->next=list_head;
   list_head=neos;
   return true;
}

void mystack::pop()
{
   node *temp;
   if (isempty())
   {
      cout<<"Κενή στοίβα"<<endl;
      return;
   }
   temp = list_head->next;
   delete list_head;
   list_head=temp;
}

bool mystack::isempty()
{
   if (list_head==NULL) return true;
   return false;
}

void mystack::gettop()
{
   if (isempty())
      cout<<"Κενή στοίβα"<<endl;
   else
      cout<<"Όνομα:"<<list_head->onoma<<" Ηλικία:"<<list_head->ilikia<<endl;
}
```

```
void mystack::display()
{
   node *p=list_head;
   if (isempty())
   {
      cout<<"Κενή στοίβα"<<endl;
      return;
   }
   while (p!=NULL)
   {
      cout<<"Όνομα:"<<p->onoma<<" Ηλικία:"<<p->ilikia<<endl;
      p=p->next;
   }
}

int main()
{
   char ch;
   string on;
   int il;
   mystack stack1;
   while (ch!='0')
   {
      cout<<"0->Εξοδος 1->Ωθηση 2->Απώθηση 3->Εμφάνιση κορυφής 4->Εμφάνιση όλων:";
      cin>>ch;
      switch (ch)
      {
         case '0':
            break;
         case '1':
            cout<<"Όνομα:";
            cin>>on;
            cout<<"Ηλικία:";
            cin>>il;
            if (stack1.push(on,il)==0)
               cout<<"Δεν υπάρχει αρκετή μνήμη"<<endl;
            break;
         case '2':
            stack1.pop();
            break;
         case '3':
            stack1.gettop();
            break;
         case '4':
            stack1.display();
            break;
         default:
            cout<<"Λάθος πλήκτρο"<<endl;
            break;
      }
   }
   return 0;
}
```

21.6

```
.....
ofstream myfile;
node *p;
myfile.open("ONOMATA");
if (!myfile)
{
```

Ανοίγει το αρχείο «ONOMATA» για εγγραφή. Ελέγχει αν η διαδικασία ανοίγματος ήταν επιτυχής.

```
   cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
   exit(2);
}
p=list_head;
while (p!=NULL)
{
   myfile << p->onoma <<endl;
   p=p->next;
}
myfile.close();
.....
```

Επισκέπτεται έναν προς έναν τους κόμβους της λίστας και καταχωρίζει στο αρχείο το μέλος του ονόματος του κάθε κόμβου.

21.7

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

class node
{
public:
   string data;
   node *next;
   node *previous;

}*list_head,*list_tail,*neos;

node *find_thesi(string d)
{
   node *p;
   p=list_head;
   while (p!=NULL)
   {
      if (d<=p->data)
         return p->previous;
      p=p->next;
   }
   return list_tail;
}

void add_node_to_list(node *thesi,string d)
{
   node *neos=new node;
   neos->data=d;
   if (list_head==NULL)
   {
      neos->next=NULL;
      neos->previous=NULL;
      list_head=neos;
      list_tail=neos;
   }
   else if (thesi==NULL)
   {
      neos->next=list_head;
      neos->previous=NULL;
      list_head->previous=neos;
      list_head=neos;
   }
   else if (thesi==list_tail)
   {
      neos->next=NULL;
```

Χρησιμοποιήθηκε ο κώδικας του προγράμματος **21_ordered_list.cpp** με τις απαραίτητες αλλαγές στη συνάρτηση **main()**.

```
      neos->previous=list_tail;
      list_tail->next=neos;
      list_tail=neos;
   }
   else
   {
      neos->next=thesi->next;
      neos->previous=thesi;
      thesi->next->previous=neos;
      thesi->next=neos;
   }
}
void display_all()
{
   node *p;
   p=list_head;
   while(p!=NULL)
   {
      cout<<p->data<<endl;
      p=p->next;
   }
}
int main()
{
   ifstream myfile;
   string onom;
   node *pos;
   list_head=list_tail=NULL;
   myfile.open("ONOMATA");
   if (!myfile)
   {
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
      return 1;
   }
   while (myfile)
   {
      myfile>>onom;
      if(myfile)
      {
         pos=find_thesi(onom);
         add_node_to_list(pos,onom);
      }
   }
   myfile.close();
   display_all();
   return 0;
}
```

Ανοίγει το αρχείο «ONOMATA» για ανάγνωση.

Διαβάζει ένα όνομα από το αρχείο και το προσθέτει στη διατεταγμένη λίστα.

Εμφανίζει τα στοιχεία όλων των κόμβων της λίστας.

21.8

```
.....
int main()
{
   myqueue filoi;
   filoi.enqueue("Νίκος");
   filoi.enqueue("Άκης");
   filoi.enqueue("Μαρία");
   filoi.enqueue("Γιάννης");
   filoi.enqueue("Αφροδίτη");
   filoi.display();
   return 0;
}
```

21.9

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

class mybtree
{
   class node
   {
   public:
      string data;
      node *left;
      node *right;
   };
   node *root;
   node *newnode(string d)
   {
      node *neos;
      neos = new(nothrow) node;
      neos->data = d;
      neos->left = NULL;
      neos->right = NULL;
      return neos;
   }
public:
   node *find(string key)
   {
      node *current;
      current=root;
      while(current->data != key)
      {
         if (key < current->data)
            current = current->left;
         else
            current = current->right;
         if (current == NULL)
            return NULL;
      }
      return current;
   }
   // εντοπίζει τον τελευταίο αριστερό κόμβο του υποδένδρου rt
   node *find_left_most(node *rt)
   {
      if(rt==NULL) return NULL;
      while(rt->left!=NULL)
      {
         rt=rt->left;
      }
      return rt;
   }
   // εντοπίζει τον τελευταίο δεξιό κόμβο του υποδένδρου rt
   node *find_right_most(node *rt)
   {
      if (rt==NULL) return NULL;
      while (rt->right!=NULL)
      {
         rt=rt->right;
      }
      return rt;
   }
```

Χρησιμοποιήθηκε η κλάση **mybtree** του προγράμματος **21_btree_object.cpp** με τις απαραίτητες αλλαγές: Η κλάση **node** προσαρμόστηκε σύμφωνα με την άσκηση, και το μέλος **data** είναι πλέον τύπου **string**.

```
   void display(node *ptr);
   void display_tree();
   void post_order_delete(node *ptr);
   bool insert(string d);
   mybtree();
   ~mybtree();
};

mybtree::mybtree()
{
   root=NULL;
}

mybtree::~mybtree()
{
   post_order_delete(root);
   root=NULL;
   cout<<"Το δένδρο καταστράφηκε"<<endl;
}

void mybtree::post_order_delete(node *ptr)
{
   if (ptr == NULL) return;
   post_order_delete(ptr->left);
   post_order_delete(ptr->right);
   delete ptr;
}

void mybtree::display_tree()
{
   display(root);
}

void mybtree::display(node *ptr)
{
   if (ptr == NULL) return;
   display(ptr->left);
   cout << ptr->data<<endl;
   display(ptr->right);
}

bool mybtree::insert(string d)
{
   node *next,*current,*ptr;
   bool isleft;
   next=current=root;
   ptr=newnode(d);
   if (ptr==NULL) return false; // δεν υπάρχει αρκετή μνήμη
   if (root == NULL)
   {
      root=ptr;
      return true;
   }
   while (1)
   {
      if (d<current->data)
      {
         next = current->left;
         isleft=true;
      }
      else
```

```
      {
         next = current->right;
         isleft=false;
      }
      if (next == NULL)
      {
         if (isleft)
            current->left=ptr;
         else
            current->right=ptr;
         return true;
      }
      current=next;
   }
}

int main()
{
   mybtree tree;
   ifstream myfile;
   string onom;
   myfile.open("ONOMATA");
   if (!myfile)
   {
      cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
      return 1;
   }
   while (myfile)
   {
      myfile>>onom;
      tree.insert(onom);
   }
   myfile.close();
   tree.display_tree();
   return 0;
}
```

Δημιουργείται ένα δυαδικό δένδρο αναζήτησης (ΔΔΑ) με όνομα **tree**.

Ανοίγει το αρχείο «ONOMATA» για ανάγνωση.

Διαβάζει ένα όνομα από το αρχείο και το προσθέτει στο ΔΔΑ.

Εμφανίζει τα στοιχεία όλων των κόμβων που ΔΔΑ.

21.10

```
.....
int main()
{
   int i;
   mybtree arithmoi;
   for (i=1;i<=100;i++)
      arithmoi.insert(rand());
   arithmoi.display_tree();
   return 0;
}
```

21.11

Τη μέγιστη ηλικία

```
int max_data(node *rt)
{
   node *max_node;
   max_node= find_right_most(rt);
   if(max_node==NULL)
   {
      cout << "Το δένδρο είναι άδειο"<<endl;
      exit(1);
   }
```

Στον δείκτη **max_node** καταχωρίζεται η διεύθυνση του τελευταίου δεξιού κόμβου του ΔΔΑ.

Στην περίπτωση που το ΔΔΑ είναι άδειο, το πρόγραμμα τερματίζεται με μήνυμα λάθους.

