Μικρή επανάληψη από pointers

```
#include <iostream>
using namespace std;
int main() {
  int p[] = {10,20,30,40};
  int *q = p;
  int x;
  x = *q;
  cout << x << endl;
  x = *q++;
  cout << x << endl;
  x = *(q++);
 cout << x << endl;
  x = (*q)++;
  cout << x << endl;
  x = *q;
  cout << x << endl;
  x = *++q;
  cout << x << endl;
  x = ++*q;
  cout << x << endl;
  return 0;
}
```

← x ισούται με τη τιμή του ακεραίου που δείχνει το q

← x ισούται με τη τιμή του ακεραίου που δείχνει το q (άρα 10).
Μετά το q δείχνει στη διεύθυνση του επόμενου ακέραιου (p[1])

← x ισούται με τη τιμή του ακεραίου που δείχνει το q (άρα 20 από το προηγούμενο).
Μετά το q αυξάνει και δείχνει στη διεύθυνση του επόμενου ακέραιου (p[2])

← x ισούται με τη τιμή του ακεραίου που δείχνει το q (άρα 20 από το προηγούμενο).
Μετά η τιμή στη διεύθυνση του q (εδώ το στοιχείο p[2]) αυξάνει κατά 1 σε 31

← x ισούται με τη τιμή του ακεραίου που δείχνει το q (άρα 31 από το προηγούμενο).

← πρώτα αυξάνει το q κατά 1 (άρα δείχνει στη διεύθυνση του p[4]) και
μετά κάνε ανάθεση στο x την τιμή στην θέση p[4] (άρα x=40).

← πρώτα αυξάνει τη τιμή της διεύθυνσης του q κατά 1 (επομένως p[4]=41) και
μετά κάνε ανάθεση της τιμής στο x ι(x=41)

Strings and streams

Το σύστημα Unix/Linux αναγνωρίζει 3 βασικές input/output ροές χαρακτήρων (streams)

Αυτές είναι για input το std::cin, για output std::cout και για σφάλμα std::cerr και std::clog

Τα δυο τελευταία χρησιμοποιούνται για να μεταφέρουν πληροφορία που δεν έχει σχέση με τα αποτελέσματα του προγράμματος αλλά προειδοποιήσεις ή ειδοποιήσεις λαθών

- Τα παραπάνω streams ορίζονται στο <iostream>
- Στο <iostream> υπάρχουν 2 μέθοδοι η **istream** και η **ostream** που επιτρέπου είσοδο και έξοδο είτε στην οθόνη ή από το πληκτρολόγιο ή από file που έχουν διευθεντηθεί προς το input/output του προγράμματος
- Δεν απαιτείται ονομασία ενός αρχείου

istream

```
#define MAX 100
int main(){

  double a[MAX];
  int n;
  ...

  n = 0;
  while(n < MAX){

     cin >> a[n];

       if (cin.fail()){
          if(cin.eof())break;
          cerr << "Data item number "
               << n+1
               << " has bad numeric format\n";
          return 1;
        }
      n++;
   }
  ...
}
```

Η cin αποτελεί μια παρουσία της κλάσης istream

Η μέθοδος eof() επιστρέφει true αν έχουν τελειώσει τα δεδομένα από το πληκτρολόγιο ή αν έχουν τελειώσει τα δεδομένα από κάποιο αρχείο που διευθύνεται στο πρόγραμμα

Η μέθοδος fail() επιστρέφει true αν υπήρξε πρόβλημα για κάποιο λόγο

Στο διπλανό κώδικα βγαίνουμε εκτός προγράμματος αν έχουν τελειώσει τα δεδομένα (eof)

Αν υπάρξει οποιοδήποτε άλλο σφάλμα (formatting error) δίνουμε προειδοποίηση

ostream

```
#include <iostream>
   ...
  double x,y;
   ...

  // Set 4 digits past decimal
  cout.precision(4);
  /* Specify right-justified numbers
     in fixed format (xxx.xxxx)
     Note: Multiple options are added and
     then passed to flags. */

  cout.flags(ios::right | ios::fixed);
  // Write a column header
   cout << "\n  x     y\n";
   cout << "-------------\n";
   while(1){
       cin >> x >> y;
       if(cin.fail())break;  // Stop on end-of-file or format error
      // Specify 7 character field width.
      // Must be done for each value.
    cout.width(7);
    cout << x;
    cout.width(7);
    cout << y << "\n";
  }
```

