Εισαγωγή στον Προγραμματισμό

Εργαστήριο Υπολογιστικής Φυσικής

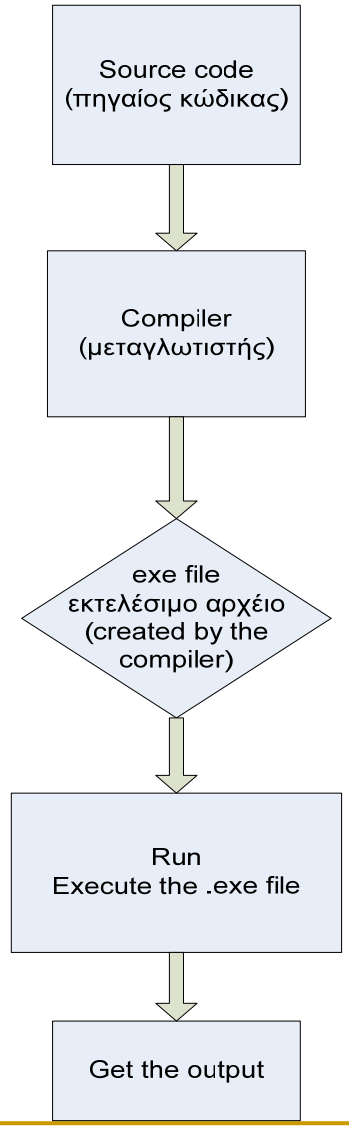
Επικοινωνία

- Νίκος Τσακίρης : tsaknik@kelifos.physics.auth.gr
- Λουκάς Σκαρπαλέζος : loukas@kelifos.physics.auth.gr

Εργαστήριο Υπολογιστικής Φυσικής

http://kelifos.physics.auth.gr/

Flow chart για εκτέλεση ενός προγράμματος στη C



Πρόκειται για το αρχείο που περιέχει τις εντολές της γλώσσας προγραμματισμού.

Μετατρέπει το source code σε γλώσσα μηχανής, κατανοητή από τον επεξεργαστή. Επιπλέον ελέγχει για σφάλματα στον κώδικα. Τελικά δημιουργεί το εκτελέσιμο (.exe) αρχείο.

Εκτελείται το .exe αρχείο.

Μετά την εκτέλεση του .exe, προκύπτει το αποτέλεσμα

Ο πρώτος κώδικας σε C

```
/*
        Source code in C
*/

#include <stdio.h>

int main()
{
        printf(“ Hello World\n”);   /* Εκτύπωσε το Hello World*/
        return 0;
}
```

Ο πρώτος κώδικας σε C - Σχόλια

- Τα σχόλια σε ένα κείμενο source code (πηγαίου κώδικα) καθορίζονται από τα /* (έναρξη σχολίου) και */ (λήξη σχολίου). Πρέπει οι χαρακτήρες /* και */ να σχηματίζουν ζεύγη, όμως δεν είναι απαραίτητο να βρίσκονται στην ίδια γραμμή.
- Ανάμεσα στα /* και */ μπορούν να περιέχονται αριθμοί, χαρακτήρες και σύμβολα.
- Ό,τι περιέχεται ανάμεσα στους χαρακτήρες /* ... */ ουσιατικά ‘προσπερνάται’ από τον compiler και αντιμετωπίζεται ως ένας χαρακτήρας κενού.
- Σκοπός του να γράφουμε σχόλια είναι να κάνουμε το πρόγραμμα πιο κατανοητό και καθαρό στην ανάγνωση καθώς παρέχουν εξηγήσεις για τμήματα του προγράμματος

Ο πρώτος κώδικας σε C – Header files

Η γραμμή **#include <stdio.h>** δηλώνει ότι το πρόγραμμα χρησιμοποιεί την πρότυπη βιβλιοθήκη εισόδου-εξόδου (**st**an**d**ard **i**nput/**o**utput).

- Ο όρος πρότυπη βιβλιοθήκη σημαίνει ότι το συγκεκριμένο αρχείο είναι ήδη προεγκεταστημένο και ουσιαστικά αποτελεί τμήμα του πακέτου της C.
- Η βιβλιοθήκη αυτή περιέχει τον ορισμό εντολών της C, όπως η **printf()** και η κλήση της επιτρέπει στον προγραμματιστή να τις χρησιμοποιεί.
- Το **stdio.h** ονομάζεται header file. Υπάρχουν πολλά διαφορετικά παρόμοια αρχεία που περιέχουν εντολές που μπορεί να χρησιμοποιήσει ο προγραμματιστής (π.χ. το math.h, που καλεί συναρτήσεις όπως το log(), το sin(), το cos(), κτλ).
- Καλούνται πάντα στην αρχή του προγράμματος, ώστε να καταστήσουν λειτουργικές τις εντολές, που περιέχουν, στη συνέχεια.

Ο πρώτος κώδικας σε C – Κυρίως μέρος του προγράμματος

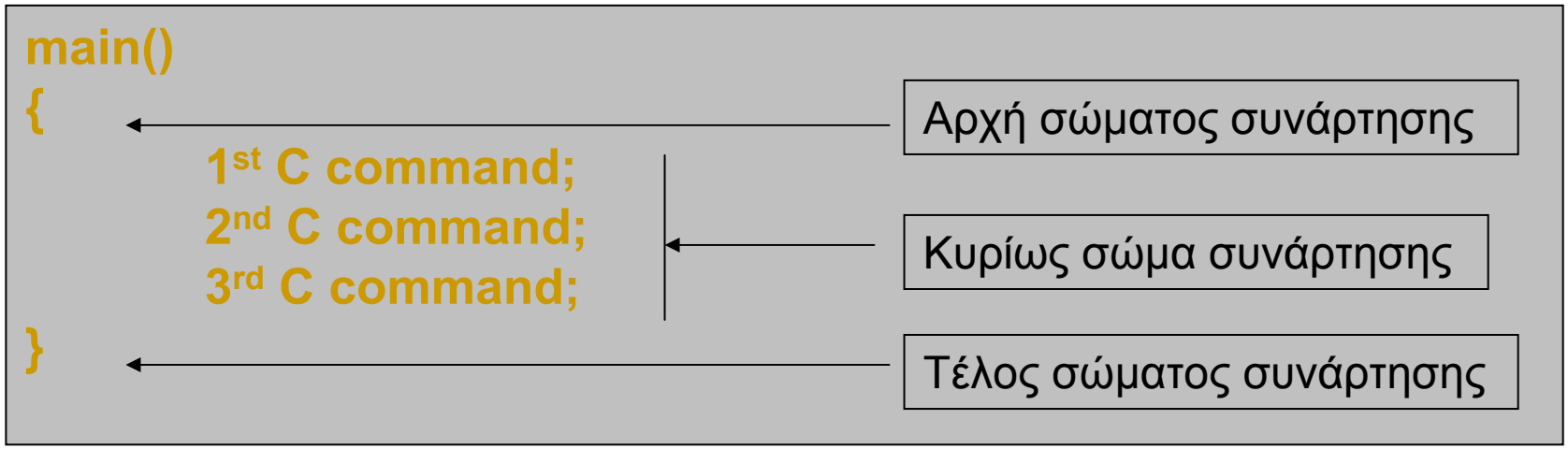
Το κυρίως πρόγραμμα αποτελείται από τις γραμμές

```
int main()
{
        printf("Hello World\n");
        return 0;
}
```

- Αυτές οι γραμμές ορίζουν μία συνάρτηση της γλώσσας C, η οποία αποτελεί μια ακολουθία μεμονομένων βημάτων του προγράμματος που έχουν ομαδοποιηθεί και στην οποία έχει δοθεί ένα όνομα.
- Η συνάρτηση main είναι η κύρια του προγράμματος, διατηρείται σταθερή και δεν επιτρέπεται στον προγραμματιστή να την αλλάξει. **Επομένως κάθε νέος κώδικας πρέπει απαραίτητα να περιέχει τη συνάρτηση main.**

Ο πρώτος κώδικας σε C – Κυρίως μέρος του προγράμματος

- Μετά τη γραμμή που περιέχει το όνομα της συνάρτησης, ακολουθεί το σώμα αυτής. Το σώμα των αγκίστρων { } είναι αυτό που καθορίζει το σώμα, που αποτελεί ένα κομμάτι κώδικα. Αυτό περιέχει δηλώσεις της γλώσσας προγραμματισμού και ορισμούς.

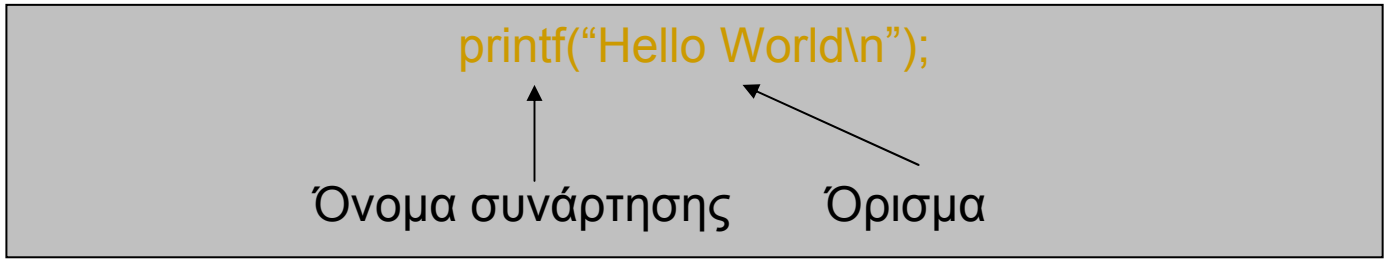


- Οι εντολές που περικλείει η συνάρτηση main εκτελούνται διαδοχικά, καθώς κινούμαστε από την αρχή του προγράμματος προς το τέλος του (πρώτα η 1st , μετά η 2nd ,κτλ).
- **Στο τέλος κάθε εντολής του προγράμματος βάζουμε ελληνικό ερωτηματικό (;)**

Ο πρώτος κώδικας σε C – Κλήση συνάρτησης

Το πρόγραμμα μέσα στη main() καλεί τη συνάρτηση **printf()** που βρίσκεται ήδη ορισμένη στο <stdio.h> και αποτελεί κομμάτι της C.

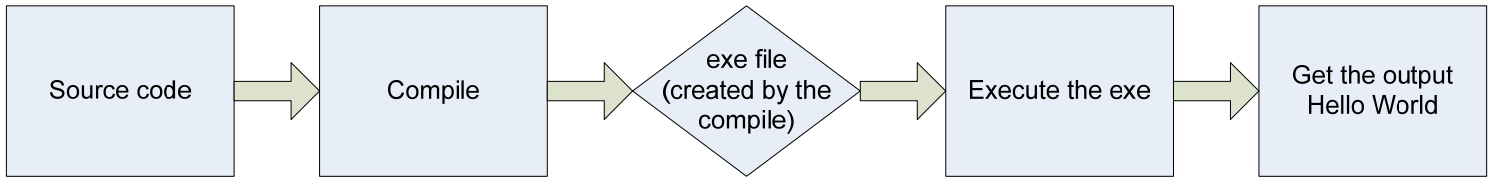
- Η κλήση της συνάρτησης δίνεται με το όνομά της ακολουθούμενο από το σώμα της, που ουσιαστικά αποτελεί το όρισμά της



- Η printf εκτυπώνει στην οθόνη το όρισμα που περιέχεται στο σώμα της, ουσιαστικά το Hello World που περιέχεται στο (΄΄ ... ΄΄).
- Το \ ορίζει ειδικούς χαρακτήρες για την printf(). Το \n αλλάζει γραμμή στην οθόνη και το \t αφήνει κενό 8 χαρακτήρων

Ο πρώτος κώδικας σε C – Εκτέλεση προγράμματος

Έχουμε λοιπόν ετοιμάσει το source code στον editor του προγράμματος και απομένει να το κάνουμε compile, να πάρουμε το εκτελέσιμο και στη συνέχεια να το τρέξουμε.



