2ο επαναληπτικό εργαστήριο

Εντολές επανάληψης

- Μερικές φορές πρέπει να επαναλάβουμε την εκτέλεση κάποιας (σύνθετης) εντολής
  - είτε για ένα συγκεκριμένο αριθμό επαναλήψεων
  - είτε για όσο ισχύει μια συνθήκη.
- Στη C αυτό μπορεί να γίνει με χρήση μίας εκ των εντολών:
  - for
  - while
  - do…while
- Ο έλεγχος της επανάληψης γίνεται με τη βοήθεια λογικών εκφράσεων/προτάσεων.
- Οι δομές που σχηματίζουν οι εντολές επανάληψης ονομάζονται «βρόχοι».

Εντολή while-Συντακτικό

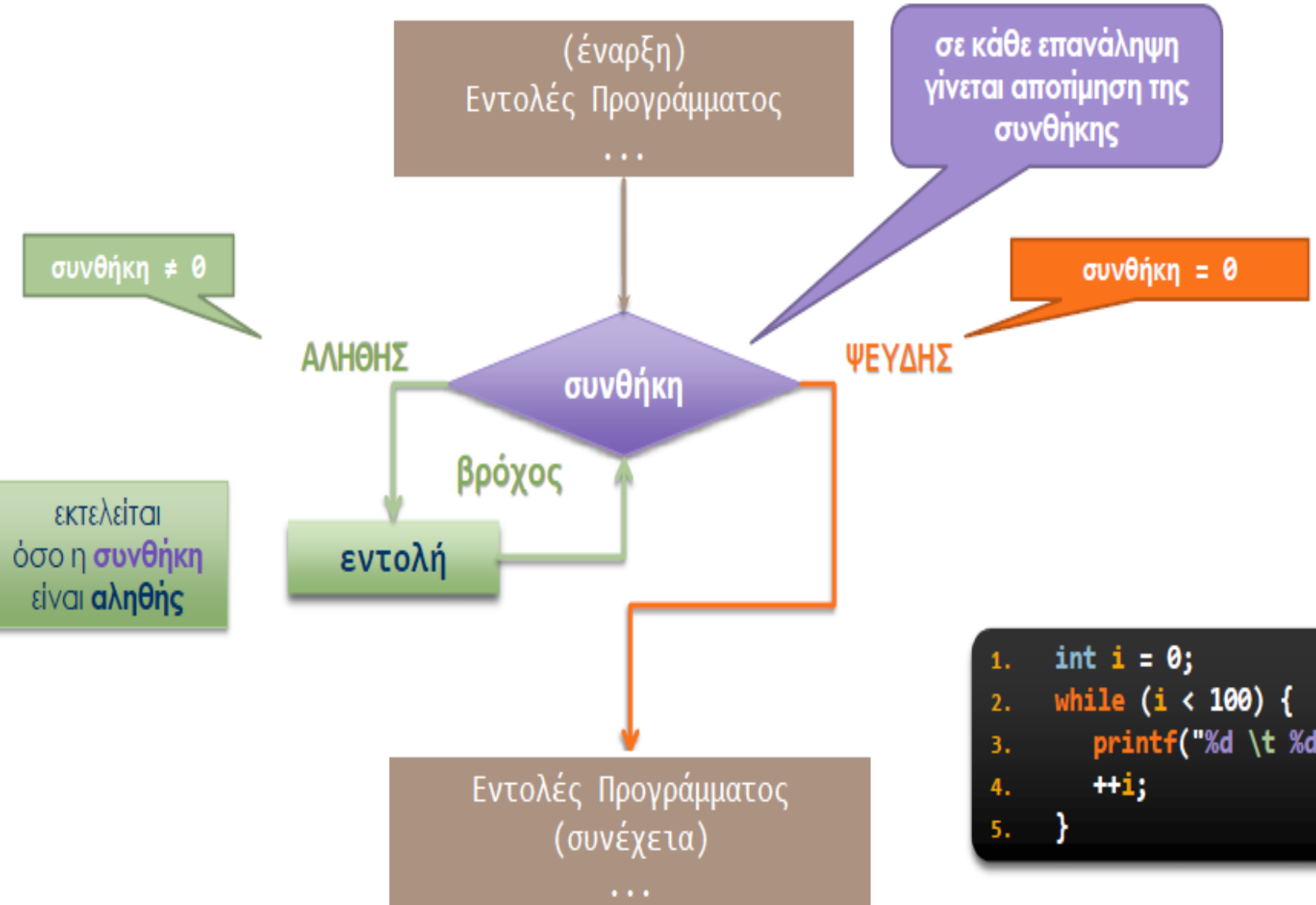
```
1.      int i = 0;
2.      while (i < 100) {
3.         printf("%d \t %d\n", i, i*i);
4.         ++i;
5.      }
```