Στο διπλανό κώδικα precision() είναι η μέθοδος που δίνει τον αριθμό των δεκαδικών ψηφίων ή το συνολικό αριθμό ψηφίων για επιστημονική σήμανση

Η μέθοδος flags χρησιμοποιείται για στοίχιση και τρόπο αναπαράστασης

Χρησιμοποιούμε την κάθετο | για συνδυασμό επιλογών

right	δεξιά στοίχιση
left	αριστερή στοίχιση
fixed	αναπαράστση δεκαδ.
scientific	επιστημονική αναπαρ.
floatfield	αναμεικτη δεκαδική/επιστ.
hex	δεκαεξαδικό

Αυτόματα χρησιμοποιείται το floatfield που σημαίνει όποιο ταιριάζει καλύτερα

- Η μέθοδος width καθορίζει τον αριθμό των χαρακτήρων για κάθε αριθμό και θα πρέπει να καλείται κάθε φορά. Αν υπάρχουν περισσότερα ψφία, η width αγνοείται

printf

Αποτελεί μια πιο εύχρηστη κλάση εξόδου με formatted τρόπο και μπορεί να χρησιμοποιηθεί μαζί με την cout

```
#include <stdio.h>   ←νέο header file
using namespace std;
int main(){
   // Write a column header
   printf("\n  x     y\n");
   printf("-------------\n");
   for( i = 0; i < 10; i++)
     printf("%7.4f%7.4f\n",x[i],y[i]);
  return 0;
}
```

πεδίο	σκοπός μετατροπής
%w.df	fixed format xx.xxxxx (για float, double)
%w.de	επιστημονική σήμανση x.xxxennn float, double)
%w.dg	μεταβλητό format (float, double)
%w.d	ακέραιοι xxxx. int
%ws	Γραμματοσειρές. char * και χωρίς το w (%s)

Ο τρόπος γραφής στην περίπτωση αυτή προσδιορίζεται από τα ορίσματα της συνάρτησης printf.

- ✓ Το σύμβολο **%** εισάγει το τρόπο μετατροπής.
- ✓ Το σύμβολο **%** εισάγει το τρόπο μετατροπής. Για κάθε τιμή που θα τυπωθεί χρειάζεται να προσδιοριστεί ο τρόπος μετατροπής και λαμβάνεται με τη σειρά γραφής τους από αριστερά στα δεξιά
- ✓ Μετά το σύμβολο **%** εισάγεται το μήκος των χαρακτήρων, W, για τον αριθμό και το 2ο νούμερο μετά την τελεία, (D), προσδιορίζει τον αριθμό των δεκαδικών ψηφίων. Η μορφή είναι **%W.D**
- ✓ Αυτό που ακολουθεί στην περιγραφή του τρόπου μετατροπής είναι το είδος της τιμής που θα τυπωθεί (όπως προσδιορίζονται στο διπλανό πίνακα)
- ✓ Οποιαδήποτε γραμματοσειρά περιεκλείεται ανάμεσα στα " " της συνάρτησης τυπώνονται κανονικά σα μυνήματα
- ✓ Για το g format μπορεί να έχουμε f, e ή d ανάλογα με το μέγεθος του αριθμού και αν υπάρχουν μόνο μηδενικά μετά την υποδιαστολή

ifstream/ofstream

- Αν θέλουμε να διαβάσουμε και να γράψουμε σε αρχεία με συγκεκριμένα ονόματα χρησιμοποιώντας τις απλές κλάσεις cin και cout μπορούμε να κάνουμε ως εξής:

  ./myprogram.exe < infilename > outfilename

- Η προηγούμενη μέθοδος δουλεύει μόνο σε περιπτώσεις ενός ρεύματος input και ενός output και όχι αν υπάρχει μεγαλύτερη ανάγκη για μεταφορά δεδομένων
- Η λύση στο πρόβλημα αυτό δίνεται από την κλάση ***ifstream*** που προέρχεται από την κλάση istream. Το επιπλέον f στο όνομα υπενθυμίζει ότι χρησιμοποιείται το όνομα κάποιου file.
  Η κλάση ***ofstream*** χρησιμοποιείται για έξοδο σε file.
- Οι δυο αυτές κλάσεις ορίζονται στο header file ***fstream***
- Τα βήματα που απαιτούνται για την χρήση ενός file για είσοδο/έξοδο δεδομένων είναι:
  - Δημιουργία ενός στιγμιότυπου της κλάσης ifstream ή ofstream
  - Άνοιγμα του file
  - Πρόσβαση στο file για ανάγνωση ή εγγραφή δεδομένων
  - Κλείσιμο του file

```
include <fstream>   ←νέο header file
 int main(){
    ifstream table;   ←δημιουργία στιγμιότυπου
    float a[10];
    table.open("tabledata");   ←άνοιγμα του αρχείου
    if(table.fail()){cerr << "Can't open tabledata\n"; return 1;}   ←έλεγχος λάθους
    for(int i = 0; i < 10; i++) table >> a[i];   ←ανάγνωση δεδομένων
    table.close();   ←κλείσιμο του αρχείου
 }
```

ifstream/ofstream

- Το όνομα του στιγμιότυπου της κλάσης μπορεί να είναι οποιοδήποτε
- Για κάθε file θα πρέπει να υπάρχει ένα νέο στιγμιότυπο των κλάσεων ifstream ή ofstream
  Χρησιμοποιούνται ωστόσο ακριβώς με τον ίδιο τρόπο όπως και τα cin και cout

```
include <fstream>   ←νέο header file
 int main(){
    ifstream table;   ←δημιουργία στιγμιότυπου
    float a[10];
    table.open("tabledata");   ←άνοιγμα του αρχείου
    if(table.fail()){cerr << "Can't open tabledata\n"; return 1;}
    for(int i = 0; i < 10; i++) table >> a[i];   ←ανάγνωση
    table.close();   ←κλείσιμο του αρχείου
 }
```

- Η μέθοδος **open** της κλάσης παίρνει σαν όρισμα το όνομα του file
- Η μέθοδος **close** της κλάσης κλείνει το file.
  Δεν είναι απαραίτητη αλλά είναι καλή πρακτική
- Τα δυο πρώτα βήματα θα μπορούσαν να συμπτιχθούν σε ένα αν χρησιμοποιήσουμε την συνάρτηση κατασκευής της κλάσης ως εξής:
  ifstream table("tabledata");
- Οι μέθοδοι ***eof*** και ***fail*** δουλεύουν με την κλάση ifstream

strings

Η κλάση αυτή επιτρέπει τον ευκολότερο χειρισμό γραμματοσειρών χωρίς χρήση μεταβλητών τύπου char και χωρίς να ανησυχούμε για το μήκος των γραμματοσειρών

```
#include <iostream>
#include <string>    ←νέο header file
using namespace std;

int main()
{
  string firstname, lastname, fullname;
  char praise[] = " the Magnificent";

  cout << "enter first name: \n";
  cin >> firstname;
  cout << "enter last name: \n";
  cin >> lastname;
  fullname = firstname + " " + lastname + praise;
  cout << "Your full name is " + fullname + "\n";
  cout << "It has " << fullname.size() << " characters\n";
  cout << "The first character is " << fullname[0] << "\n";
}
```

Το μήκος των γραμματοσειρών τύπου string αυξομειώνεται δυναμικά ανάλογα με τη χρήση

Το σύμβολο + χρησιμοποιείται για τη σύμπτυξη γραμματοσειρών σε μια

Μερικές χρήσιμες μέθοδοι της κλάσης είναι:

empty(): true αν η string είναι κενή

size(): επιστρέφει το μέγεθος της γραμματοσειράς

c_str(): επιστρέφει pointer για το πρώτο χαρακτήρα της string

ροές strings

Αν θέλαμε να μπορούμε να εκτυπώσουμε αριθμούς σαν string ενώνοντάς τους με χαρακτήρες θα χρησιμοποιούσαμε τις κλάσεις που περιλαμβάνται στο header file <sstream>