Αφού ακολουθήσουμε τα παραπάνω στάδια με επιτυχία, τότε το πρόγραμμα όταν τρέξει θα εκτυπώσει στην οθόνη το επόμενο μήνυμα

```
Hello World
```

Ο πρώτος κώδικας σε C – Εκτέλεση προγράμματος

Αντιγράψτε το πρόγραμμα της διαφάνειας 3 σε κάποιο διαθέσιμο editor, κάντε το compile, εκτελέστε το και δείτε να εκτυπώνει στην οθόνη το αποτέλεσμα

Στοιχεία της γλώσσας C

```
#include <stdio.h>
#include <math.h>

int main()
{
        int number1,number2,total_sum;
        number1=10;
        number2=20;
        printf("This program adds two numbers, which are %d,\t
       %d\n",number1,number2);
        total_sum=number1+number2;
        printf("The total sum is %d\n", total_sum);
        return 0;
}
```

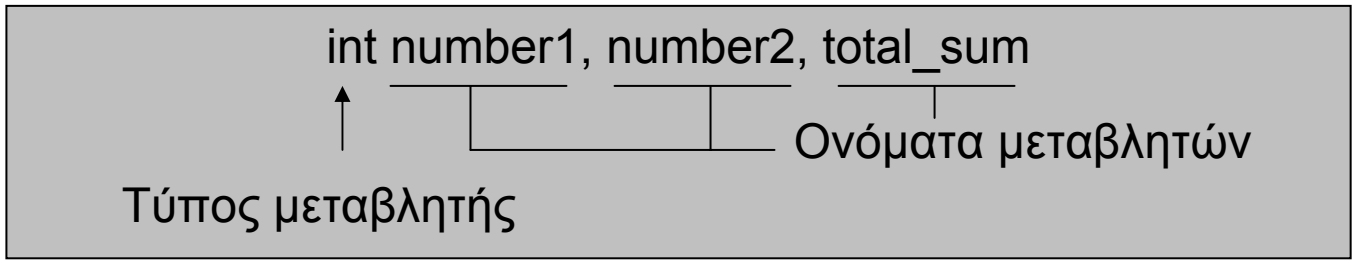
Στοιχεία της γλώσσας C

Στο νέο αυτό πρόγραμμα θέλουμε να προσθέσουμε δύο αριθμούς και να εκτυπώσουμε το αποτέλεσμα της άθορισης στην οθόνη.

- Ένα νέο στοιχείο είναι η δήλωση μεταβλητών (number1,number2 ,total_sum), η εκχώρηση κάποιας τιμής σε αυτές (number1=10).
- Επιπλέον επιθυμούμε το αποτέλεσμα του προγράμματος να είναι η εκτύπωση στην οθόνη μας των μεταβλητών αυτών.
- Τέλος θέλουμε να τις προσθέσουμε χρησιμοποιώντας τις κατάλληλες εντολές που διαθέτει η γλώσσα C.

Στοιχεία της γλώσσας C – Δήλωση μεταβλητών

Η γραμμή **int number1,number2,total_sum;** Περιέχει τη δήλωση των μεταβλητών του προγράμματος. Αυτές είναι οι number1,number2 και total_sum. Το int δηλώνει τον τύπο της μεταβλητής.



Οι κυριότεροι τύποι μεταβλητών είναι

- **int**, πρόκειται για ακέραιου τύπου που απαιτεί 16 bits μνήμης
- **double**, πρόκειται για μεταβλητή με υποδιαστολή για μη ακέραιους αριθμούς και απαιτεί 32 bits μνήμης

Στοιχεία της γλώσσας C – Έγκυρα ονόματα μεταβλητών

Δεν μπορούμε να δίνουμε σαν όνομα μιας μεταβλητής ό,τι επιθυμούμε. Συγκεκριμένα πρέπει να υπόκειται στους εξής περιορισμούς

- ο πρώτος χαρακτήρας πρέπει να είναι μη αριθμητικός.
- οι υπόλοιποι χαρακτήρες μπορούν να είναι από το a-z, A-Z, _ και τα ψηφία 0 – 9.
- δεν μπορούν να είναι ονόματα εντολών

Παραδείγματα μη έγκυρων μεταβλητών είναι τα εξής: 2apple, interest%, int, float, in come, one.two

Παραδείγματα έγκυρων μεταβλητών είναι τα εξής: apple2, interest, xint, xfloat, income, one_two

Στοιχεία της γλώσσας C – Εκχώρηση τιμής σε μεταβλητή

Ο ορισμός δημιουργεί τη μεταβλητή που επιθυμούμε (τύπος και όνομα) και της αποδίδει κάποια θέση μνήμης. Μπορούμε στη συνέχεια να εκχωρήσουμε κάποια τιμή στη μεταβλητή αυτή, η οποία θα αποθηκευτεί σε αυτή τη θέση.

Η γραμμή number1=10 εκχωρεί στη μεταβλητή number1 την τιμή 10.

Στη γλώσσα C, μια απλή εντολή εκχώρησης είναι της μορφής

όνομα_μεταβλητής = τιμή;

Είναι δυνατό να εκχωρηθεί τιμή σε κάποια μεταβλητή κατά τη διάρκεια της εκτέλεσης του προγράμματος, όπως συμβαίνει με την total_sum που της αποδίδεται το άθροισμα των number1 και number2 που ορίστηκαν από τον χρήστη.

Στοιχεία της γλώσσας C – Εκτύπωση μεταβλητών

Στο συγκεκριμένο πρόγραμμα η printf() εκτυπώνει στην οθόνη τις τιμές των μεταβλητών number1 και number2.

```
printf("This program adds two numbers, which are %d, \t %d\n",number1,number2);
```

• 'Οταν κατά την εκτέλεσή της η printf συναντά τα \n, \t αντιλαμβάνεται ότι πρέπει να αλλάξει γραμμή και να αφήσει ένα κενό.

• Όταν συναντά ένα χαρακτήρα της μορφής %d αντιλαμβάνεται ότι πρέπει να εκτυπώσει μία μεταβλητή τύπου int. Το ποια θα εκτυπώσει καθορίζεται από τις μεταβλητές που είναι γραμμένες μετά τα ΄΄ ...΄΄. Στο παράδειγμα θα εκτυπώσει το εξής

```
This program adds two numbers, which are 10,       20.
```

Όταν οι μεταβλητές είναι τύπου double τότε πρέπει να χρησιμοποιήσουμε το %lf, αντί για %d που αντιστοιχεί σε αυτές τύπου int.

Στοιχεία της γλώσσας C – Εκτέλεση μαθηματικών πράξεων

Η γραμμή **total_sum=number1+number2;** δίνει την εντολή για την εκτέλεση της πράξης της άθροισης και της εκχώρησης του αποτελέσματος στην μεταβλητή total_sum.

Τελεστής	Ονομασία
+	Πρόσθεση
-	Αφαίρεση
*	Πολ/σμός
/	Διαίρεση
=	Εκχώρηση

Αριθμητικοί τελεστές που χρησιμοποιούνται στη γλώσσα C.

Σε περίπτωση που οι υπολογισμοί πραγματοποιούνται ανάμεσα σε μεταβλητές τύπου int και double, το αποτέλεσμα είναι πάντα double.

Στοιχεία της γλώσσας C – Διαίρεση

Όταν πραγματοποιούνται πράξεις μεταξύ μεταβλητών τύπου int, τότε το αποτελέσμα είναι πάντα του ίδιου τύπου. Αυτό δημιουργέι ένα παράδοξο, όταν έχουμε να κάνουμε με τη διαίρεση μεταβλητών int, καθώς και σε αυτή την περίπτωση οδηγεί σε αποτέλεσμα τύπου int.

Το παράδοξο μπορεί να γίνει κατανοητό εαν εκτελέσουμε τη διαίρεση ½, η οποία οδηγεί σε αποτέλεσμα ίσο προς 1 και όχι 0.5. Το υπόλοιπο της διαίρεσης αγνοείται ουσιαστικά και στρογγυλοποιείται προς τον πλησιέστερο ακέραιο.

Για τον σωστό υπολογισμό μπορούμε να γράψουμε 1.0 / 2.0 ή 1 / 2. ή 1 / (double 2).

Στοιχεία της γλώσσας C – Εκτέλεση προγράμματος

Αντιγράψτε το πρόγραμμα της διαφάνειας 11, ορίσετε δύο νέες μεταβλητές (ονομάστε τες όπως επιθυμείτε), εκχωρήστε τους κάποια τιμή και στη συνεχεια

- προσθέστε και αυτές στο total_sum
- πολλαπλασιάστε τες με τις number1, number2
- εκτυπώστε τα αποτελέσματά σας στην οθόνη.

Στοιχεία της γλώσσας C – Αποθήκευση σε αρχείο

```
#include <stdio.h>
#include <math.h>

int main()
{
        int number1,number2,total_sum;
        FILE* myfile=fopen("Data.dat","w");
        number1=10;
        number2=20;
        printf("This program adds two numbers, which are %d,\t
        %d\n",number1,number2);
        total_sum=number1+number2;
        fprintf(myfile,"The total sum is \t %d\n",total_sum);
        fclose(myfile);
        return 0;
}
```

Στοιχεία της γλώσσας C – Αποθήκευση σε αρχείο

Το πρόγραμμα αυτό διαφοροποιείται από το προηγούμενο στο ότι του ζητάμε να αποθηκεύσει το αποτέλεσμα της άθροισης σε ένα αρχείο με το όνομα data.dat. Οι νέες εντολές είναι οι εξής

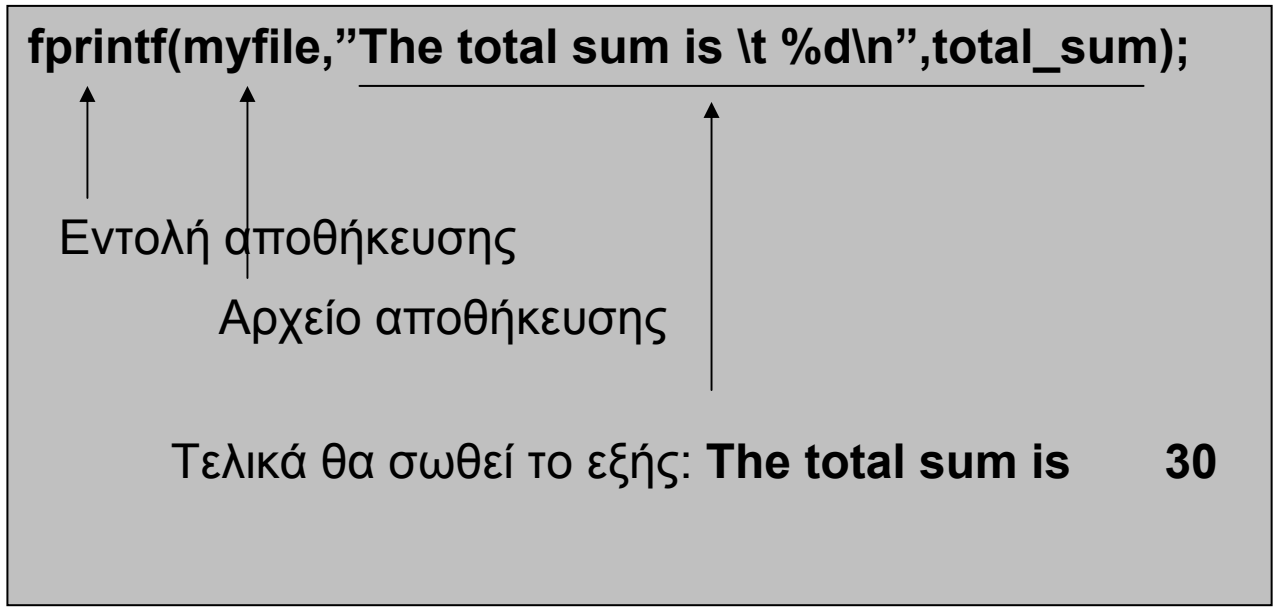
```
FILE* myfile=fopen(“Data.dat”,”w”);

….
fprintf(myfile,”The total sum is \t %d\n”,total_sum);
fclose(myfile);
```

Η γραμμή **FILE* myfile=fopen(“Data.dat”,”w”);** ουσιαστικά δημιουργεί μία μεταβλητή τύπου FILE, την myfile. Αυτή με τη σειρά της ανοίγει / δημιουργεί μέσω της εντολής fopen το αρχείο που επιθυμούμε, το Data.dat. Το .dat αναφέρεται στο format του αρχείου που μπορεί να είναι διαφορετικό, ανάλογα με τις επιθυμίες του προγραμματιστή (π.χ. τύπου .txt). Το ”w” δηλώνει ότι είναι αρχείο προς εγγραφή.