```
   else
      return max_node->ilikia;
}
node *find_right_most(node *rt)
{
   node *current;
   if(rt==NULL) return NULL;
   while(rt->right!=NULL)
   {
      rt=rt->right;
   }
   return rt;
}
```

Επιστρέφει ως τιμή το πεδίο **ilikia** του τελευταίου δεξιού κόμβου.

Εντοπίζει τον τελευταίο δεξιό κόμβο του δυαδικού δένδρου αναζήτησης. Ο κόμβος αυτός περιέχει τη μεγαλύτερη τιμή κλειδιού.
Επιστρέφει έναν δείκτη σε αυτόν τον κόμβο. Στη περίπτωση που το ΔΔΑ είναι άδειο, επιστρέφει τιμή NULL.

☞ Εφόσον το πεδίο της ηλικίας είναι το πεδίο-κλειδί, ο κόμβος με τη μέγιστη ηλικία είναι ο τελευταίος δεξιός κόμβος του δυαδικού δένδρου. Η συνάρτηση max_data() χρησιμοποιεί τη find_right_most() για να εντοπίσει τον τελευταίο δεξιό κόμβο (δείτε την ενότητα «Υλοποίηση της δομής δυαδικού δένδρου» του βιβλίου). Επιστρέφει την τιμή του πεδίου ilikia του κόμβου αυτού. Αν το ΔΔΑ είναι άδειο, τερματίζεται με ένα μήνυμα λάθους.

Την ελάχιστη ηλικία

```
int min_data(node *rt)
{
   node *min_node;
   min_node= find_left_most(rt);
   if(min_node==NULL)
   {
      cout << "Το δένδρο είναι άδειο"<<endl;
      exit(1);
   }
   else
      return min_node->ilikia;
}
node *find_left_most(node *rt)
{
   node *current;
   if(rt==NULL) return NULL;
   while(rt->left!=NULL)
   {
      rt=rt->left;
   }
   return rt;
}
```

Στον δείκτη **min_node** καταχωρίζεται η διεύθυνση του τελευταίου αριστερού κόμβου του ΔΔΑ.

Αν το ΔΔΑ είναι άδειο, τερματίζεται με μήνυμα λάθους.

Επιστρέφει ως τιμή το πεδίο **Ilikia** του τελευταίου αριστερού κόμβου.

Εντοπίζει τον τελευταίο αριστερό κόμβο του δυαδικού δένδρου αναζήτησης. Ο κόμβος αυτός περιέχει τη μικρότερη τιμή κλειδιού.
Επιστρέφει έναν δείκτη σε αυτόν τον κόμβο. Αν το ΔΔΑ είναι άδειο, επιστρέφει τιμή NULL.

☞ Εφόσον το πεδίο της ηλικίας είναι το πεδίο-κλειδί, ο κόμβος με την ελάχιστη ηλικία είναι ο τελευταίος αριστερός κόμβος του δυαδικού δένδρου αναζήτησης (ΔΔΑ). Η συνάρτηση min_data() χρησιμοποιεί τη find_left_most() για να εντοπίσει τον τελευταίο αριστερά κόμβο (δείτε την ενότητα «Υλοποίηση της δομής δυαδικού δένδρου»). Επιστρέφει την τιμή του πεδίου ilikia του κόμβου αυτού. Αν το ΔΔΑ είναι άδειο, εμφανίζει μήνυμα λάθους.

Τον μέσο όρο των ηλικιών

```
float mo(node *rt)
{
   if(count(rt)!=0)
      return sum(rt)/count(rt);
   else
   {
      cout << "Το δένδρο είναι άδειο\n";
      exit(1);
   }
}
```

Αν το ΔΔΑ έχει τουλάχιστον έναν κόμβο, υπολογίζει και επιστρέφει τον μέσο όρο. Σύνολο_ηλικιών/Πλήθος_κόμβων.

Στην περίπτωση που το ΔΔΑ είναι άδειο, τερματίζεται με μήνυμα λάθους.

```
int count(node *ptr)
{
   if (ptr == NULL) return 0;
   return 1+count(ptr->left)+count(ptr->right);
}
```

Η συνάρτηση **count()** χρησιμοποιεί αναδρομική διαδικασία για να υπολογίσει το πλήθος των κόμβων του ΔΔΑ.

```
float sum(node *ptr)
{
   if (ptr == NULL) return 0;
   return ptr->ilikia+sum(ptr->left)+sum(ptr->right);
}
```

Η συνάρτηση **sum()** χρησιμοποιεί αναδρομική διαδικασία για να υπολογίσει το συνολικό άθροισμα του πεδίου **ilikia** όλων των κόμβων του ΔΔΑ.

☞ Η συνάρτηση mo() καλεί την sum() για να υπολογίσει το συνολικό άθροισμα των ηλικιών των κόμβων και την count() για να υπολογίσει το πλήθος των κόμβων. Ο μέσος όρος των ηλικιών υπολογίζεται από τον τύπο Συνολικό_άθροισμα/πλήθος.

21.12

```
.....
int main()
{
   mybtree tree;
   string onom;
   while (1)
   {
      getline(cin,onom);
      if (onom=="") break;
      tree.insert(onom);
   }
   tree.display_tree();
   return 0;
}
```

Χρησιμοποιήθηκε η κλάση **mybtree** από την Άσκηση 21.9

Προσθέτει το όνομα που διάβασε στο δένδρο **tree**.

21.13

```
void list_to_tree(node_list *lh, mybtree *tr)
{
   while(lh!=NULL)
   {
      tr->insert(lh->num);
      lh=lh->next;
   }
}
```

Καλείται η μέθοδος **insert()** του αντικειμένου δυαδικού δένδρου αναζήτησης, στο οποίο δείχνει ο δείκτης **tr**. Στο δένδρο καταχωρίζονται κόμβοι με τιμή τους αριθμούς που υπάρχουν στη λίστα όπου δείχνει ο δείκτης **lh**. Γίνεται διάσχιση όλης της λίστας.

21.14

```
#include <iostream>
#include <string>
using namespace std;
class mybtree
{
   class node
   {
   public:
      string onoma;
      int ilikia;
      float varos;
      float ypsos;
      node *left;
      node *right;
   };
```

Χρησιμοποιήθηκε η κλάση **mybtree** του προγράμματος **21_btree_object.cpp** με τις απαραίτητες αλλαγές: Η κλάση **node** προσαρμόστηκε σύμφωνα με την άσκηση. Ως κλειδί του ΔΔΑ χρησιμοποιήθηκε το πεδίο **ilikia** και διαγράφηκαν οι μέθοδοι που δεν χρειάζονται (π.χ. διαγραφή κόμβου, αναζήτηση κόμβου).

Η κλάση **node** περιέχει τα κατάλληλα μέλη που απαιτούνται από την άσκηση