Παρέχονται δυο κλάσεις ***istringstream*** και ***ostringstream***

Η εκτύπωση σε ***ostringstream*** δημιουργεί ένα string με συνένωση των εκτυπούμενων ποσοτήτων

```
#include <sstream>
using namespace std;
int main(){
   ostringstream os;
   os<< "filename_";
   os<< 3;
   os<< ".dat" ;
```

← νέο header file

← η os περιέχει τη γραμματοσειρά filename_3.dat

Για να δούμε το περιεχόμενο της os θα πρέπει να καλέσουμε τη μέθοδο ***str()***

cout << os.str() << endl;

Για να ανοίξουμε επομένως ένα file με όνομα "filename_3.dat" δίνουμε την εντολή

ofstream outstr(os.str().c_str());

Αντικείμενο τύπου istringstream χρησιμοποιείται για την ανάγνωση τιμών από το string με το οποίο συνδέεται όπως θα γινόταν από αρχείο:

```
#include <sstream>
int main() {
  std::istringstream is("5_6_7_a");
  int i,j,k;
  is >> i;    // i = 5
  is >> j;    // j = 6
  is >> k;    // k = 7
  char ch;  is >> ch;  // ch = 'a'
```

Παράδειγμα – άνοιγμα αρχείων με δυναμικά ονόματα

```
#include <iostream>
#include <fstream>
#include <sstream>

using namespace std;
void out(int);

int main()
{
  for (int numb=0; numb<3; numb++){
    out(numb);
  }
  return 0;
}

void out(int n)
{
  ostringstream ss;
  ss << "crap" <<n<<".dat" ;
  cout << ss.str() << endl;
  ofstream open(ss.str());
}
```

Δομές/structure: struct

Δομές είναι μια συλλογή μεταβλητών, διαφορετικού τύπου εν γένει, κάτω από ένα όνομα, το οποίο βοηθά στην καλύτερη οπτικοποίηση του προβλήματος και καλύτερη οργάνωση.

Μοιάζουν με τους arrays αλλά εκεί έχουμε κάτω από το ίδιο όνομα, συλλογές του ίδιου μόνο τύπου μεταβλητών/στοιχείων

Ο ορισμός μιας δομής γίνεται με τον ακόλουθο τρόπο:

```
struct particles {
    char name[20];
    float momentum;
    float energy;
    float charge;
};
```

Δήλωση του ονόματος της δομής (particles)

Τα μέλη (members) της δομής περικλείονται σε {...}

Η δομή αποτελείται από 4 μέλη, name, momentum, energy και charge

Η δήλωση της δομής δεν καταλαμβάνει χώρο στη μνήμη

Προσοχή στο καταληκτικό ;

Η δομής προσδιορίζει ένα νέο τύπο δεδομένων με κάποιες ιδιότητες που προσδιορίζουν τα μέλη Έτσι μπορούμε να ορίσουμε μεταβλητές αυτού του τύπου (όπως κάνουμε για int, float κλπ)

particles hadron Ορισμός της μεταβλητής hadron τύπου particle

Εδώ γίνεται ο προσδιορισμός της μνήμης της δομής (20+3x4)=32bytes (κάθε μεταβλητή τύπου char καταλαμβάνει 1byte)

Πρόσβαση στα μέλη μιας μεταβλητής τύπου δομής αποκτάται δίνοντας το όνομα της μεταβλητής τύπου δομής ακολουθούμενη από τον τελεστή "**.**" και το όνομα του μέλους

hadron.energy = 20;

Ανάθεση τιμών στα μέλη μπορεί να γίνει όπως και στα στοιχεία των πινάκων τηρώντας τη σειρά δήλωσης των μελών: **hadron = {"kaon",20,10,-1};**

structures

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

struct  particles {
    char name[20];
    float energy;
    float momentum;
};

int main() {

    particles hadron;

    cout << "Enter Full name: ";
    cin.get(hadron.name, 20);
    cout << "Enter energy: ";
    cin >> hadron.energy;
    cout << "Enter momentum: ";
    cin >> hadron.momentum;

    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << hadron.name << endl;
    cout <<"Energy: " << hadron.energy << endl;
    cout << "Momentum: " << hadron.momentum;

    return 0;
}
```

Δομές σαν ορίσματα συναρτήσεων

Δομές μπορούν να δοθούν σαν ορίσματα σε συναρτήσεις όπως συμβαίνει και με μεταβλητές συνηθισμένων τύπων.