Στοιχεία της γλώσσας C – Αποθήκευση σε αρχείο

Αφού δημιουργήθηκε το αρχείο, φροντίζουμε να αποθηκεύσουμε σε αυτό όσα δεδομένα θεωρούμε αναγκαία. Η γραμμή **fprintf(myfile,"The total sum is \t %d\n",total_sum);** Καθορίζει το αρχείο αποθήκευσης (myfile) και το τι θα σωθεί σε αυτό (το περιεχόμενο των ΄΄ ... ΄΄).



Δομή ελέγχου if

- **Γενική μορφή : if (*έκφραση ελέγχου*) *εντολή* ;**

  Εάν θέλουμε να εκτελεστούν πάνω από μια εντολή χρειάζονται τα { }

```
if (έκφραση ελέγχου)
{
    εντολή 1;
    εντολή 2;
    ...
}
```

- **Η έκφραση ελέγχου συγκρίνει τιμές δυο η περισσότερων αριθμητικών εκφράσεων (είναι δηλαδή μια λογική έκφραση) και παράγει ένα αποτέλεσμα *αληθές* η *ψευδές. Εάν το αποτέλεσμα είναι αληθές τότε εκτελείται η εντολή, αλλιώς όχι.***
- **Η γενική μορφή της έκφρασης ελέγχου είναι :**

  (*αριστερό μέλος τελεστής σχέσης δεξιό μέλος*)
- **Τελεστές σχέσης :**

  < μικρότερο
  <= μικρότερο ή ίσο
  == ίσο
  > μεγαλύτερο
  >= μεγαλύτερο ή ίσο
  != διαφορετικό (όχι ίσο)

Δομή ελέγχου if – Παράδειγμα

```
#include <stdio.h>

int main ()
{
   int guess,jackpot=8;
   printf (“ Find the jackpot number which is between 1 and 10 \n Enter a number: \n”);
   scanf (“%d”,&guess);
   if (guess<jackpot) printf (“Try a larger number \n”);
   if (guess>jackpot) printf (“Try a smaller number \n”);
   if (guess==jackpot) printf (“ This is the good number. Enter it again \n”);
   scanf (“%d”,&guess);
   if (guess==jackpot) printf (“JACKPOT!\n”);
   if (guess!=jackpot)
   {
       printf (“You didn’t find the correct number! \n”);
       printf (“You have lost! \n”);
   }
   return 0;
}
```

Δομή ελέγχου if – Επεξήγηση του παραδείγματος

- Αρχικά δηλώνονται 2 μεταβλητές τύπου integer (int), η guess και η jackpot, στην jackpot δίνεται τιμή 8.
- Με την γνωστή μας πλέον printf τυπώνονται δυο φράσεις στην οθόνη η μια κάτω από την άλλη (χάρη στα σύμβολα \n) που μας ζητάνε να εισάγουμε έναν αριθμό μεταξύ 1 και 10.
- Η scanf είναι μια συνάρτηση της βιβλιοθήκης stdio.h η οποία διαβάζει από την οθόνη κάποιον χαρακτήρα (η και περισσότερους) και δίνει τιμή σε μια μεταβλητή (η περισσότερες) του προγράμματος . Όπως κάθε συνάρτηση της γλώσσας C δέχεται παραμέτρους εντός παρενθέσεων (). Στο πρώτο σκέλος της scanf (“%d”,&guess); μεταξύ των “ “ υπάρχει το κομμάτι που η συνάρτηση αντιλαμβάνεται ότι πρέπει να διαβάσει από την οθόνη (εδώ με %d αντιλαμβάνεται ότι θα διαβάσει έναν ακέραιο αριθμό). Στο δεύτερο σκέλος, με το σύμβολο & γίνεται η απόδοση της τιμής που διαβάστηκε, στην μεταβλητή που δηλώνεται (στην περίπτωση μας η guess).

  Σημείωση: Όπως υπάρχει η fprintf που τυπώνει σε αρχείο, υπάρχει η fscanf που διαβάζει από αρχείο, για παράδειγμα: fscanf(my_file,”%d %lf”,&kapio_integer,&kapio_double)
- Στην συνεχεία οι τρεις εντολές if εξετάζουν τις τρεις πιθανές περιπτώσεις σύγκρισης των guess και jackpot, και ανάλογα τυπώνονται στην οθόνη τα αντίστοιχα μηνύματα.
- Ξανακαλείται η scanf για να διαβαστεί η δεύτερη προσπάθεια του χρηστή. Η guess έχει πια σαν τιμή αυτήν που πληκτρολογεί ο χρηστής στην δεύτερη προσπάθεια.
- Οι 2 επόμενες εντολές if εξετάζουν μόνο τις 2 εξής περιπτώσεις, (α) βρέθηκε ο σωστός αριθμός (if (guess==jackpot)), (β) δεν βρέθηκε (if (guess!=jackpot)) και ανάλογα τυπώνονται στην οθόνη τα αντίστοιχα μηνύματα.

Δομή ελέγχου if - else

- **Σύνταξη μιας απλής εντολής if – else:**

```
if (έκφραση ελέγχου)
{
   εντολή 1a;
   εντολή 1b;
   …
 }
else
{
   εντολή 2a;
   εντολή 2b;
   …
 }
```

που διαβάζεται

«*Εάν η έκφραση ελέγχου δίνει αποτέλεσμα αληθές, τότε εκτέλεσε την ομάδα εντολών 1 αλλιώς εκτέλεσε την ομάδα εντολών 2*».

Δομή ελέγχου if - else if

- **if (*έκφραση ελέγχου 1*)**
  {
  ομάδα εντολών 1
  }
  else if (*έκφραση ελέγχου 2*)
  {
  ομάδα εντολών 2
  }
  else if (*έκφραση ελέγχου 3*)
  {
  ομάδα εντολών 3
  }
  else
  {
  τελική ομάδα εντολών
  }

Λογικοί τελεστές

Η γλώσσα C διαθέτει 3 λογικούς τελεστές που κατά σειρά προτεραιότητας είναι :

- ! Λογικό ΟΧΙ (NOT)
- && Λογικό ΚΑΙ (AND)
- || Λογικό Η (OR)

Παραδείγματα :

- if (x>0 && y<=0) { …}
- if (x==0 || y==0) {…}
- if (!(x==y)) {…}

Σχόλια :

- *Στην γλώσσα C το αληθές (TRUE) αντιστοιχεί σε οποιοδήποτε αριθμητική τιμή διάφορη του μηδενός ενώ το ψευδές (FALSE) αντιστοιχεί στο 0. Έτσι η εντολή* if (a!=0) *μπορεί να γραφτεί απλώς* if (a) .

- *Να μην ξεχνάμε να βάζουμε παρενθέσεις σε όλες τις σχέσεις για να μην κάνουμε λάθη με τις προτεραιότητες.*

Βρόγχοι έλεγχου do while, while

- **do while**

do { ομάδα εντολών } while (*έκφραση ελέγχου*)

```
int count=0;
do
{
    count++;
    printf(“%d \n“,count);
}
while (count < 10);
```

- **while**

while (*έκφραση ελέγχου*) { ομάδα εντολών }

```
int count=0;
while (count < 10)
{
    count++;
    printf(“%d\n“,count);
}
```

Βρόγχοι έλεγχου do while, while

- Άσκηση : Γράψτε ένα πρόγραμμα που να τυπώνει δέκα φορές το «Hello world!» στην οθόνη.

- Διαφορά μεταξύ do while και while:
  Με το do while ο βρόγχος εκτελείται τουλάχιστον μια φορά.
  Ελέγξτε το βάζοντας μια έκφραση ελέγχου με ψευδή αποτέλεσμα.

- Ο βρόγχος while είναι ο πιο διαδεδομένος.

- Οι βρόγχοι do while και while επιτρέπουν κυρίως να εκτελέσεις εντολές για αδιευκρίνιστο αριθμό επαναλήψεων ώσπου να διαψευστεί η συνθήκη ελέγχου, σε αντίθεση με το for όπου ο αριθμός των επαναλήψεων είναι προκαθορισμένος.

Βρόγχος έλεγχου while– Παράδειγμα

```
#include <stdio.h>

int main ()
{
   int guess=0;
   int jackpot=8;
   printf (" Find the jackpot number which is between 1 and 10 \n Enter a number:
     \n");
   while (guess!=jackpot)
   {
        scanf ("%d",&guess);
        if (guess==jackpot) printf ("JACKPOT!\n");
        if (guess!=jackpot) printf ("You didn't find the correct number! Retry! \n");
    }
    return 0;
}
```

Βρόγχος έλεγχου for

- **Γενική σύνταξη του for**

for (μετρητής = αρχική τιμή; μετρητής < τελική τιμή; μετρητής++;) { ομάδα εντολών }

- **++ σημαίνει «αύξησε κατά ένα»**

- **Ο βρόγχος for εκτελεί μια ομάδα εντολών για ορισμένο αριθμό βημάτων αυξάνοντας κατά ενα κάθε φορά τον μετρητή .**

- **Παράδειγμα**

```
#include <stdio.h>
 int main()
 {
     int i;
     for ( i=0; i < 10; i++ ) printf("%d \n",i);
     return 0;
 }
```

Βρόγχος έλεγχου for– Παράδειγμα

```
#include <stdio.h>

int main ()
{
   int i,x,y;
   for (i=0;i<20;i++ )
   {
       x=i;
       y=x*x+2;
       printf("%d    %d\n",x,y);
   }
    return 0;
}
```

- Άσκηση1: Τυπώστε στην οθόνη δέκα τιμές των x και y όπου y=exp(x/2) για x από 0 μέχρι 1.
- Άσκηση 2: Τυπώστε σε αρχείο τα αποτελέσματα.

Στοιχεία της γλώσσας C – Πίνακες

```
#include <stdio.h>
#include <math.h>
#define N 100                                   // Define the number of the elements of the tables

/* We want to exchange dollars for euros. The currency rate is 1.3 and therefore it is 1euro = 1.3*dollar */
int main( )
{
      int i;
      double currency_rate=1.3;                 /* 1euro = 1.3dollars */
      double euro[N], dollar[N];                /* Define the tables */
      FILE* myfile=fopen("Euros-Dollars.dat","w");  /* Create the file Euros-Dollars.dat */
      for(i=0;i<N;i++){                         /* Initialize the tables */
            euro[i]=0.;
            dollar[i]=0.;
      }
      for(i=0;i<N;i++){                         /* Assign values to the table dollar */
            dollar[i]=i.;
      }
      for(i=0;i<N;i++){                         /* Calculate the euros that correspond to the dollars */
            euro[i]=currency_rate*i.;
      }
      for(i=0;i<N;i++){                         /* Print the euros and the dollars */
            printf("%lf euros correspond to %lf dollars\n",euro[i],dollar[i]);
      }
      for(i=0;i<N;i++){                         /* Save the euros and the dollars */
            fprintf(myfile,"%lf \t %lf\n",euro[i],dollar[i]);
      }
      fclose(myfile);
      return 0;
}
```

Στοιχεία της γλώσσας C – Πίνακες

Το πρόγραμμα αυτό μετατρέπει για εσάς ένα ποσό ευρώ στο αντίστοιχό του σε δολλάρια. Εαν υποθέσουμε ότι η ισοτιμία ευρώ προς δολλάριο είναι ίση προς 1.3, τότε η μετατροπή γίνεται ώς εξής

1 ευρώ = 1.3 * 1 δολλάριο

Επιπλέον το πρόγραμμα εκτυπώνει στην οθόνη τις τιμές αυτές και στη συνέχεια τις αποθηκεύει σε κάποιο αρχείο.