```
   node *root;
   node *newnode(string o,int il, float var, float yps)
   {
      node *neos;
      neos = new(nothrow) node;
      neos->onoma = o;
      neos->ilikia = il;
      neos->varos = var;
      neos->ypsos = yps;
      neos->left = NULL;
      neos->right = NULL;
      return neos;
   }
public:
   void display(node *ptr);
   void display_tree();
   bool insert(string o,int il, float var, float yps);
   mybtree();
};
mybtree::mybtree()
{
   root=NULL;
}
void mybtree::display_tree()
{
   display(root);
}
void mybtree::display(node *ptr)
{
   if (ptr == NULL) return;
   display(ptr->left);
   cout << ptr->onoma<<endl;
   cout << ptr->ilikia<<endl;
   cout << ptr->varos<<endl;
   cout << ptr->ypsos<<endl;
   cout << "================"<<endl;
   display(ptr->right);
}

bool mybtree::insert(string o,int il, float var, float yps)
{
   node *next,*current,*ptr;
   bool isleft;
   next=current=root;
   ptr=newnode(o,il,var,yps);
   if (ptr==NULL) return false; // δεν υπάρχει αρκετή μνήμη
   if (root == NULL)
   {
      root=ptr;
      return true;
   }
   while (1)
   {
      if (il < current->ilikia)
      {
         next = current->left;
         isleft=true;
      }
      else
      {
         next = current->right;
```

Στη μέθοδο **newnode()** μεταβιβάζονται οι τιμές των μελών του νέου κόμβου.

Η μέθοδος **display()** εμφανίζει τα στοιχεία των μελών του κόμβου.

Στη μέθοδο **insert()** μεταβιβάζονται επίσης οι τιμές των μελών του νέου κόμβου.

Χρησιμοποιείται ως κλειδί του ΔΔΑ το πεδίο της ηλικίας.

```
            isleft=false;
         }
         if (next == NULL)
         {
            if (isleft)
               current->left=ptr;
            else
               current->right=ptr;
            return true;
         }
         current=next;
      }
   }

   int main()
   {
      int i;
      string onom;
      int il;
      float v,y;
      mybtree btree1;
      for (i=1;i<=10;i++)
      {
         cout << "Όνομα:";
         cin >> onom;
         cout << "Ηλικία:";
         cin >> il;
         cout << "Βάρος:";
         cin >> v;
         cout << "Ύψος:";
         cin >> y;
         btree1.insert(onom,il,v,y);
      }
      btree1.display_tree();
      return 0;
   }
```

Δημιουργείται ένα ΔΔΑ με όνομα **btree1**.

Ο χρήστης πληκτρολογεί τα στοιχεία δέκα ατόμων.

Τα στοιχεία του κάθε ατόμου προστίθενται ως κόμβος του ΔΔΑ.

Εμφανίζονται τα στοιχεία όλων των κόμβων του ΔΔΑ.

Ασκήσεις Κεφαλαίου 22

22.1

- ❑ Τα πρότυπα συναρτήσεων μπορούν να έχουν μόνο μία παράμετρο.
- ☑ Η έκδοση ενός προτύπου συνάρτησης που θα μεταγλωττιστεί αποφασίζεται κατά το στάδιο της μεταγλώττισης.
- ☑ Τα πρότυπα συναρτήσεων μπορούν να υπερφορτώνονται.
- ❑ Τα πρότυπα κλάσεων δεν μπορούν να έχουν περισσότερες από μία παραμέτρους.
- ☑ Τα πρότυπα κλάσεων, όπως και τα πρότυπα συναρτήσεων, επιτρέπουν την επαναχρησιμοποίηση κώδικα.

22.2

```
template <class T>
T myabs(T a)
{
   if (a<0)
      return (-a);
   else
      return a;
}

int main()
{
   float f1=-23.4;
   int i1=156;
   cout<<myabs(f1)<<endl;
   cout<<myabs(i1)<<endl;
   return 0;
}
```

Η **T** ορίζεται ως παράμετρος προτύπου

Στην πρώτη κλήση της **myabs()**, στην παράμετρο **T** μεταβιβάζεται ο τύπος **int** ενώ στη δεύτερη ο τύπος **float**.

22.3

```
template <class T>
void swap(T &x, T &y)
{
   T temp;
   temp=x;
   x=y;
   y=temp;
}
```

Η **T** ορίζεται ως παράμετρος προτύπου

22.4

```
template <class T>
int find(T pin[], int plithos, T timi)
{
   int i,thesi=-1;
   for (i=0;i<plithos;i++)
   {
      if (pin[i]==timi)
      {
         thesi=i;
         break;
      }
   }
   return thesi;
}
```

Στην περίπτωση που η συνάρτηση κληθεί για να εντοπίσει ένα αντικείμενο προσαρμοσμένης κλάσης σε έναν πίνακα αυτής της κλάσης, τότε για αυτή την κλάση θα πρέπει να έχει υπερφορτωθεί ο τελεστής σύγκρισης ==.

22.5

group<> g1

*Δημιουργεί ένα αντικείμενο **g1** με δύο μέλη. Το πρώτο είναι ένας πίνακας τύπου **float** 5 θέσεων, και το δεύτερο είναι ένας πίνακας τύπου **int** 10 θέσεων.*

group<**int**,**float**> g2

*Δημιουργεί ένα αντικείμενο **g2** με δύο μέλη. Το πρώτο είναι ένας πίνακας τύπου **int** 5 θέσεων, και το δεύτερο είναι ένας πίνακας τύπου **float** 10 θέσεων.*

group<**double**,**float**,20,30> g3

*Δημιουργεί ένα αντικείμενο **g3** με δύο μέλη. Το πρώτο είναι ένας πίνακας τύπου **double** 20 θέσεων, και το δεύτερο είναι ένας πίνακας τύπου **float**, 30 θέσεων.*

22.6

```
#include <iostream>
using namespace std;
template<class T1, class T2, int num>
class group
{
public:
   T1 obj1[num];
   T2 obj2[num];
   int theseis;
   group() {theseis=num;}
   T1 sum1();
   T2 sum2();
};
template<class T1, class T2, int num>
T1 group<T1,T2,num>::sum1()
{
   int i;
   T1 s=0;
   for (i=0;i<theseis;i++) s=s+obj1[i];
   return s;
}
template<class T1, class T2, int num>
T2 group<T1,T2,num>::sum2()
{
   int i;
   T2 s=0;
   for (i=0;i<theseis;i++) s=s+obj2[i];
   return s;
}
// Η παρακάτω συνάρτηση main() δείχνει τη χρήση των
// δύο μεθόδων sum1() και sum2()
int main()
{
   int i;
   group<int,int,10> g1;
   group<float,float,20> g2;
   // Συμπλήρωση του πίνακα obj1[] του αντικειμένου g1 με ακέραιους αριθμούς
   for (i=0;i<g1.theseis;i++) g1.obj1[i]=i;
   // Συμπλήρωση του πίνακα obj2[] του αντικειμένου g2 με δεκαδικούς αριθμούς
   for (i=0;i<g2.theseis;i++) g2.obj2[i]=i*i/4.0;
   // Εμφάνιση των περιεχομένων των δύο πινάκων
   for (i=0;i<g1.theseis;i++) cout<<g1.obj1[i]<<endl;
   cout<<"------------------"<<endl;
   for (i=0;i<g2.theseis;i++) cout<<g2.obj2[i]<<endl;
   cout << g1.sum1()<<endl;
   cout << g2.sum2()<<endl;
   return 0;
}
```

Δήλωση των δύο μεθόδων **sum1()** και **sum2()**.

Ορισμός της μεθόδου **sum1()**. Παρατηρούμε ότι η μέθοδος δηλώνεται ως πρότυπο συνάρτησης. Η μέθοδος επιστρέφει τιμή τύπου **T1**.

Υπολογίζεται το άθροισμα των στοιχείων του πίνακα.

Δημιουργείται ένα αντικείμενο **g1** με μέλη δύο πίνακες τύπου **int** 10 θέσεων.

Δημιουργείται ένα αντικείμενο **g2** με μέλη δύο πίνακες τύπου **float** 20 θέσεων.

Εμφανίζεται το άθροισμα των στοιχείων του πίνακα **obj1[]** του αντικειμένου **g1** και το άθροισμα των στοιχείων του πίνακα **obj2[]** του αντικειμένου **g2**.

- Τι εργασία διεκπεραιώνει η μέθοδος δόμησης της κλάσης;

  Η μέθοδος δόμησης δημιουργεί ένα νέο αντικείμενο της κλάσης και καταχωρίζει στη μεταβλητή-μέλος theseis την τιμή της άτυπης παραμέτρου num (δηλαδή το πλήθος θέσεων των πινάκων).

- Σκεφτείτε σε ποια περίπτωση οι μέθοδοι sum1() και sum2() δεν θα μεταγλωττιστούν σωστά;

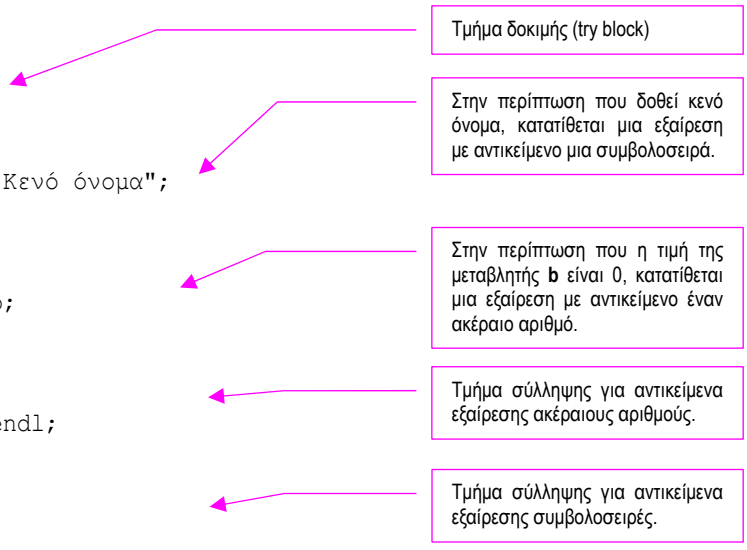
  Στην περίπτωση που δημιουργηθούν αντικείμενα της κλάσης group στα οποία δεν χρησιμοποιούνται βασικοί τύποι δεδομένων αλλά προσαρμοσμένες κλάσεις, π.χ. group<rectangle, triangle, 20>. Στην περίπτωση αυτή, οι μέθοδοι sum1() και sum2() δεν θα μεταγλωττιστούν, εκτός αν έχει υπερφορτωθεί ο τελεστής + (τον οποίο χρησιμοποιούν) για αυτές τις προσαρμοσμένες κλάσεις.

22.7

- ❑ Το τμήμα throw συλλαμβάνει μια εξαίρεση.
- ❑ Ένα τμήμα σύλληψης catch(…) συλλαμβάνει την πρώτη εξαίρεση που θα δημιουργηθεί.
- ☑ Για να μπορεί να συλληφθεί μια εξαίρεση πρέπει να κατατεθεί μέσα από ένα τμήμα δοκιμής try.
- ☑ Όλες οι εξαιρέσεις που προκύπτουν στις συναρτήσεις ενός προγράμματος, με τον κατάλληλο σχεδιασμό μπορούν να συλλαμβάνονται από τμήματα σύλληψης στη συνάρτηση main().
- ❑ Μια εξαίρεση που δεν συλλαμβάνεται οδηγεί στον τερματισμό του προγράμματος.
- ☑ Η C++ διαθέτει έτοιμες κλάσεις εξαιρέσεων.

22.8