```
#include <iostream>
using namespace std;
struct  particle {
    char name[50];
    float energy;
    float momentum;
};
void displayData(particle) ←
int main() {
    particles hadron;
    cout << "Enter Full name: ";
    cin.get(hadron.name, 20); ←
    cout << "Enter energy: ";
    cin >> hadron.energy;
    cout << "Enter momentum: ";
    cin >> hadron.momentum;
    displayData(hadron);
    return 0;
}
void displayData(particle had) { ←
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << had.name << endl;
    cout <<"Energy: " << had.enerrgy << endl;
    cout << "Momentum: " << had.momentum;
}
```

Δήλωση του συνάρτησης με όρισμα τύπου particles

ανάγνωση του ονόματος

κλήση της συνάρτησης δίνοντας το όνομα της δομής μεταβλητής

η συνάρτηση με όρισμα δομή

Δομές σαν επιστροφή συναρτήσεων

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

struct  particles {
    char name[20];
    float ene;
    float mom;
};

particles getData(particles);
void displayData(particles);

int main() {
    particles part;
    part = getData(part);
    displayData(part);
    return 0;
}
```

Δήλωση συνάρτησης τύπου struct particles

Πέρασμα της δομής part σαν όρισμα στη συνάρτηση. Τα μέλη της δομής γεμίζουν στη συνάρτηση και αποθηκεύονται στη δομή had που επιστρέφει η συνάρτηση και κάνει ανάθεση στην part

```
particles getData(particles had) {
    cout << "Enter Full name: ";
    cin.get(had.name, 20);
    cout << "Enter energy: ";
    cin >> had.ene;
    cout << "Enter momentum: ";
    cin >> had.mom;
    return had;
}
```

```
void displayData(particles had) {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << had.name << endl;
    cout <<"Energy: " << had.ene << endl;
    cout <<"Momentum: " << had.mom;
}
```

Δομές – οποιοσδήποτε τύπος

Μεταβλητές τύπου δομής μπορεί να είναι επίσης arrays: **particles hadron[20];**

Η πρόσβαση σε κάποιο στοιχείο γίνεται ως: **hadron[2].energy;**

Μπορεί να έχουμε vectors τύπου struct στα οποία αποθηκεύουμε τύπους struct

```
#include <iostream>
using namespace std;
struct  particles {
    float ene;
    float mom;
};

void getData(particles &);

int main() {
    vector<particles> partcoll;
    particles had;
    getData(had);
    partcoll.push_back(had);
    return 0;
}

void getData(particles &part){
     part.ene = 10;
     part.mom = 5;
}
```

Δομές και δείκτες

Ένας δείκτης μπορεί να οριστεί και για δομές όπως συμβαίνει για τους κανονικούς τύπου:

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

struct temp {
    int i;
    float f;
};

int main() {
    temp *ptr;
    return 0;
}
```

pointer τύπου structure temp

Η γραφή **(*ptr).member** είναι ισοδύναμη με **ptr->member** που χρησιμοποιεί τον τελεστή ->

- Προσοχή το **(*ptr).member** είναι τελείως διαφορετικό από *(ptr.member)

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

struct Distance {
    int feet;
    float inch;
};

int main() {
    Distance *ptr, d;

    ptr = &d;

    cout << "Enter feet: ";
    cin >> (*ptr).feet;
    cout << "Enter inch: ";
    cin >> (*ptr).inch;

    cout << "Displaying information." << endl;
    cout << "Distance = " << (*ptr).feet << " feet "
         << (*ptr).inch << " inches";

    return 0;
}
```

αποθήκευση της διεύθυνσης της κανονικής μεταβλητής struct στο ptr

πρόσβαση στα μέλη της struct μέσω του pointer

Δομές με μέλη συναρτήσεις – C++ μόνο

Μέχρι τώρα έχουμε χρησιμοποιήσει τις δομές μόνο με data μέλη. Είδαμε πως αν θελήσουμε να χρησιμοποιήσουμε τις δομές σε συναρτήσεις θα πρέπει να περάσει τη δομή σαν όρισμα στην συνάρτηση.