• Η γραμμή #define N 100 μας επιτρέπει να ορίζουμε global μεταβλητές, οι οποίες διατηρούνται σταθερές κατά την εκτέλεση του προγράμματος. Ουσιαστικά πρόκειται για μεταβλητές που τις ορίζουμε μία φορα και στη συνέχεια δεν μπορούμε να τις αλλάξουμε και επομένως όπου εμφανίζεται το N στον κώδικα θα ισούται με 100.

Στοιχεία της γλώσσας C – Πίνακες

Ένα από τα σημαντικότερα χαρακτηριστικά της C, είναι η δυνατότητα που προσφέρει στον χρήστη να ορίζει *δομές δεδομένων,* γνωστές ως πίνακες.

• Ένας πίνακας είναι μία συλλογή μεταβλητών του ίδιου τύπου (π.χ. int ή double), οι οποίες αποθηκεύονται σε αύξουσες θέσεις μνήμης. Ουσιαστικά με τον τρόπο αυτό ορίζει κανείς ταυτόχρονα πολλές μεταβλητές, χωρίς να χρειάζεται να τις ορίζει μία μία.

• Οι πίνακες δίνουν στον προγραμματιστή τη δυνατότητα να εκχωρεί τιμές στα στοιχεία – μεταβλητές του πίνακα άμεσα, χρησιμοποιώντας μία επαναληπτική διαδικασία (π.χ. με ένα for).

• Παράλληλα επιτρέπουν στον προγραμματιστή να έχει πρόσβαση σε όλες αυτές τις τιμές άμεσα, χρησιμοποιώντας μία επαναληπτική διαδικασία (π.χ. με ένα for). Με τον τρόπο αυτό μπορεί κανείς να εκτυπώσει εύκολα όλες τις τιμές ενός πίνακα ή να τις αποθηκεύσει σε ένα αρχείο.

Στοιχεία της γλώσσας C – Πίνακες

Η γραμμή **double euro[N], dollar[N];** ορίζει δύο μονοδιάστατους πίνακες τύπου double, τους euro[N] και dollar[N].

• Στη C οι πίνακες γράφονται με τετράγωνες αγκύλες που περιέχουν μία θετική ακέραια τιμή (π.χ. N, το οποίο έχει οριστεί να είναι 100).

• Η τιμή που περικλείεται ανάμεσα στις αγκύλες ονομάζεται δείκτης (index). Είναι ενδεικτική του αριθμού των μεταβλητών που θα δημιουργηθούν και θα αποτελούν τα στοιχεία του πίνακα. Στην περίπτωση του προγράμματος που εξετάζουμε, για καθένα από τους πίνακες euro και dollar θα δημιουργηθούν N=100 διαφορετικές μεταβλητές.

• Οι μεταβλητές που δημιουργούνται είναι οι euro[0], euro[1], euro[2], …, euro[N-1] και αντιστοίχως οι dollar[0], dollar[1], dollar[2], …, dollar[N-1]. Όλες είναι μεταβλητές τύπου double, όπως είχε οριστεί εξαρχής.

• **Στη C, εαν ορίσουμε ένα πίνακα με N στοιχεία, τότε θα δημιουργηθούν N μεταβλητές, των οποίων ο δείκτης θα μεταβάλλεται από 0 έως N-1 και όχι από 1 έως N**

Η σύνταξη ενός μονοδιάστατου πίνακα είναι

τύπος_στοιχείων όνομα_πίνακα [πλήθος_στοιχείων]

Στοιχεία της γλώσσας C – Πίνακες

Στο επόμενο κομμάτι του προγράμματος ουσιαστικά αρχικοποιούμε τους πίνακες μας, αποδίδοντας σε καθένα από τα στοιχεία τους την τιμή 0.

```
for(i=0;i<N;i++){          /* Initialize the tables */
        euro[i]=0.;
        dollar[i]=0.;
}
```

Χρησιμοποιούμε την επαναληπτική διαδικασία for, η οποία αναλαμβάνει για κάθε τιμή του i, να αποδώσει στο αντίστοιχο στοιχείο του πίνακα, π.χ. euro[i], την τιμή 0. Ουσιαστικά λειτουργεί ως εξής

```
Για i=0 εκτελεί τις εκχωρήσεις euro[0]=0 και dollar[0]=0
Για i=1    #      #      #          euro[1]=0 και dollar[1]=0
Για i=2    #      #      #          euro[2]=0 και dollar[2]=0
…
Για i=N-1 #      #      #          euro[N-1]=0 και dollar[N-1]=0
```

Η αρχικοποίηση είναι μία διαδικασία που πρέπει πάντα να εφαρμόζουμε στα προγράμματα μας

Στοιχεία της γλώσσας C – Πίνακες

Στο επόμενο κομμάτι του προγράμματος ουσιαστικά εκχωρούμε τιμές στους πίνακες μας. Η διαδικασία είναι παρόμοια με αυτή της εκχώρησης τιμής σε μία μεταβλητή, απλά με τη διαδικασία των επαναληπτικών βρόγχων, μπορούμε ταυτόχρονα να αποδίδουμε τιμές στα στοιχεία των πινάκων.

```
for(i=0;i<N;i++){
            euro[i]=i.;
}
```


Για i=0 είναι euro[0]=0
Για i=1 # euro [1]=1
Για i=2 # euro [2]=2
…
Για i=99 # euro [99]=99

```
for(i=0;i<N;i++){
            dollar[i]=currency_rate*euro[i];
}
```


Για i=0 είναι dollar[0]=0
Για i=1 # dollar[1]=1.3
Για i=2 # dollar[2]=2.6
…
Για i=99 # dollar[99]=128.7

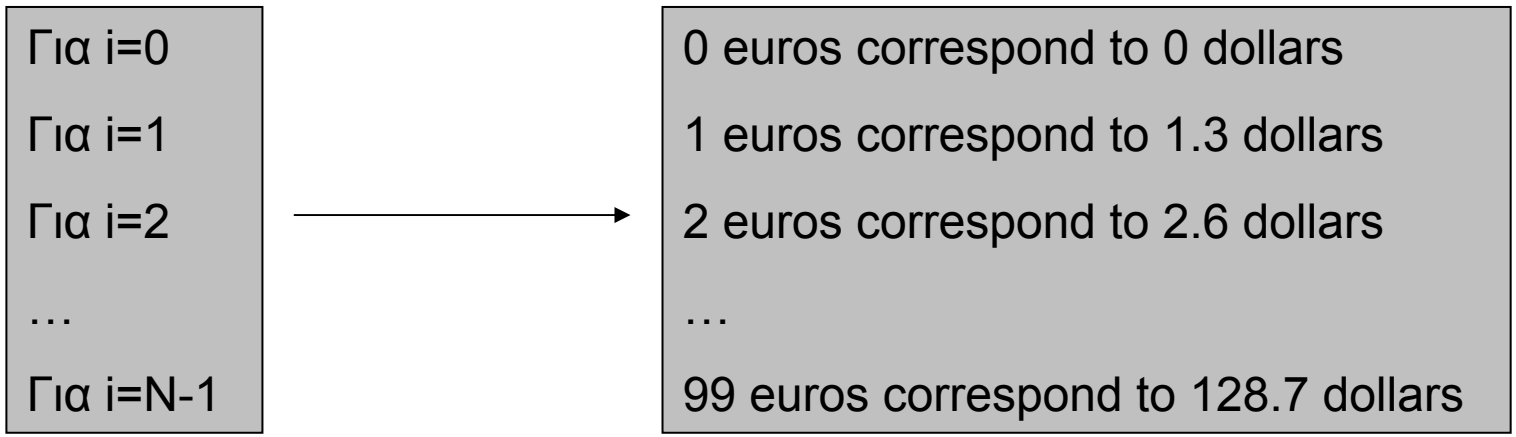
Στον πίνακα euro ορίζουμε την ποσότητα σε εύρω και στον πίνακα dollar παίρνουμε το αντίστοιχο ποσό σε δολλάρια που προκύπτει μετά την μετατροπή.

Στοιχεία της γλώσσας C – Πίνακες

Στη συνέχεια εκτυπώνουμε στην οθόνη όλα τα στοιχεία των δύο πινάκων

```
for(i=0;i<N;i++){
        printf("%lf euros correspond to %lf dollars\n",euro[i],dollar[i]);
}
```

Για κάθε τιμή του i (από i=0 μέχρι i=N-1), εκτυπώνονται στην οθόνη του υπολογιστή οι παρακάτω φράσεις

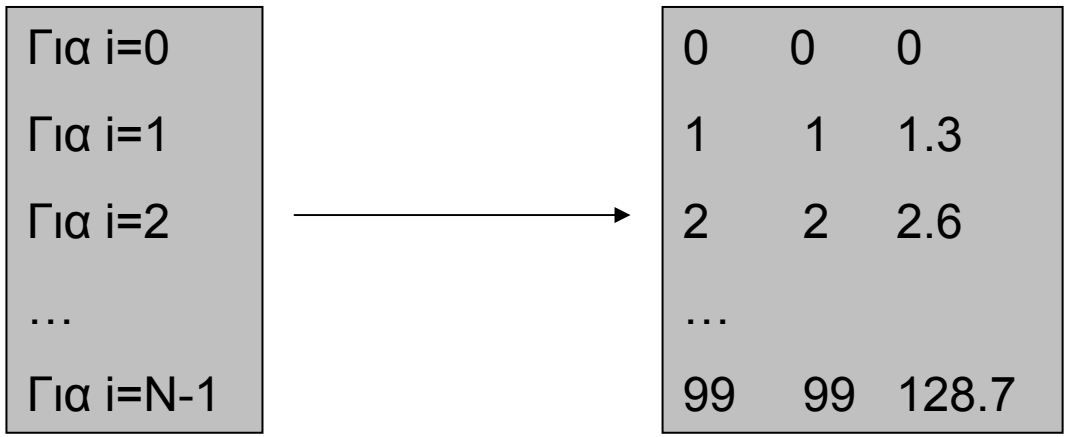


Στοιχεία της γλώσσας C – Πίνακες

Στη συνέχεια αποθηκεύουμε τα αποτελέσματα των υπολογισμών μας στο αρχείο Euros-Dollars.dat. Για τη ακρίβεια δημιουργούνται τρεις στήλες, η πρώτη με τις τιμές του i, η δεύτερη με τα ευρώ και η τρίτη με τα αντίστοιχα δολλάρια

```
for(i=0;i<N;i++){    /* Save the euros and the dollars */
            fprintf(myfile,"%d \t %lf \t %lf\n",i,euro[i],dollar[i]);
}
```

Για κάθε τιμή του i (από i=0 μέχρι i=N-1), αποθηκεύονται τα εξής στο αρχείο



Στοιχεία της γλώσσας C – Πίνακες

Άσκηση

Γράψτε ένα πρόγραμμα το οποίο να υπολογίζει το διάστημα x που διανύει κάθε δευτερόλεπτο t ένα σώμα το οποίο κινείται με σταθερή ταχύτητα v.

Υποθέτουμε ότι η ταχύτητα του σώματος είναι v=5 m/s και ότι κινείται για

$t_{total=}$=100 s.

Ζητούνται

1. Να εκτυπώνει το πρόγραμμά σας στην οθόνη τα διαστήματα που προκύπτουν για χρόνο μεγαλύτερο από 20 δευτερόλεπτα.
2. Να αποθηκεύει σε ένα αρχείο δύο στήλες. Η πρώτη θα είναι ο χρόνος (από 0 έως 100) και η δεύτερη θα είναι το διάστημα.
3. Να ανοίξετε το αρχείο που δημιουργέι το πρόγραμμά σας με κάποιο πρόγραμμα διαχείρισης δεδομένων (Origin, Excel) και νά κάνετε τη γραφική παράσταση του διαστήματος προς το χρόνο.