```
void myfunc()
{
   string onom;
   int a,b;
   try
   {
      cout<<"Δώσε όνομα:";
      cin>>onom;
      if (onom.size()==0) throw "Κενό όνομα";
      cout<<"Δώσε δύο αριθμούς:";
      cin >> a >> b;
      if (b==0) throw b;
      cout << onom << " " <<  a/b;
   }
   catch(int n)
   {
      cout<<"Παρονομαστής "<<n<<endl;
   }
   catch(const char *mes)
   {
      cout<<mes<<endl;
   }
}
```



22.9

```
#include <iostream>
#include <string>
using namespace std;
class keno_onoma
{

};
class mideniki_timi
{

};
void myfunc()
{
   string onom;
   int a,b;
   cout<<"Δώσε όνομα:";
   cin>>onom;
   if (onom.size()==0) throw keno_onoma();
   cout<<"Δώσε δύο αριθμούς:";
   cin >> a >> b;
   if (b==0) throw mideniki_timi();
   cout << onom << " " << a/b;
}

int main()
{
   try
   {
      myfunc();
   }
   catch(mideniki_timi)
   {
      cout<<"Μηδενική τιμή "<<endl;
   }
   catch(keno_onoma)
   {
      cout<<"Κενό όνομα"<<endl;
   }
   return 0;
}
```

Ορισμός των κλάσεων που θα χρησιμοποιηθούν για τη διαχείριση εξαιρέσεων. Προσέξτε ότι το σώμα των κλάσεων είναι κενό.

Στη συνάρτηση **myfunc()** δεν υπάρχει τμήμα δοκιμής ούτε τμήματα σύλληψης. Οι εξαιρέσεις που δημιουργούνται μέσα από τη συνάρτηση συλλαμβάνονται από κώδικα προηγούμενων επιπέδων.

Η κλήση της συνάρτησης **myfunc()** γίνεται μέσα από ένα τμήμα δοκιμής. Οι εξαιρέσεις που δημιουργούνται μέσα στη συνάρτηση συλλαμβάνονται από τα τμήματα σύλληψης του παρόντος τμήματος δοκιμής.

22.10

```
.....
int main()
{
   bool ok=false;
   while (!ok)
   {
      try
      {
         myfunc();
         ok=true;
      }
      catch(mideniki_timi)
      {
         cout<<"Μηδενική τιμή "<<endl;
      }
      catch(keno_onoma)
```

Η κλήση της συνάρτησης **myfunc()** επαναλαμβάνεται μέχρι να μην προκαλείται εξαίρεση.

Στην περίπτωση που δεν προκύψει εξαίρεση, στη μεταβλητή **ok** θα καταχωριστεί τιμή **true**.

```
        {
           cout<<"Κενό όνομα"<<endl;
        }
     }
     return 0;
}
```

22.11

```
#include <iostream>
#include <string>
using namespace std;

class diairesi_me_miden
{
   string message;
public:
   diairesi_me_miden()
   {
      message="Προσπάθεια διαίρεσης με μηδέν";
   }
   string ti_egine()
   {
      return message;
   }
};

// Η συνάρτηση diairesi() επιστρέφει το πηλίκο των δύο παραμέτρων της
double diairesi(double ar1, double ar2)
{
   if (ar2==0)
      throw(diairesi_me_miden());
   return ar1/ar2;
}

int main()
{
   double a,b,pil;
   cin>>a>>b;
   try
   {
      pil=diairesi(a,b);
      cout<<"Το πηλίκο της διαίρεσης του "<<a<<" δια "<<b<<" είναι "<<pil<<endl;
   }
   catch (diairesi_me_miden exobj)
   {
      cout<<"Δημιουργήθηκε η εξαίρεση -> "<<exobj.ti_egine()<<endl;
   }
   return 0;
}
```

Η κλάση **diairesi_me_miden** χρησιμοποιείται ως *κλάση εξαίρεσης* για τη διαχείριση εξαιρέσεων στην περίπτωση προσπάθειας διαίρεσης με το 0.

Η μέθοδος **ti_egine()** επιστρέφει το μήνυμα ενός αντικειμένου εξαίρεσης αυτής της κλάσης!

Στην περίπτωση αρνητικού αριθμού δημιουργεί μια εξαίρεση, καταθέτοντας ένα αντικείμενο της κλάσης **diairesi_me_miden**.

Η συνάρτηση **diairesi()** καλείται μέσα από ένα τμήμα δοκιμής (try block).

Το τμήμα σύλληψης συλλαμβάνει εξαιρέσεις αντικειμένων κλάσης **diairesi_me_miden** και εμφανίζει το μήνυμα της εξαίρεσης!

22.12

- ☑ Οι χώροι ονομάτων καθορίζουν διαφορετικούς χώρους εμβέλειας.
- ☑ Η καθιερωμένη βιβλιοθήκη της C++ ορίζεται στον χώρο ονομάτων std.
- ❑ Μια πρόταση της μορφής nik::cout<<100; θα μπορούσε να εμφανίζει το άθροισμα των αριθμών από το 1 μέχρι το 100!
- ❑ Σε διαφορετικούς χώρους ονομάτων δεν μπορούν να ορίζονται κλάσεις με το ίδιο όνομα.
- ❑ Δεν επιτρέπεται η χρήση της εντολής namespace για τον ίδιο χώρο ονομάτων περισσότερες από μία φορές.

22.13