Στη C++ μπορούμε να ορίσουμε στις δομές συναρτήσεις μέλη με 2 προϋποθέσεις:

- Η συνάρτηση μέλος ορίζεται μέσα στο block της δομής για να φανεί η συσχέτιση της συνάρτησης και της δομής
- Η συνάρτηση καλείται συσχετίζοντάς την με κάποιο αντικείμενο της δομής

```
#include <iostream>
using namespace std;
struct  particles {
    float ene;
    float mom;
};

void init(particles &);
int main() {
    vector<particles> partcoll;
    particles had;
    had.init();
    return 0;
}

void init(particles &p1){
   p1.ene = 0;
   p1.mom = 0;
}
```

```
#include <iostream>
using namespace std;
struct  particles {
    float ene;
    float mom;
    void init{
       ene = 0.;
       mom = 0.;
    }
};

int main() {
    vector<particles> partcoll;
    particles had;
    had.init();
    return 0;
}
```

← κλήση της συνάρτησης

```
#include <iostream>
using namespace std;
struct  particles {
    float ene;
    float mom;
    void particles::init();
};

int main() {
    vector<particles> partcoll;
    particles had;
    had.init();
    return 0;
}

void particles::init() {
     particles had = *this;
     had.ene = 0; had.mom=0;
}
```

Δομές με μέλη συναρτήσεις – C++ μόνο

```
#include <iostream>
using namespace std;
struct  particles {
    float ene;
    float mom;
    void particles::init();
};

int main() {
    vector<particles> partcoll;
    particles had;
    had.init();
    return 0;
}

void particles::init() {
     particles had = *this;
     had.ene = 0; had.mom=0;
}
```

Ο pointer this **this** (ακριβώς αυτό το όνομα έχει η C++) για να δηλώσει ένα pointer στο τρέχον αντικείμενο με το οποίο έχει κληθεί η συνάρτηση.

Ωστόσο ο τρόπος με τον οποίο γράψαμε την συνάρτηση είναι πολύ περισσότερο πολύπλοκος απ' ότι χρειάζεται.

Αφού καλείται με κάποιο αντικείμενο, αν η συνάρτηση περιέχει μέλη της structure χωρίς την χρήση του τελεστή . τότε είναι αυτονόητο ότι αναφέρεται στο αντικείμενο με το οποίο κλήθηκε και μπορούμε να αγνοήσουμε την δήλωση του αντικειμένου

```
#include <iostream>
using namespace std;
struct  particles {
    float ene;
    float mom;
    void particles::init();
};
int main() {
    vector<particles> partcoll;
    particles had;
    had.init();
    return 0;
}
void particles::init() {
    ene = 0; mom=0;
}
```

Κλάσεις στη C++

- Ο χρήστης μπορεί να ορίσει νέους τύπους αντικειμένων, η πρόσβαση στις ιδιότητες (στοιχεία) των οποίων γίνεται μέσω συγκεκριμένων συναρτήσεων
- Τύπος είναι η αναπαράσταση ενός «αντικειμένου» όπως float με συναρτήσεις +,-,*,/,... είναι η αναπαράσταση των αριθμών κινητής υποδιαστολής
- Θα πρέπει να υπάρχει διαχωρισμός των λεπτομερειών υλοποίησης του τύπου από τις ιδιότητες / λειτουργίες του τύπου
- Η πρόσβαση στις ιδιότητες των τύπων γίνεται μέσω συναρτήσεων που ονομάζονται member functions της κλάσης. (Μπορεί να γίνει και με φιλικών προς την κλάση (friends)
- Όταν δημιουργούνται τα αντικείμενα της κλάσης, καλούνται συγκεκριμένες συναρτήσεις που δηλώνονται στην κλάση και λέγονται συναρτήσεις κατασκευής (constructors)
- Κατ΄αναλογία υπάρχουν και συναρτήσεις που αποδομούν το χώρο που χρησιμοποιήθηκε για τα αντικείμενα της κλάσης (destructors)
- Το αντικείμενο μιας κλάσης μπορεί να δημιουργηθεί σαν static είτε σαν automatic ή new