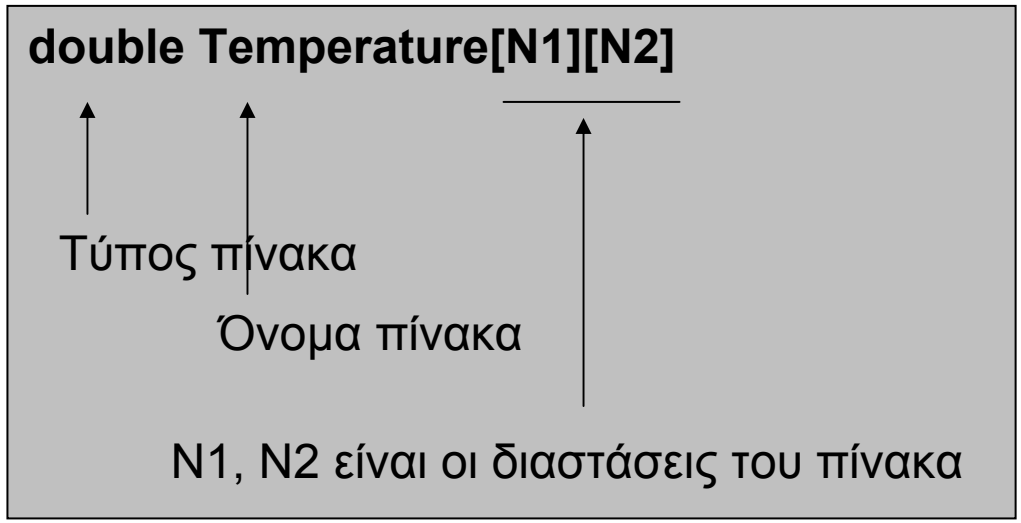
Στοιχεία της γλώσσας C – Πίνακες πολλών διαστάσεων

```
#include <stdio.h>
#include <math.h>
#define N1 24                                   // Define the number of the hours of a day
#define N2 60                                   // Define the number of the minutes of an hour

/* We want to calculate the temperature for every minute of enery hour in one day*/
int main( )
{
      int hour,sec;
      double Temperature[N1][N2];               /* Define the table Temperature*/
      for(hour=0;hour<=N1;hour++){              /* Initialize the table Temperature */
            for(sec=1;sec<=N2;sec++){
                  Temperature[hour][sec]=0.;
            }
      }
      for(hour=0;hour<=N1;hour++){
            for(sec=0;sec<=N2;sec++){
                  Temperature[hour][sec]=i*j;         /* Assign values to the table Temperature */
            }
      }
      for(i=0;i<N;i++){                         /* Print the table Temperature */
            for(j=0;j<N;j++){
                  printf("At the %d minute of the %d hour the temp was %d\n",sec,hour,Temperature[hour][sec]);
            }
      }
      return 0;
}
```

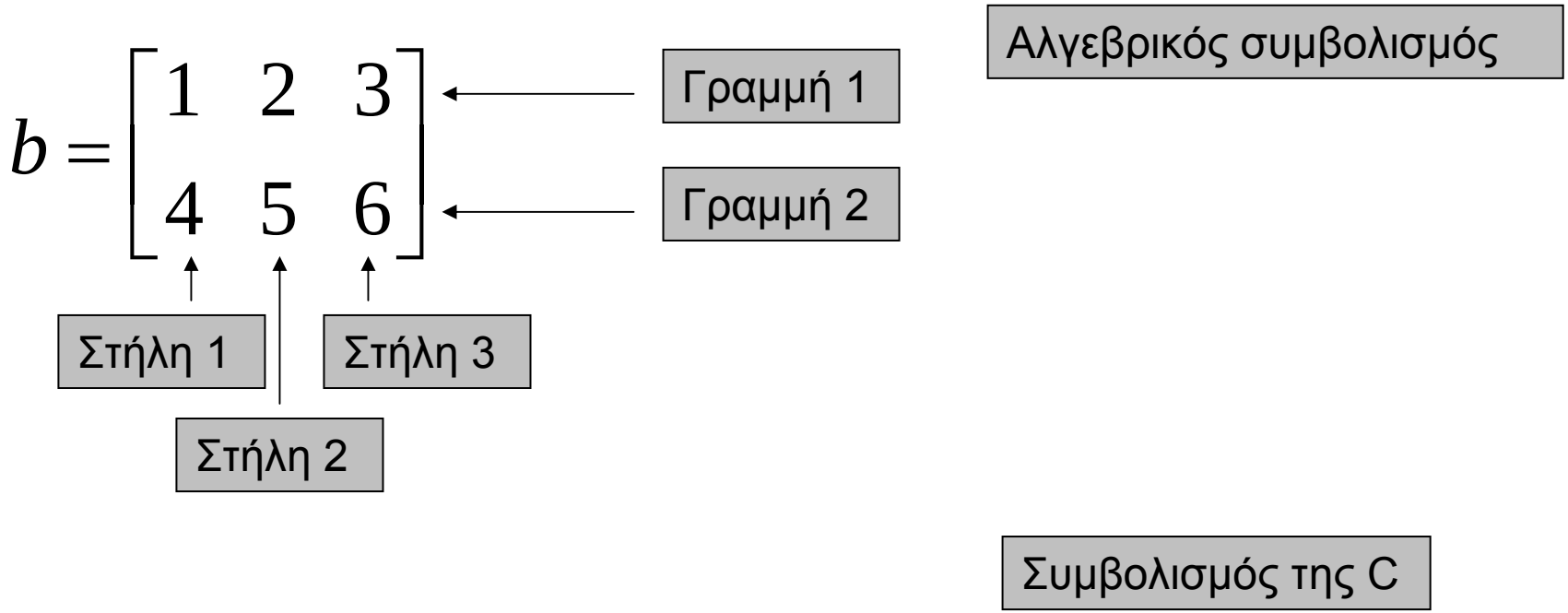
Στοιχεία της γλώσσας C – Πίνακες πολλών διαστάσεων

Σε πολλές περιπτώσεις οι πίνακες μίας διάστασης δεν είναι αρκετοί για να τους χρησιμοποιήσουμε. Για το λόγο αυτό παρέχεται η δυνατότητα στον προγραμματιστή να ορίζει πίνακες με περισότερες από μία διαστάσεις.Ο ορισμός είναι παρόμοιος με ένα μονοδιάστατο πίνακα



Στοιχεία της γλώσσας C – Πίνακες πολλών διαστάσεων

Ιδεατή εικόνα ενός πίνακα b[2][3]



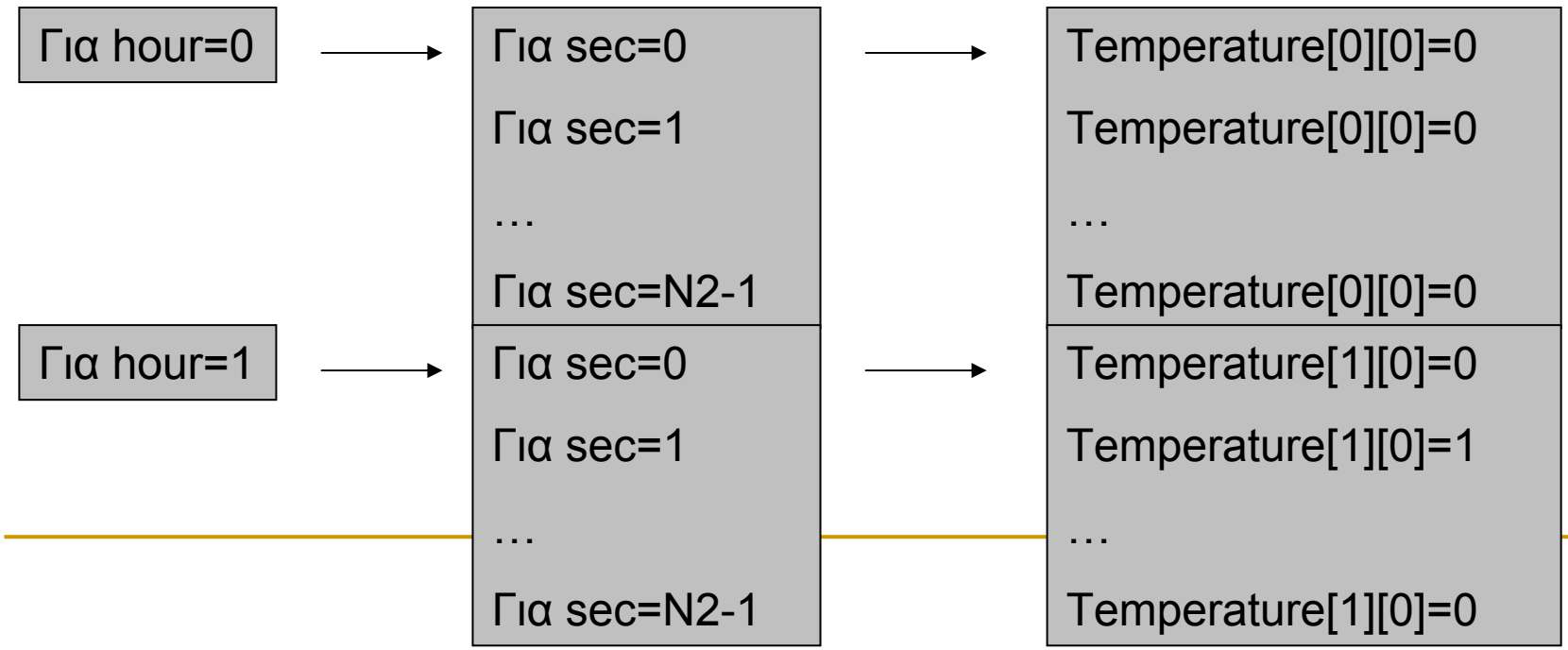
Πίνακας int b[2][3], όπου b[0][0]=1, b[0][1]=2, b[0][2]=3

b[1][0]=4, b[1][1]=5, b[1][2]=6

Στοιχεία της γλώσσας C – Πίνακες πολλών διαστάσεων

Για να αρχικοποιήσουμε ένα πίνακα δύο διαστάσεων, χρειαζόμαστε δύο επαναληπτικές διαδικασίες, μία για κάθε διάσταση

```
for(hour=0;hour<N1;hour++){
                    for(sec=1;sec<N2;sec++){
                                Temperature[hour][sec]=0.;
                    }
          }
```



Στοιχεία της γλώσσας C – Γεννήτρια τυχαίων αριθμών

```
#include <stdio.h>
#include <stdlib.h>                    // Include the stdlib header file


int main()
{
   int random_number,seed=100;  // Define the seed
   srand(seed);                          // Initialize the random number generator
   random_number=rand();           // Produce a random number
   printf("Random_number=%d\n”, random_number);
   printf(“RAND_MAX = %d\n",RAND_MAX);
   getchar();
   return 0;
}
```

Στοιχεία της γλώσσας C – Γεννήτρια τυχαίων αριθμών

Κάθε γεννήτρια τυχαίων αριθμών (**random number generator**) παράγει τυχαίους αριθμούς σε ένα συγκεκριμένο διάστημα. Στη C υπάρχει ήδη ΄έτοιμη προς χρήση΄ η γεννήτρια τυχαίων αριθμών **rand().**

Για να χρησιμοποιήσετε την **rand()** θα χρειαστεί

1. Να συμπεριλάβετε το επόμενο header file

```
#include <stdlib.h>
```

2. Να ορίσετε τη μεταβλητή **seed**, τύπου int

```
int random_number,seed=100;
```

στην οποία μπορείτε να εκχωρήσετε οποιαδήποτε αυθαίρετη τιμή επιθυμείτε

3. Να αρχικοποιήσετε την γεννήτρια τυχαίων αριθμών χρησιμοποιώντας την συνάρτηση

```
srand(seed);
```

Στοιχεία της γλώσσας C – Γεννήτρια τυχαίων αριθμών

Στη συνέχεια αρκεί να καλέσετε τη συνάρτηση **rand()** για να σας επιστρέψει κάποιον τυχαίο αριθμό

```
random_number=rand();
```

Σε αυτή την περίπτωση, κάθε φορά που θα καλείται η rand(), θα επιστρέφει ένα διαφορετικό τυχαίο αριθμό και θα εκχωρεί την τιμή αυτού του αριθμού στην μεταβλητή **random_number**.

Για τη **σωστή χρήση της rand()** πρέπει να γνωρίζουμε ότι

1. Επιστρέφει τυχαίες τιμές με περιοδικότητα, οπότε από ένα σημείο και μετά για συγκεκριμένο seed, επαναλαμβάνει τους τυχαίους αριθμούς
2. Θα πρέπει να αλλάζουμε τακτικά το seed καθώς εκτελείται το πρόγραμμα, ώστε να μην ολοκληρώσει μία περίοδο η γεννήτρια και επιστρέφει τις ίδιες τιμές.