```
#include <iostream>
namespace first {int a=20,b,d=44;}
namespace second {int d = 3;}
namespace first {int k=20;}
using namespace first;

int main()
{
   int a = 1;
   a=a+5;
   std::cout << a << std::endl;
   std::cout << d << " " << k << std::endl;
   std::cout << second::d << std::endl;
   std::cout << first::a << std::endl;
   return 0;
}
```

```
6
44 20
3
20
```

Αναφέρεται στη μεταβλητή **a** της **main()**.

☞ Από τη στιγμή που έχουμε ορίσει ως προκαθορισμένο χώρο ονομάτων τον first, η αναφορά στη μεταβλητή d χωρίς πρόθεμα αναφέρεται στη μεταβλητή d που δηλώθηκε στον χώρο ονομάτων first και έχει τιμή 44.

22.14

```
int main()
{
   int res;
   res=two::add(one::a,one::b);
   std::cout << res <<std::endl;
   return 0;
}
```

☞ Τόσο η συνάρτηση add() όσο και οι μεταβλητές a και b χρειάζονται το πρόθεμα του χώρου ονομάτων στον οποίο ανήκουν.

☞ Επειδή δεν έχει οριστεί ως προκαθορισμένος χώρος ονομάτων ο std, η αναφορά στο αντικείμενο cout καθώς και στον χειριστή endl πρέπει να γίνεται με χρήση του προθέματος του χώρου ονομάτων στον οποίον ανήκουν.

22.15

```
#include <iostream>
using std::cout;
using std::endl;

namespace one
{
   int calc(int x,int y)
   {
      return x+y;
   }
}

namespace two
{
   int calc(int x,int y)
   {
      return x*y;
   }
}

namespace three
{
   int calc(int x,int y)
```

```
5
50
15
```

```
    {
        return x-y;
    }
}
int main()
{
    cout << three::calc(10,5)<<endl;    // Καλείται η συνάρτηση calc() του χώρου ονομάτων three
    using namespace two;                // Ορίζεται ως προκαθορισμένος χώρος ονομάτων ο two
    cout << calc(10,5)<<endl;           // Καλείται η συνάρτηση calc() του χώρου ονομάτων two
    cout << one::calc(10,5)<<endl;      // Καλείται η συνάρτηση calc() του χώρου ονομάτων one
    return 0;
}
```

Ασκήσεις Κεφαλαίου 23

23.1

- ☑ Ο μεταγλωττιστής GCC μπορεί να εκτελεί τόσο το στάδιο της μεταγλώττισης όσο και το στάδιο της σύνδεσης πηγαίων αρχείων της C και της C++.
- ☑ Η επιλογή -c του g++ έχει ως αποτέλεσμα μόνο τη μεταγλώττιση (και όχι τη σύνδεση) του πηγαίου αρχείου.
- ❑ Το εκτελέσιμο αρχείο έχει υποχρεωτικά το ίδιο όνομα με το αρχείο πηγαίου κώδικα αλλά με την προέκταση .exe.
- ☑ Στη C++ μπορούμε να δημιουργήσουμε αρχεία βιβλιοθήκης τα οποία να περιέχουν δικές μας συναρτήσεις και δικές μας κλάσεις.
- ☑ Η δημιουργία βιβλιοθηκών γίνεται με χρήση του προγράμματος αρχειοθέτησης ar, το οποίο συνοδεύει τον μεταγλωττιστή GCC.

23.2

```
#include <iostream>
using namespace std;

int main()
{
   cout<<"Χρήση του μεταγλωττιστή GCC"<<endl;
   return 0;
}
```

❶ Αρχικά θα πρέπει να κάνουμε τρέχοντα φάκελο τον c:\myfiles και να ορίσουμε ως μεταβλητή PATH τη διαδρομή των εκτελέσιμων αρχείων του μεταγλωττιστή GCC:

```
c:\cd \myfiles
c:\myfiles>path %path%;C:\Program Files\CodeBlocks\MinGW\bin
```

❷ Η μεταγλώττιση και σύνδεση του πηγαίου αρχείου γίνεται ως ακολούθως:

```
C:\myfiles>g++ -Wall ↵ first.cpp -o first
```

Για τη σωστή εμφάνιση των ελληνικών χαρακτήρων, στη θέση του συμβόλου ↵, παρεμβάλουμε τις επιλογές -finput-charset=cp1253 -fexec-charset=cp737.

❸ Εκτελούμε το αρχείο πληκτρολογώντας απλώς το όνομά του:

```
C:\myfiles>first
```

23.3

❶ Αρχικά θα πρέπει να μεταφερθούμε στον φάκελο c:\myfiles και να ορίσουμε ως μεταβλητή PATH τη διαδρομή των εκτελέσιμων αρχείων του μεταγλωττιστή GCC:

```
c:\cd \myfiles
c:\myfiles>path %path%;C:\Program Files\CodeBlocks\MinGW\bin
```

❷ Η μεταγλώττιση των δύο πηγαίων αρχείων γίνεται ως ακολούθως:

```
C:\myfiles>g++ -Wall -c ↵ info.cpp
C:\myfiles>g++ -Wall -c ↵ classes.cpp
```

Για τη σωστή εμφάνιση των ελληνικών χαρακτήρων, στη θέση του συμβόλου ↵, παρεμβάλουμε τις επιλογές -finput-charset=cp1253 -fexec-charset=cp737.

❸ Δημιουργούμε το ακόλουθο αρχείο κεφαλίδας στο οποίο περιέχονται οι δηλώσεις των συναρτήσεων και της κλάσης των πηγαίων αρχείων info.cpp και classes.cpp. Το αποθηκεύουμε στον ίδιο φάκελο με τα πηγαία αρχεία.

```
// Το αρχείο κεφαλίδας myheader.h
void info();
void chars(char ch, int fores);
class message
{
   string mes;
public:
   message();
   void display(int fores);
   void set(string m);
};
```

❹ Τροποποιούμε το αρχείο kyrio.cpp ώστε να συμπεριλάβει και το αρχείο κεφαλίδας myheader.h. Περικλείουμε το όνομα του αρχείου σε διπλά εισαγωγικά, το οποίο σημαίνει ότι το αρχείο θα αναζητηθεί στον ίδιο φάκελο με το πηγαίο αρχείο και όχι στους φακέλους εγκατάστασης του GCC.

```
// Το τροποποιημένο αρχείο πηγαίου κώδικα kyrio.cpp
#include <iostream>
using namespace std;
#include "myheader.h"

int main()
{
   message m1;
   m1.set("Η γλώσσα C++ σε βάθος");
   info();
   chars('=',50);
   m1.display(3);
   chars('#',45);
   return 0;
}
```

❺ Μεταγλωττίζουμε το πηγαίο αρχείο kyrio.cpp:

```
C:\myfiles>g++ -Wall -c ⇙ kyrio.cpp
```

Για τη σωστή εμφάνιση των ελληνικών χαρακτήρων, στη θέση του συμβόλου ⇙, παρεμβάλουμε τις επιλογές -finput-charset=cp1253 -fexec-charset=cp737.

❻ Συνδέουμε τα αρχεία αντικειμενικού κώδικα που έχουν προκύψει από τις προηγούμενες μεταγλωττίσεις σε ένα εκτελέσιμο αρχείο με όνομα all.exe:

```
C:\myfiles>g++ info.o classes.o kyrio.o -o all
```

❼ Εκτελούμε το αρχείο πληκτρολογώντας απλώς το όνομά του:

```
C:\myfiles>all
```

23.4

❶ Αρχικά θα πρέπει να μεταφερθούμε στον φάκελο c:\myfiles και να ορίσουμε ως μεταβλητή PATH τη διαδρομή των εκτελέσιμων αρχείων του μεταγλωττιστή GCC:

```
c:\cd \myfiles
c:\myfiles>path %path%;C:\Program Files\CodeBlocks\MinGW\bin
```

❷ Θα χρησιμοποιήσουμε το πρόγραμμα αρχειοθέτησης (archiver) ar το οποίο δημιουργεί ένα αρχείο βιβλιοθήκης από τις οντότητες που βρίσκονται στα διαφορετικά αρχεία αντικειμενικού κώδικα info.o και classes.o:

```
c:\myfiles>ar cr mylib.a info.o classes.o
```

❸ Πληκτρολογούμε, αποθηκεύουμε και μεταγλωττίζουμε το πηγαίο αρχείο test.cpp:

```
C:\myfiles>g++ -Wall ↵ test.cpp mylib.a -o dokimi
```

Παρατηρούμε την αναφορά στη βιβλιοθήκη mylib.a που θα χρησιμοποιηθεί, καθώς και στο όνομα του εκτελέσιμου αρχείου (dokimi.exe).

Για τη σωστή εμφάνιση των ελληνικών χαρακτήρων, στη θέση του συμβόλου ↵, παρεμβάλουμε τις επιλογές -finput-charset=cp1253 -fexec-charset=cp737.

❹ Εκτελούμε το αρχείο πληκτρολογώντας απλώς το όνομά του:

```
C:\myfiles>dokimi
```

23.5

Αφού αντικατασταθούν οι μακροεντολές και οι σταθερές που έχουν οριστεί με τις οδηγίες #define, το πρόγραμμα ισοδυναμεί με το ακόλουθο:

```
#include <iostream>
#include <cstdlib>
using namespace std; // O_XOROS_MOY
int main()  // TO_PROGRAMMA_MOY
{           // BEGIN
   do {     // APO_EDO
         cout<<"=== "<<rand()%100)<<endl; // EMFANISE(TYXAIOS);
         cin,get(); // PATA_ENTER_GIA_TON_EPOMENO;
      } while(1);   // MEXRI_EDO_KANTO_SYNEXEIA;
   return 0;  // GYRNA;
}             // END
```

Στην πρόταση **EMFANISE(TYXAIOS)**, το **X** της μακροεντολής **EMFANISE** αντικαθίσταται από το **TYXAIOS** και το **TYXAIOS** από το **rand()%100**.

☞ Το πρόγραμμα εμφανίζει συνέχεια τυχαίους αριθμούς από το 0 μέχρι το 99. Περιμένει να πατηθεί κάποιο πλήκτρο για να εμφανίσει τον επόμενο τυχαίο αριθμό. Δεν σταματάει ποτέ!

23.6

```
#include <iostream>

#define O_XOROS_MOY using namespace std;
#define FORES 5

#define A_RE_TI_FTIAXNO_SHMERA int main()
#define GYRNA_PISO return 0
#define KSEKINA_ORE {
#define TSOS }
#define GRAPSE(X) cout << X << endl;
#define KANTO_APO_EDO(X) for (int i=1;i<=X;i++) {
```

```
#define MEXRI_EDO }

O_XOROS_MOY       // Αντικαθίσταται με: using namespace std;
A_RE_TI_FTIAXNO_SHMERA // Αντικαθίσταται με: int main()
KSEKINA_ORE       // Αντικαθίσταται με: {
   KANTO_APO_EDO(FORES) // Αντικαθίσταται με: for (int i=1;i<=X;i++) {
      GRAPSE("Η ΓΛΩΣΣΑ C++ ΣΕ ΒΑΘΟΣ");      // Αντικαθίσταται με: cout << "Η γλώσσα C++ σε βάθος" << endl;
      GRAPSE("----------------------");      // Αντικαθίσταται με: cout << "----------" << endl;
   MEXRI_EDO      // Αντικαθίσταται με: }
   GYRNA_PISO;    // Αντικαθίσταται με: return 0;
TSOS              // Αντικαθίσταται με: }
```

23.7

```
#include <iostream>
#include <cassert >

#define SIZE 5
#define AKERAIOI
// #define APLHS
// #define DIPLHS
using namespace std;
int main()
{
   int i;
   assert(SIZE<=10000);
   #ifdef AKERAIOI
      int p[SIZE];
      cout<<"Δώσε "<<SIZE<<" ακέραιους αριθμούς"<<endl;
   #endif
   #ifdef APLHS
      float p[SIZE];
      cout<<"Δώσε "<<SIZE<<" αριθμούς απλής ακρίβειας"<<endl;
   #endif
   #ifdef DIPLHS
      double p[SIZE];
      cout<<"Δώσε "<<SIZE<<" αριθμούς διπλής ακρίβειας"<<endl;
   #endif
   for (i=0;i<SIZE;i++)
      cin>>p[i];
   return 0;
}
```

Καθορίζει το μέγεθος του πίνακα

Ανάλογα με το όνομα που έχει οριστεί με την οδηγία **#define**, δημιουργείται πίνακας ακεραίων, πραγματικών απλής ακρίβειας, ή πραγματικών διπλής ακρίβειας.

Στην περίπτωση που οριστεί πίνακας με μέγεθος (SIZE) μεγαλύτερο από 10000 θέσεις, η **assert** σταματάει την εκτέλεση με ανάλογο μήνυμα.

Περίπτωση που έχει οριστεί η σταθερά **AKERAIOI**.

Περίπτωση που έχει οριστεί η σταθερά **APLHS**.

Περίπτωση που έχει οριστεί η σταθερά **DIPLHS**.

23.8

```
#include <iostream>
#define SWAP(T,A,B) {T temp; temp=A; A=B; B=temp;}
using namespace std;

int main()
{
   int a=5,b=10;
   double k=6.7,l=19.8;
   SWAP(int,a,b);
   cout<<"a="<<a<<" b="<<b<<endl;
   SWAP(float,k,l);
   cout<<"k="<<k<<" l="<<l<<endl;
   return 0;
}
```

Αντιμεταθέτει τα περιεχόμενα των μεταβλητών **a** και **b**.

Αντιμεταθέτει τα περιεχόμενα των μεταβλητών **k** και **l**.

23.9

```
#include <iostream>
#include <cstdlib>
#define TYXAIOS(A,B) rand()%(B-A+1)+A
using namespace std;
int main()
{
   int i,ta;
   for (i=1;i<=10;i++)
   {
      ta=TYXAIOS(100,200);
      cout<<ta<<endl;
   }
   return 0;
}
```

Η παράσταση αυτή επιστρέφει έναν τυχαίο ακέραιο αριθμό μεταξύ του **A** και του **B**.

23.10

```
#include <iostream>

#define PRINT_INT_ARRAY(NAME,SIZE) {int i; for (i=0;i<SIZE;i++) \
cout<<NAME[i]<<endl;}

using namespace std;
int main()
{
   int pin[10]={5,6,8,2,45,6,7,9,21,32};
   PRINT_INT_ARRAY(pin,10);
   return 0;
}
```

Χαρακτήρας συνέχισης, ώστε η επόμενη γραμμή να θεωρηθεί συνέχεια αυτής της γραμμής.

23.11

- ❑ Οι μακροεντολές και οι συναρτήσεις είναι το ίδιο πράγμα.
- ☑ Οι μακροεντολές καλούνται κατά την ώρα εκτέλεσης του προγράμματος.
- ☑ Μια οδηγία μπορεί να συνεχιστεί σε περισσότερες από μία γραμμές.
- ❑ Με την #define πρέπει οπωσδήποτε να αναθέσουμε τιμή σε ένα αναγνωριστικό.
- ❑ Η assert() είναι συνάρτηση.
- ☑ Με την οδηγία #define μπορούμε να ορίσουμε μια μακροεντολή.
- ❑ Οι μακροεντολές δεν μπορούν να έχουν παραμέτρους.
- ❑ Με την οδηγία #include μπορούμε να συμπεριλάβουμε μόνο αρχεία κεφαλίδων της καθιερωμένης βιβλιοθήκης της C++.
- ☑ Η οδηγία #undef ακυρώνει τον ορισμό ενός αναγνωριστικού, ή μιας μακροεντολής, που προηγουμένως έχει οριστεί με #define.
- ☑ Με την οδηγία #if μπορούμε να ελέξουμε αν σε ένα αναγνωριστικό έχει ανατεθεί κάποια ακέραιη τιμή.

23.12

```
#include <iostream>
#include <cassert>
#include <cstdlib>
using namespace std;

void test_assert(int p[],int a,int e)
{
   int i;
   assert(e<10);
   assert(a>=0);
   for (i=a;i<e;i++)
      cout<<p[i]<<endl;
   cout<<"=====<<endl;
}

int main()
{
   int i,ar[10];
   for (i=0;i<=9;i++)
      ar[i]=rand()%100;
   test_assert(ar,4,9);
   test_assert(ar,0,2);
   test_assert(ar,5,10);
   return 0;
}
```

Η **assert()** εξασφαλίζει οτι η συνάρτηση θα κληθεί με ορίσματα **a** και **e** μέσα στα όρια του πίνακα.

Οι κλήσεις της συνάρτησης **test_assert()** με ορίσματα 0, 2 και 5, 10 ενεργοποιούν τις μακροεντολές **assert()** και σταματούν την εκτέλεση του προγράμματος με ανάλογο μήνυμα.

23.13

```
#include <iostream>
#include <cstdlib>
#include <cassert>
using namespace std;

int main()
{
   int i,ar[10],a;
   float b;
   for (i=0;i<=9;i++)
      ar[i]=rand()%100;
   cin>>a>>b;
   for (i=0;i<=a;i++)
   {
      assert(a>=0 && a<=9);
      assert(b!=0);
      cout<<ar[i]/b<<endl;
   }
   return 0;
}
```

Οι **assert()** εξασφαλίζουν ότι η τιμή της μεταβλητής **a** είναι μέσα στα όρια του πίνακα, καθώς και ότι η τιμή της **b** δεν θα είναι 0.

Ασκήσεις Κεφαλαίου 24

24.1

- ☑ Οι αλγόριθμοι επενεργούν σε αποδέκτες.
- ☑ Η λειτουργία ενός αποδέκτη μπορεί να αλλάξει με τη μεταβίβαση αντικειμένων συνάρτησης.
- ☑ Η STL διαθέτει έτοιμα αντικείμενα συναρτήσεων αλλά μπορούμε να δημιουργήσουμε και τα δικά μας.
- ❑ Οι αλγόριθμοι δεν μπορούν να εφαρμοστούν σε πίνακες των βασικών τύπων.
- ☑ Για να μπορέσουμε να χρησιμοποιήσουμε έναν αλγόριθμο θα πρέπει να έχουμε συμπεριλάβει στο πρόγραμμά μας το αρχείο κεφαλίδας algorithm.

24.2

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main()
{
   int pos;
   string *ptr;
   string pin[6]={"Νίκος","Κατερίνα","Χρυσάνθη",
                  "Άννα","Μαρία","Νίκος"};
   ptr=find(pin,pin+6,"Άννα");
   pos=ptr-pin;
   cout<<pos<<endl;
   pos=count(pin,pin+6,"Νίκος");
   cout<<pos<<endl;
   return 0;
}
```

```
3
2
```

☞ Ο αλγόριθμος find() επιστρέφει έναν δείκτη στη θέση που εντόπισε την τιμή (στην περίπτωσή μας, το «Άννα») μέσα στον αποδέκτη. Η παράσταση ptr-pin αφαιρεί τη διεύθυνση εντοπισμού από την αρχική διεύθυνση του πίνακα, επομένως επιστρέφει τη θέση του πίνακα στην οποία εντόπισε την τιμή, δηλαδή το 3.

☞ Ο αλγόριθμος count() επιστρέφει το πλήθος των φορών που εντόπισε την τιμή (στην περίπτωσή μας, το «Νίκος») μέσα στον αποδέκτη, δηλαδή το 2.

24.3

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main()
{
   int i;
   string pin[6]={ "Σοφία","Ιωάννα","Χριστίνα",
                   "Δέσποινα","Μάρκος","Νίκος"};
   sort(pin,pin+6,greater<string>());
   for (i=0;i<6;i++) cout<<pin[i]<<endl;
   return 0;
}
```

```
Χριστίνα
Σοφία
Νίκος
Μάρκος
Ιωάννα
Δέσποινα
```

☞ Ο αλγόριθμος sort() θα έχει ως αποτέλεσμα την ταξινόμηση του πίνακα pin. Το αντικείμενο συνάρτησης greater<string>() που μεταβιβάστηκε στον αλγόριθμο έχει ως αποτέλεσμα η ταξινόμηση να είναι φθίνουσα. Επομένως το πρόγραμμα θα εμφανίσει τα ονόματα του πίνακα pin με φθίνουσα σειρά.

24.4

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

bool kata_deytero(string s1,string s2) {return s1[1]<s2[1];}

int main()
{
   string str[5]={"Νίκος","Μάκης","Μένη","Χρυσάνθη","Άννα"};
   int i;
   sort(str,str+5,kata_deytero);
   for (i=0;i<5;i++) cout<<str[i]<<endl;
   return 0;
}
```

```
Μάκης
Μένη
Νίκος
Άννα
Χρυσάνθη
```

☞ Ο αλγόριθμος sort() θα έχει ως αποτέλεσμα τη ταξινόμηση του πίνακα str. Στον αλγόριθμο μεταβιβάζεται το αντικείμενο συνάρτησης kata_deytero, το οποίο συγκρίνει δύο strings κατά τον δεύτερο χαρακτήρα τους (s1[1]<s2[1]).

24.5

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
   vector<int> coll(5,5);
   int i;
   for (i=0;i<5;i=i+2) coll[i]=i;
   coll.push_back(10);   // προσθέτει στο τέλος το 10
   coll.push_back(20);   // προσθέτει στο τέλος το 20
   coll.pop_back();
   cout<<"Μέγεθος = "<<coll.size()<<endl;
   // προσθέτει σε κάθε στοιχείο του διανύσματος το 1
   for (i=0;i<coll.size();i++) coll.at(i)++;
   // εμφανίζει όλα τα στοιχεία του διανύσματος
   for (i=0;i<coll.size();i++) cout<<coll[i]<<endl;
   return 0;
}
```

Δημιουργεί ένα διάνυσμα ακεραίων πέντε θέσεων, με αρχική τιμή κάθε θέσης το 5.

Καταχωρίζει στις ζυγές θέσεις την τιμή του **i** αντικαθιστώντας την υπάρχουσα.

```
Μέγεθος = 6
1
6
3
6
5
11
```

24.6

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

int main()
{
   vector<string> filoi;
   vector<string>:: iterator it;
   filoi.push_back("Νίκος");
   filoi.push_back("Άκης");
   filoi.push_back("Μαρία");
   filoi.push_back("Γιώργος");
   filoi.push_back("Ελένη");
   it=filoi.begin();
   it=it+2;
```

Δημιουργείται ένα διάνυσμα αντικειμένων της κλάσης **string**, καθώς και ένας επαναλήπτης για αντικείμενα τέτοιων διανυσμάτων.

Προσθέτει τα ονόματα στο διάνυσμα

Ο επαναλήπτης δείχνει στην τρίτη θέση του διανύσματος.

```
   // παρεμβολή του αλφαριθμητικού "*****" πριν από τη τρίτη θέση
   filoi.insert(it,"*****");
   sort(filoi.begin(),filoi.end());
   for (it=filoi.begin();it!=filoi.end();it++) cout<<*it<<endl;
   return 0;
}
```

Ταξινόμηση του διανύσματος.

Εμφάνιση των στοιχείων του διανύσματος.

24.7

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

class person
{
public:
   string onoma;
   int ilikia;
   float poso;
};

bool kata_ilikia(person p1,person p2) {return p1.ilikia<p2.ilikia;}

int main()
{
   vector<person> atoma;
   vector<person>:: iterator it;
   person temp;
   temp.onoma="Νίκος";
   temp.ilikia=34;
   temp.poso=56.50;
   atoma.push_back(temp);
   temp.onoma="Τάκης";
   temp.ilikia=26;
   temp.poso=100.0;
   atoma.push_back(temp);
   temp.onoma="Μαρία";
   temp.ilikia=19;
   temp.poso=80.0;
   atoma.push_back(temp);
   temp.onoma="Ιωάννα";
   temp.ilikia=22;
   temp.poso=40.5;
   atoma.push_back(temp);
   sort(atoma.begin(),atoma.end(),kata_ilikia);
   for (it=atoma.begin();it!=atoma.end();it++)
      cout<<it->onoma<<endl;
   return 0;
}
```

Η συνάρτηση **kata_ilikia()** θα χρησιμοποιηθεί ως αντικείμενο συνάρτησης για τον αλγόριθμο **sort**, βάση της οποίας θα συγκρίνει δύο αντικείμενα της κλάσης **person**.

Δημιουργείται ένας αποδέκτης διανύσματος αντικειμένων της κλάσης **person**, με όνομα **atoma**, καθώς και ένας επαναλήπτης για αντικείμενα τέτοιων αποδεκτών.

Κάθε σετ τιμών (όνομα/ηλικία/ποσό) καταχωρίζεται στο προσωρινό αντικείμενο **temp** και κατόπιν προστίθεται στον αποδέκτη διανύσματος.

Γίνεται ταξινόμηση των αντικειμένων του διανύσματος με βάση το αντικείμενο συνάρτησης **kite_illicit**. Το αντικείμενο αυτό έχει ως αποτέλεσμα ο αλγόριθμος **sort** να ταξινομήσει τα αντικείμενα της κλάσης **person** ως προς το πεδίο της ηλικίας.

Χρησιμοποιώντας τον επαναλήπτη **it** έχουμε πρόσβαση στο κάθε αντικειμένου του διανύσματος. Εμφανίζουμε το μέλος **onoma** του κάθε αντικειμένου στο οποίο δείχνει ο επαναλήπτης!

24.8