Στοιχεία της γλώσσας C – Γεννήτρια τυχαίων αριθμών

Η μεταβλητή **RAND_MAX** ορίζει τη μέγιστη τυχαία τιμή που μπορεί να δώσει η γεννήτρια τυχαίων αριθμών. Στην περίπτωση που εκτελέσετε το πρόγραμμα της διαφάνειας 47 με seed = 100 , θα πάρετε ως output στην οθόνη το εξής

random_number = 365 RAND_MAX = 32767

Με seed = 1000, το output θα είναι

random_number = 3304 RAND_MAX = 32767

Επομένως το **RAND_MAX**

1. είναι μία σταθερά
2. μπορεί να χρησιμοποιηθεί για την κανονικοποίηση των τυχαίων τιμών, ώστε να προκύπτουν στο διάστημα [0, 1), σύμφωνα με τη σχέση

$$x = \text{rand}() / (\text{RAND\_MAX}+1),\ 0 \le x < 1$$

Στοιχεία της γλώσσας C – Γεννήτρια τυχαίων αριθμών

Άσκηση

Εκτυπώστε στην οθόνη 100 τυχαίους αριθμούς

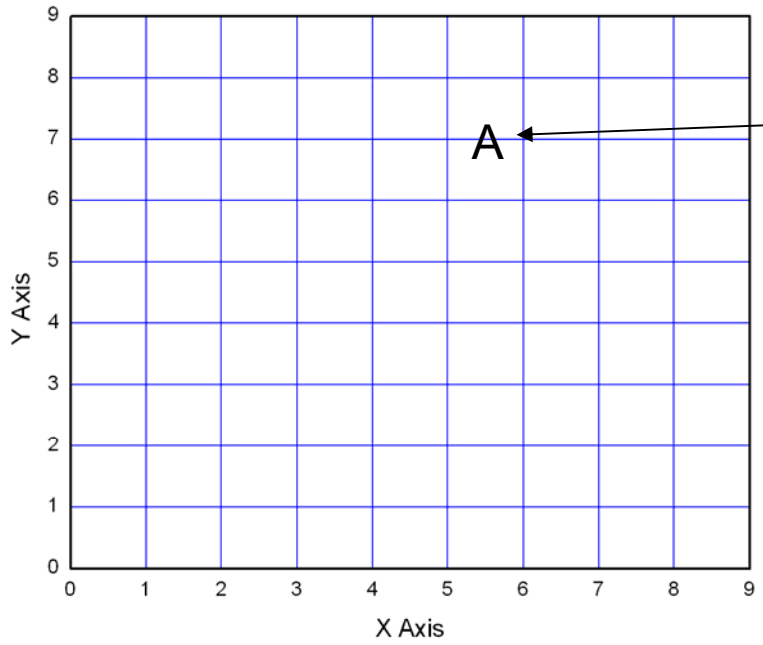
1. χρησιμοποιώντας το ίδιο seed για όλους τους αριθμούς
2. χρησιμοποιώντας διαφορετικό seed για κάθε αριθμό

Στο ίδιο πρόγραμμα πραγματοποιήστε τις απαραίτητες αλλαγές, ώστε οι 100 τυχαίοι αυτοί αριθμοί να προκύπτουν στο διάστημα [0, 1).

1. Αποθηκεύστε αυτούς τους αριθμούς σε ένα αρχείο
2. Υπολογίστε το μέσο όρο τους

Δημιουργία πλέγματος

Θέλουμε να προσομοιώσουμε μία τετραγωνική επιφάνεια (**2D square lattice**).



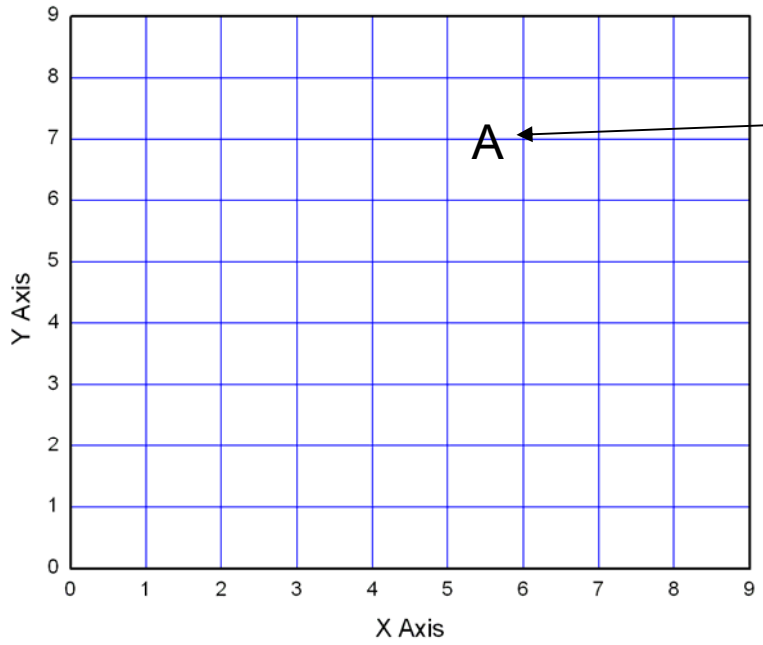
Σε κάθε σημείο Α του χώρου αντιστοιχούν δύο ακέραιες συντεταγμένες (x, y).

Κάθε σημείο του πλέγματος μπορεί να παρασταθεί ως ένα στοιχείο ενός διαδιάστατου πίνακα.

Πλέγμα = ο χώρος που θέλουμε να προσομοιώσουμε

Δημιουργία πλέγματος

Θέλουμε να προσομοιώσουμε μία τετραγωνική επιφάνεια (**2D square lattice**).



Σε κάθε σημείο του χώρου αντιστοιχούν δύο ακέραιες συντεταγμένες (x, y). Το A έχει συντεταγμένες (6, 7).

Το A(6, 7) μπορεί να παρασταθεί ως site[6][7].

Κάθε άλλο σημείο με συντεταγμένες (x, y) μπορεί να παρασταθεί ως site[x][y].

Δημιουργία πλέγματος

Θεωρούμε ότι στο σημείο Α με συντεταγμένες (x, y) τοποθετούμε ένα σωματίδιο, οπότε

- η θέση Α είναι κατηλλειμένη, οπότε ισούται με 1
- οποιαδήποτε άλλη θέση είναι κενή, οπότε ισούται με 0

Μπορούμε να δημιουργήσουμε ένα διδιάστατο πίνακα, π.χ. site[x][y]

- κάθε στοιχείο του οποίου αντιστοιχεί σε μία πλεγματική θέση
- κάθε στοιχείο του οποίου μας δείχνει αν η πλεγματική θέση είναι κενή (τιμή 0) ή όχι (τιμή 1)

Εάν θεωρήσουμε ότι το σωματίδιο βρίσκεται στη θέση Α(6, 7), θα είναι

- site[6][7] = 1
- site[x][y] = 0, για κάθε $x \neq 6$ και για κάθε $y \neq 7$.

Δημιουργία πλέγματος

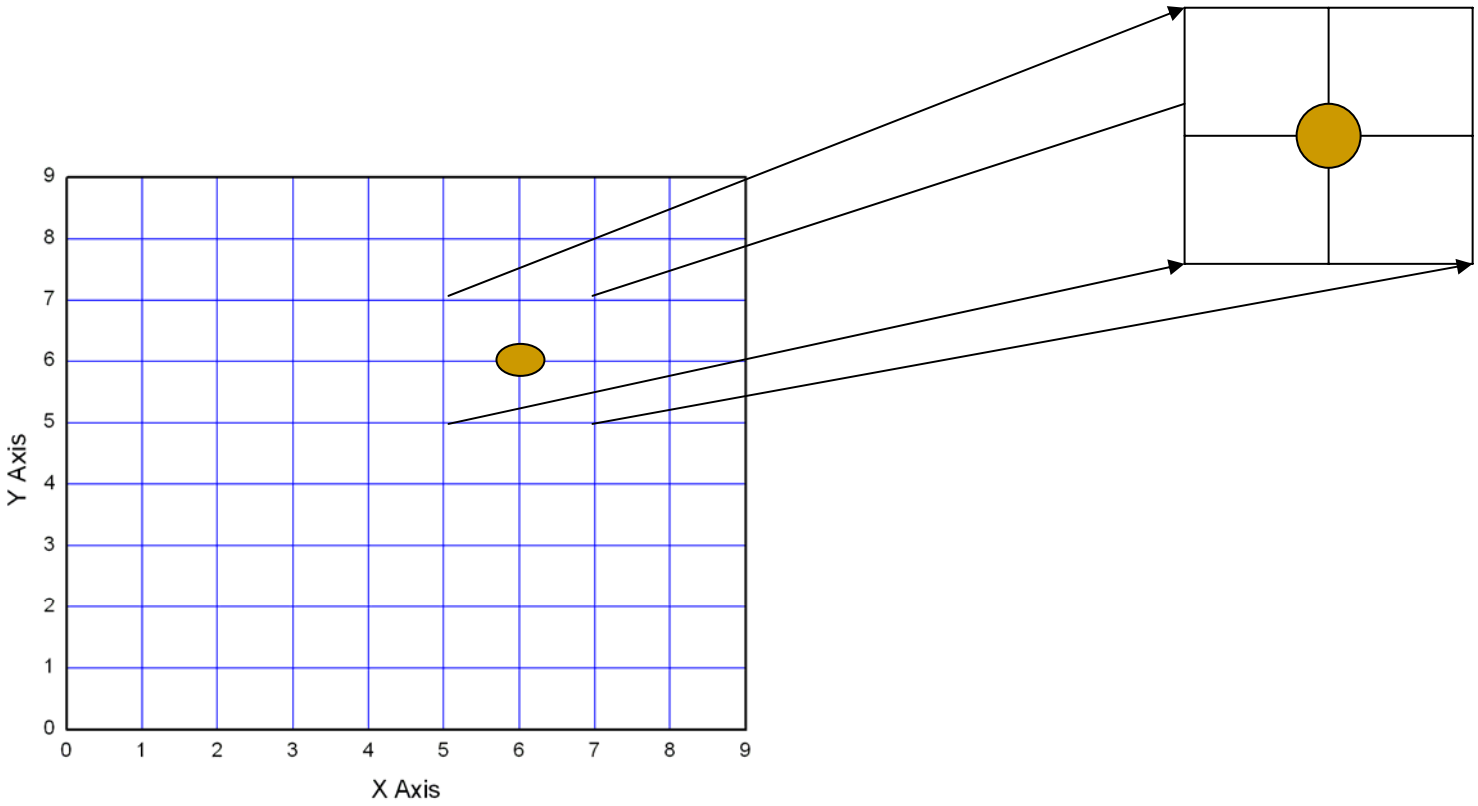
Εκχωρούμε την τιμή 0 σε όλα τα στοιχεία του πίνακα, οπότε θεωρούμε ότι καμία θέση του πλέγματος δεν είναι κατηλλειμένη

```
for(x=0;x<10;x++){
        for(y=0;y<10;y++){
                site[x][y]=0;
        }
}
```

Εκχωρούμε την τιμή 1 στο στοιχείο του πίνακα που αντιστοιχεί στην κατηλλειμένη θέση

```
site[6][7]=1;
```