```
#include <iostream>
#include <string>
#include <map>
using namespace std;

int main()
{
   map<string,string> names;
   map<string,string>::iterator iter;
   names["B12345"] = "Νίκος";
   names["Γ23465"] = "Μανώλης";
   names["X76534"] = "Πάτροκλος";
   names["K89345"] = "Βενετία";
   names["M67856"] = "Ελένη";
   for (iter = names.begin(); iter != names.end(); iter++)
      cout << iter->first << " -- " << iter->second << endl;
   return 0;
}
```

Δημιουργείται ένας αποδέκτης αντιστοιχίας αντικειμένων της κλάσης **string**, με κλειδί επίσης κλάσης **string**, με όνομα **names.** Ορίζεται επίσης και ένας επαναλήπτης για αντικείμενα τέτοιων αποδεκτών, με όνομα **iter**.

Στον αποδέκτη **names** καταχωρίζονται τα πέντε ονόματα με τιμές κλειδιων τον αριθμό ταυτότητας του καθενός.

Χρησιμοποιώντας τον επαναλήπτη **iter** έχουμε πρόσβαση στο τμήμα κλειδιού (first) και στο τμήμα δεδομένων (second) του κάθε αντικειμένου της αντιστοιχίας.

24.9

```
#include <iostream>
#include <string>
#include <map>
using namespace std;

class person
{
public:
   string onoma;
   int ilikia;
   void display()
   {
      cout<<"Όνομα :"<<onoma<<endl;
      cout<<"Ηλικία :"<<ilikia<<endl;
      cout<<"--------------------"<<endl;
   }
};

map<int,person> mymap;
map<int,person>::iterator myit;

int main()
{
   person temp;
   do
   {
      getline(cin,temp.onoma);
      if (temp.onoma=="") break;
      cin>>temp.ilikia;
      mymap.insert(map<int,person>::value_type(temp.ilikia,temp));
      cin.get();
   } while (1);
   for (myit=mymap.begin();myit!=mymap.end();myit++)
      myit->second.display();
   return 0;
}
```

Η μέθοδος **display()** εμφανίζει τα στοιχεία ενός αντικειμένου κλάσης **person**.

Δημιουργείται ένας αποδέκτης αντιστοιχίας αντικειμένων της κλάσης **person**, καθώς και ένας επαναλήπτης για αντικείμενα τέτοιων αποδεκτών.

Κάθε ζευγάρι τιμών που πληκτρολογείται (όνομα/ηλικία) καταχωρίζεται στο προσωρινό αντικείμενο **temp** και κατόπιν προστίθεται στον αποδέκτη αντιστοιχίας **mymap** με κλειδί την ηλικία.

Εμφανίζονται τα στοιχεία όλων των αντικειμένων της αντιστοιχίας.

☞ Το πρόγραμμα ζητάει από τον χρήστη να πληκτρολογεί ονόματα και ηλικίες μέχρι να πληκτρολογήσει κενό όνομα (δηλαδή να πατήσει απλώς το <Enter>). Κάθε ζευγάρι τιμών που πληκτρολογείται (όνο-

μα/ηλικία) καταχωρίζεται στο προσωρινό αντικείμενο temp και κατόπιν προστίθεται στον αποδέκτη αντιστοιχίας mymap με κλειδί την ηλικία.

☞ Ακολούθως εμφανίζονται τα στοιχεία των αντικειμένων του αποδέκτη αντιστοιχίας, τα οποία θα εμφανιστούν με αύξουσα σειρά ηλικίας επειδή το κλειδί είναι η ηλικία. Τα στοιχεία εμφανίζονται με χρήση του επαναλήπτη myit, που εφαρμόζει τη μέθοδο display() στο τμήμα δεδομένων (second) του αντικειμένου στο οποίο δείχνει κάθε φορά ο επαναλήπτης.

24.10

```
#include <iostream>
#include <string>
#include <map>

using namespace std;

class person
{
public:
   string onoma;
   int ilikia;
   float poso;
};

map<int,person> mymap;
map<int,person>::iterator myit;

int main()
{
   person temp;
   float mo_il,sum_il=0;
   float sum_poso=0;
   temp.onoma="Νίκος";
   temp.ilikia=34;
   temp.poso=56.50;
   mymap.insert(map<int,person>::value_type(123,temp));
   temp.onoma="Τάκης";
   temp.ilikia=26;
   temp.poso=100.00;
   mymap.insert(map<int,person>::value_type(101,temp));
   temp.onoma="Μαρία";
   temp.ilikia=19;
   temp.poso=80.00;
   mymap.insert(map<int,person>::value_type(201,temp));
   temp.onoma="Ιωάννα";
   temp.ilikia=22;
   temp.poso=40.50;
   mymap.insert(map<int,person>::value_type(204,temp));
   for (myit=mymap.begin();myit!=mymap.end();myit++)
   {
      sum_il=sum_il+myit->second.ilikia;
      sum_poso=sum_poso+myit->second.poso;
   }
   mo_il=sum_il/mymap.size();
   cout<<"Μέσος όρος ηλικιών:"<<mo_il<<endl;
   cout<<"Συνολικό ποσό:"<<sum_poso<<endl;
   return 0;
}
```

Δημιουργείται ένας αποδέκτης αντιστοιχίας αντικειμένων της κλάσης **person**, καθώς και ένας επαναλήπτης για αντικείμενα τέτοιων αποδεκτών.

Στο προσωρινό αντικείμενο **temp** καταχωρίζονται τα στοιχεία της εκφώνησης και κατόπιν προστίθενται στον αποδέκτη αντιστοιχίας **mymap** με κλειδί τις τιμές που ορίζονται στην εκφώνηση.

Χρησιμοποιώντας τον επαναλήπτη **myit** έχουμε πρόσβαση στο τμήμα δεδομένων (second) του κάθε αντικειμένου της αντιστοιχίας. Τα πεδία **ilikia** και **poso** προστίθενται αντίστοιχα στους αθροιστές **sum_il** και **sum_poso**.

Η μέθοδος **size()** επιστρέφει το πλήθος των στοιχείων του αποδέκτη **mymap** και χρησιμοποιείται για τον υπολογισμό του μέσου όρου των ηλικιών.

24.11

- ☑ Τα αντικείμενα που περιέχουν οι ακολουθιακοί αποδέκτες καταλαμβάνουν συνεχόμενες θέσεις μνήμης.
- ☑ Οι επαναλήπτες χρησιμοποιούν όπως και οι δείκτες τον τελεστή * για την αναφορά στο αντικείμενο στο οποίο δείχνουν.
- ❑ Όλοι οι αλγόριθμοι μπορούν να εφαρμοστούν σε όλους τους αποδέκτες.
- ☑ Αν αυξήσουμε έναν επαναλήπτη κατά 1, τότε αυτός θα δείχνει στο επόμενο αντικείμενο ενός αποδέκτη, ανεξάρτητα από το είδος του αποδέκτη.
- ❑ Αν αυξήσουμε έναν επαναλήπτη κατά 2, τότε αυτός θα δείχνει στο μεθεπόμενο αντικείμενο ενός αποδέκτη, ανεξάρτητα από το είδος του αποδέκτη.