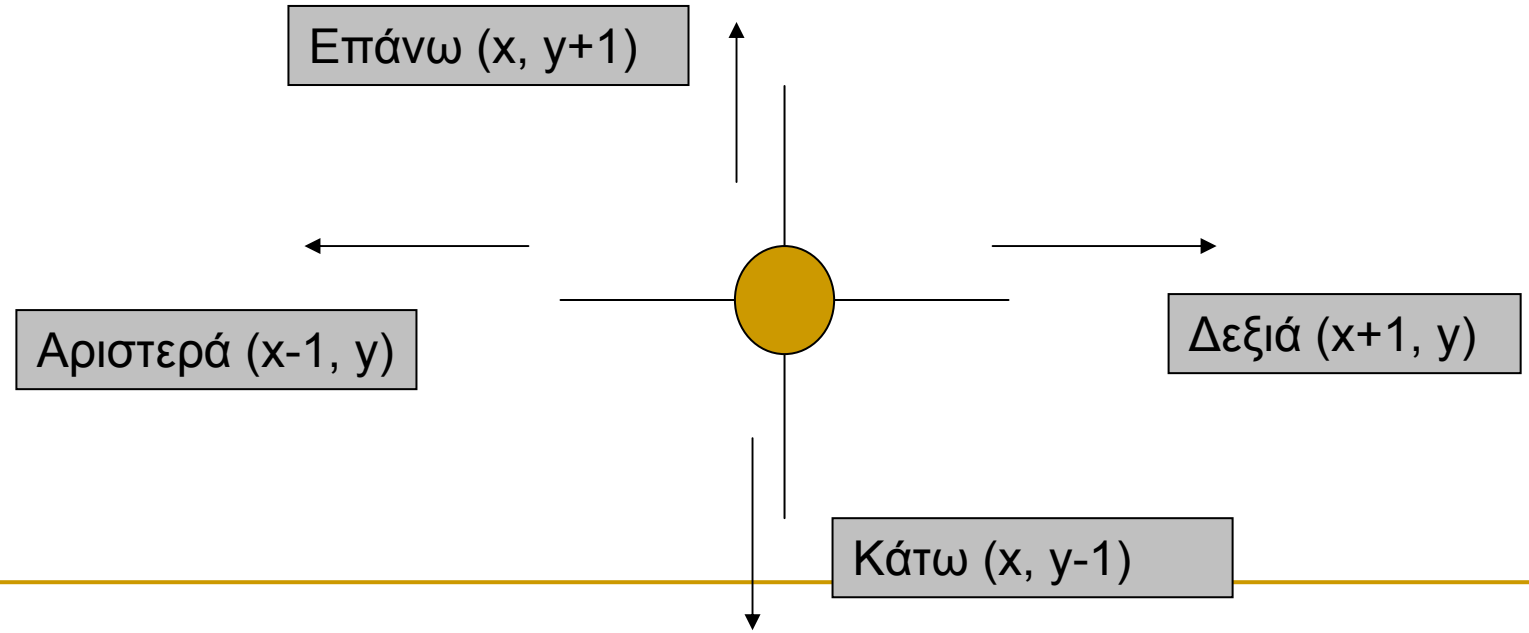
Δημιουργία πλέγματος – Τυχαία κίνηση σωματιδίου



Δημιουργία πλέγματος – Τυχαία κίνηση σωματιδίου

Ένα σωματίδιο που εκτελεί τυχαίο βηματισμό (random walk) σε ένα πλέγμα δύο διαστάσεων

- περιβάλλεται από 4 θέσεις (αριστερά, δεξιά, πάνω, κάτω)
- μπορεί να κινηθεί με την ίδια πιθανότητα προς οποιαδήποτε από αυτες τις θέσεις
- η πιθανότητα να κινηθεί προς κάθε μία από αυτές τις θέσεις είναι ίση προς 0.25



Δημιουργία πλέγματος – Τυχαία κίνηση σωματιδίου

Αλγόριθμος

- Επιλέγουμε ένα τυχαίο αριθμό (random_number)
- Εάν είναι 0.00 ≤random_number < 0.25 κινείται προς τα δεξιά
- Εάν είναι 0.25 ≤ random_number < 0.50 κινείται προς τα αριστερά
- Εάν είναι 0.50 ≤ random_number < 0.75 κινείται προς τα πάνω
- Εάν είναι 0.75 ≤ random_number < 1.00 κινείται προς τα κάτω

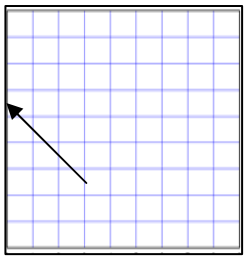
Κώδικας

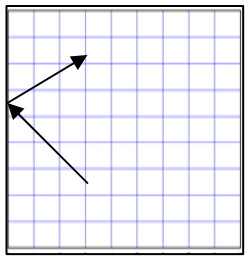
- Επιλέγουμε ένα τυχαίο αριθμό (random_number)
- if(0.00 ≤random_number < 0.25) x=x+1;
- else if(0.25 ≤ random_number < 0.50) x=x-1;
- else if(0.50 ≤ random_number < 0.75) y=y+1;
- else y=y-1;

Δημιουργία πλέγματος – Οριακές συνθήκες

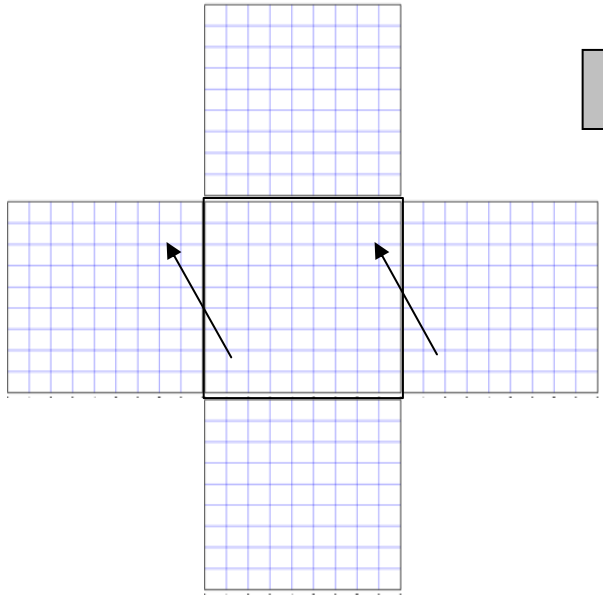
Όταν ένα σωματίδιο φτάνει στα όρια του πλέγματος

- είτε διακόπτει την κίνησή του
- είτε ανακλάται και επιστρέφει στο πλέγμα (reflective boundary conditions)
- είτε κινείται περιοδικά και επιστρέφει στο πλέγμα από κάποιο άλλο σημειο (periodic boundary conditions)





Reflective



Periodic

Δημιουργία πλέγματος – Περιοδικές οριακές συνθήκες

Σύμφωνα με τις περιοδικές οριακές συνθήκες

• όταν ένα σωματίδιο εξέρχεται από το πλέγμα, επιστρέφει σε αυτό από κάποιο σημείο που βρίσκεται στην απέναντι πλεύρά από αυτή που εξήλθε

Αλγόριθμος

if(το σωματίδιο βρίσκεται στα όρια του πλέγματος)

μετέφερέ το στην αντίστοιχη θέση στην απέναντι πλευρά

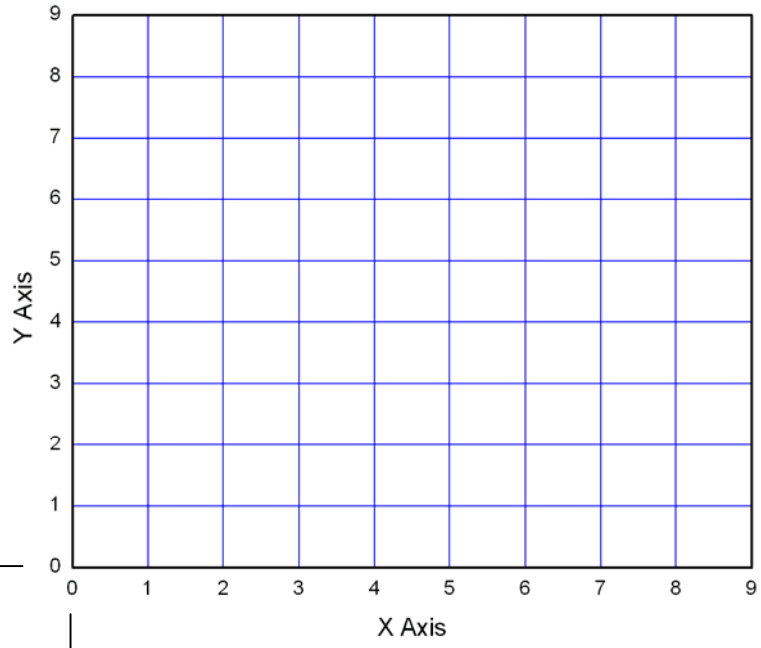
Κώδικας

```
if(x<0)              x=x+N;
else if(x>=N)        x=x-N;
else if(y<0)         y=y+N;
else if(y>=N)        y=y-N;
```

Δημιουργία πλέγματος – Περιοδικές οριακές συνθήκες

Κώδικας

```
if(x<0)              x=x+N;
else if(x>=N)        x=x-N;
else if(y<0)         y=y+N;
else if(y>=N)        y=y-N;
```

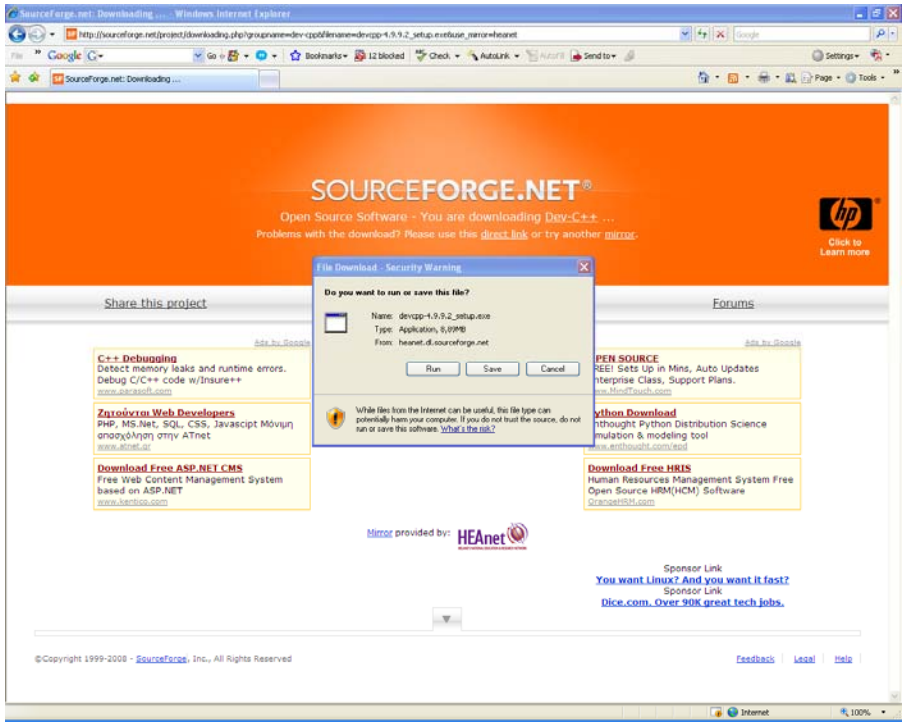


Οδηγίες σχετικά με την εγκατάσταση και τη λειτουργία της DevC++

Compiler της C – DevC++

Μπορείτε να βρείτε τον compiler της DevC++ στο παρακάτω link

http://sourceforge.net/project/downloading.php?groupname=dev-cpp&filename=devcpp-4.9.9.2_setup.exe&use_mirror=heanet



Θα χρειαστεί να κάνετε save το πρόγραμμα σε κάποιον φάκελο στον Η/Υ σας και στη συνέχεια run για να το εγκαταστήσετε

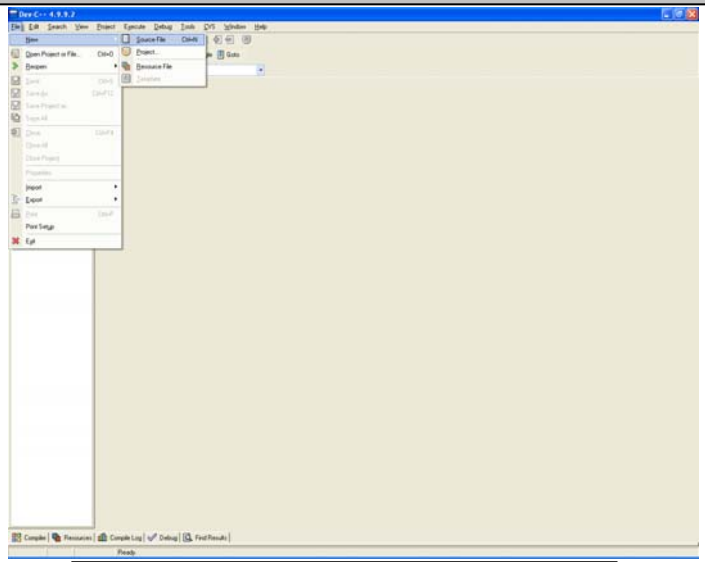
Compiler της C – DevC++

Για να δημιουργήσετε ένα αρχείο της C αρκεί να επιλέξετε

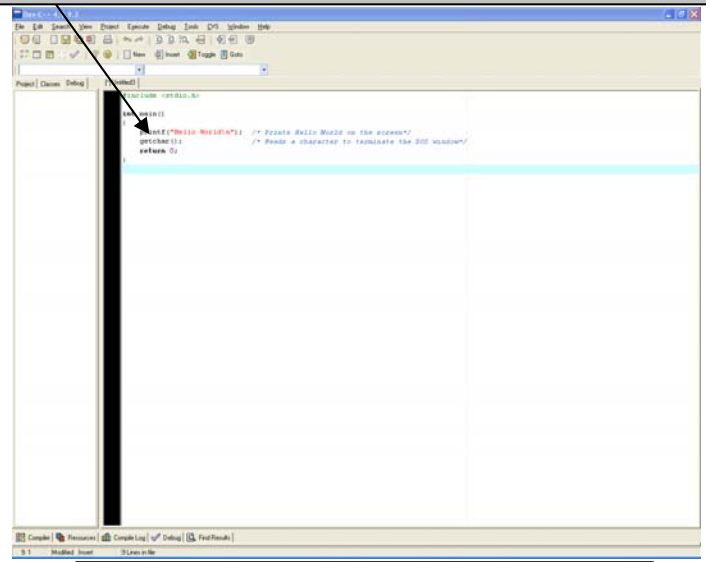
File ⟶ New ⟶ Source File

Με τον τρόπο αυτό δημιουργείτε ένα νέο αρχείο, ανοίγει ο κειμενογράφος και μπορείτε πλέον να γράψετε τον source code που επιθυμείτε.

Όταν κατά την εκτέλεσή του ένα πρόγραμμα συναντήσει την εντολή getchar(), τότε αναμένει τον χρήστη να πατήσει ένα οποιοδήποτε πλήκτρο, πριν τερματιστεί η εφαρμογή.



Άνοιγμα νέου αρχείου



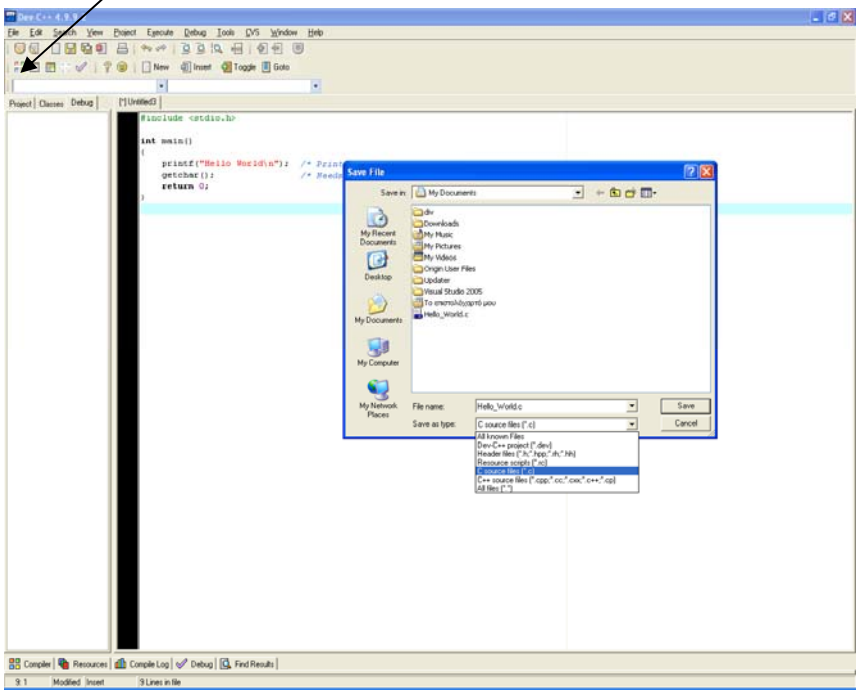
Πρόγραμμα Hello World

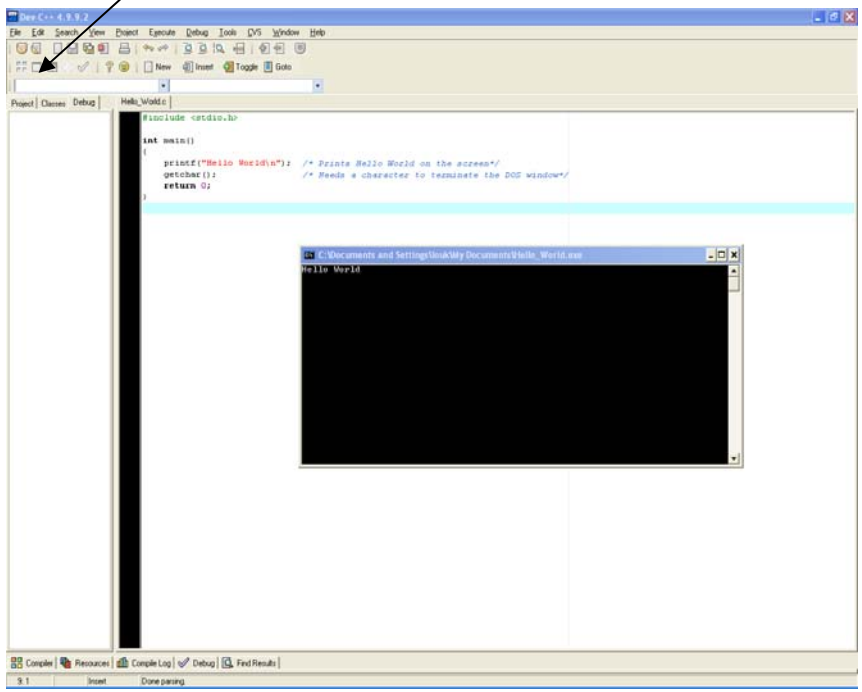
Compiler της C – DevC++

Αφού έχετε ετοιμάσει το source code, αρκεί να κάνετε compile, αποθηκεύοντας το αρχείο εκεί που επιθυμείτε, ως C source files και στη συνέχεια μπορείτε να το εκτελέσετε.

Πλήκτρο για compile

Πλήκτρο για run





Compile και save as

Εκτέλεση του προγράμματος

Θεωρητική Φυσική Στερεάς Κατάστασης

Άσκηση 5

Ενδεικτική Επίλυση

Άσκηση 5 – Many Particles Trapping

Ορίζουμε τον πίνακα που παριστάνει το πλέγμα

int plegma [N][N];

Ορίζουμε τον πίνακα που δηλώνει την κατάσταση ενός σωματιδίου (0 για παγιδευμένο, 1 για ελεύθερο)

int state_of_part[N_part];

Ορίζουμε τον πίνακα του αριθμού των μη παγιδευμένων σωματιδίων για κάθε χρονική στιγμή

int N_free_particles [N_mcs];

Δήλωση πινάκων (-1) θέσεων των σωματιδίων

int pos_x [N_part];
int pos_y [N_part];

Τοποθετούμε την παγίδα στο κέντρο

plegma[(int)(N/2)][(int)(N/2)]=-1;

N_part : ολικός αριθμός σωματιδίων (C*N*N)

N_mcs : αριθμός των Monte-Carlo steps (μονάδα χρόνου)

Άσκηση 5 – Many Particles Trapping

Τοποθετούμε τα N_part σωματίδια σε τυχαίες θέσεις στο πλέγμα. Με την ακόλουθη διαδικασία αποφεύγουμε να τοποθετήσουμε δύο σωματίδια στην ίδια θέση.

```
for (part=0;part<N_part;part++)
{
        do
       { // τραβάμε 2 ακέραιους τυχαίους αριθμούς μεταξύ 0 και N-1 (rand1, rand2)
          x=rand1;
          y=rand2;
        }while (plegma[x][y]!=0)

        plegma[x][y]=1;
        pos_x[part]=x;
        pos_y[part]=y;
        state_of_part[part]=1;
}
```

Άσκηση 5 – Many Particles Trapping

Ένα **Monte-Carlo step** ορίζεται ως προσπάθεια κίνησης όλων των μη παγιδευμένων σωματιδίων. Αποτελεί τη μονάδα χρόνου (time step) του προβλήματος.

```
for (run=0;run<N_runs;run++)    // Επαναληπτική διαδικασία για τα run
{
    for (mcs=0; mcs<N_mcs;mcs++) // Επαναληπτική διαδικασία για τα time steps
    {
         for (part=0;part<N_part;part++) // Επαναληπτική διαδικασία για κάθε particle
        {
            if(state_of_part[part]==1)  // Δεν έχει παγιδευτεί το particle
            Εκτελούμε το random walk.
            Ελέγχουμε εάν το σωματίδιο παγιδεύεται στη νέα θέση.
        }
   }
}
```

Άσκηση 5 – Many Particles Trapping

Τοποθετούμε τα N_part σωματίδια σε τυχαίες θέσεις στο πλέγμα. Με την ακόλουθη διαδικασία αποφεύγουμε να τοποθετήσουμε δύο σωματίδια στην ίδια θέση.

```
for (run=0;run<N_runs;run++)    // Επαναληπτική διαδικασία για τα run
{
    for (mcs=0; mcs<N_mcs;mcs++) // Επαναληπτική διαδικασία για τα time steps
    {
         for (part=0;part<N_part;part++) // Επαναληπτική διαδικασία για κάθε particle
        {
            if(state_of_part[part]==1)  // Δεν έχει παγιδευτεί το particle
            Εκτελούμε το random walk.
            Οριακές συνθήκες
            Ελέγχουμε εάν το σωματίδιο παγιδεύεται στη νέα θέση.
        }
   }
}
```

Άσκηση 5 – Random Walk

Random Walk

Στο πρόβλημα υπάρχουν πολλά σωματίδια. Επομένως υπάρχει η περίπτωση ένα σωματίδιο να ¨προσπαθεί¨ να κινηθεί σε μία ήδη κατειλημμένη θέση, κάτι το οποίο δεν μπορεί να συμβεί, οπότε θα παραμείνει στην ίδια. Η ανεπιτυχής αυτή προσπάθεια θεωρείται ως τμήμα του Monte-Carlo step.

• Το σωματίδιο A βρίσκεται στην θέση (x,y)

• Με βάση ένα τυχαίο αριθμό επιλέγουμε την κατεύθυνση προς την οποία θα προσπαθήσει να κινηθεί το σωματίδιο A (διαφάνεια 59), π.χ. προς x+1

• Εφαρμόζουμε τις οριακές συνθήκες για τη νέα πιθανή θέση x+1 του A

• Αν στην θέση x+1 υπάρχει ήδη κάποιο σωματίδιο, τότε το A θα παραμείνει στην αρχική του θέση x και όλοι οι πίνακες παραμένουν ως έχουν.

• Αν δεν υπάρχει κάποιο άλλο σωματίδιο στη νέα θέση, τότε το A θα κινηθεί. Στην περίπτωση αυτή φροντίζουμε να κάνουμε update τους πίνακες pos[x], pos[y] για το A.

Άσκηση 5 – Trap

Εάν το σωματίδιο Α τελικά μετακινηθεί, ελέγχουμε το κατά πόσο η νέα του θέση αντιστοιχεί σε παγίδα.

• Αν πρόκειται για παγίδα

-- το σωματίδιο διαγράφεται από τον πίνακα plegma.

-- στον πίνακα state_of_part το θεωρούμε πλέον ίσο προς μηδέν.

• Αν δεν πρόκειται για παγίδα το μετακινούμε στη νέα θέση και φροντίζουμε να ανανεώσουμε τον πίνακα plegma.