- συντακτικό εντολής **while**:

```
while (condition)
      statement        // while the condition is true, do statement
```

- η εντολή (statement) μπορεί να είναι απλή (μία) ή σύνθετη (μπλοκ)
- η εντολή εκτελείται όσο ισχύει η συνθήκη (condition)
  - η συνθήκη αποτιμάται σε κάποια τιμή a και είναι
    - **αληθής** όταν a ≠ 0
    - **ψευδής** όταν a = 0
- όταν η συνθήκη γίνει ψευδής → σταματά η εκτέλεση της εντολής

Επανάληψη-while -Διάγραμμα ροής




```
1.  int i = 0;
2.  while (i < 100) {
3.      printf("%d \t %d\n", i, i*i);
4.      ++i;
5.  }
```

Επανάληψη-while - Παρατηρήσεις

- η συνθήκη μπορεί να είναι
  1. μια έκφραση συσχετισμού `x>10`
  2. μια λογική πρόταση `(x>10 && x<20)`
  3. το αποτέλεσμα κάποιας πράξης `x+y`
  4. μεταβλητή ή τιμή `x` ή `100`
- αν η εντολή δεν είναι σύνθετη → τα άγκιστρα **{ }** δεν είναι απαραίτητα
- αν η συνθήκη είναι **ψευδής** από την αρχή → δε θα εκτελεστεί ποτέ η εντολή

```
while (2 > 3) statement
```

- αν η συνθήκη είναι πάντα **αληθής** → ατέρμων βρόχος

```
while (1) statement
```

Επανάληψη-while - παραδείγματα

```
1.  #include <stdio.h>
2.
3.  main ()
4.  {
5.    int i = 1;
6.    while (i < 5) {
7.      printf("%d\n", i);
8.      ++i;
9.    }
10.   printf("end\n");
11. }
```

i	(i<5)	printf
1	1	1
2	1	2
3	1	3
4	1	4
5	0	end

```
1.  #include <stdio.h>
2.
3.  main ()
4.  {
5.    int i = 1;
6.    while (++i < 5) {
7.      printf("%d\n", i);
8.    }
9.    printf("end\n");
10. }
```

i	(++i<5)	printf
1	(2<5) → 1	2
2	(3<5) → 1	3
3	(4<5) → 1	4
4	(5<5) → 0	end

Επανάληψη-while - παραδείγματα

- γράψτε ένα πρόγραμμα το οποίο να ζητά από το χρήστη έναν ακέραιο n και να εμφανίζει όλους τους ακεραίους που είναι μικρότεροι από το n

```c
// εμφάνιση αριθμών μικρότερων του n

#include <stdio.h>

main ()
{
   int i, n;
   printf("Enter integer: ");
   scanf("%d", &n);
   i = 0;
   while (i < n) {
      printf("i is now: %d\n", i);
      ++i;
   }
}
```

- γράψτε ένα πρόγραμμα το οποίο να ζητά από το χρήστη έναν ακέραιο n και να εμφανίζει όλους τους ακεραίους από το **0** έως και το n

```c
// εμφάνιση αριθμών από το 0 έως και το n

#include <stdio.h>

main ()
{
   int i, n;
   printf("Enter integer: ");
   scanf("%d", &n);
   i = 0;
   while (i <= n) {
      printf("i is now: %d\n", i);
      ++i;
   }
}
```

Επανάληψη-while - παραδείγματα

- γράψτε ένα πρόγραμμα το οποίο να ζητά από το χρήστη έναν ακέραιο n και υπολογίζει το άθροισμα: **1 + 2 + ... + n**

```
1.    int i, n, sum;
2.
3.    printf("Enter integer: ");
4.    scanf("%d", &n);
5.
6.    i = 0;
7.    sum = 0;
8.    while (i <= n) {
9.      sum += i;
10.     ++i;
11.   }
12.
13.   printf("Sum = %d\n", sum);
```

Επανάληψη - do..while - Συντακτικό

- συντακτικό εντολής **do…while**:

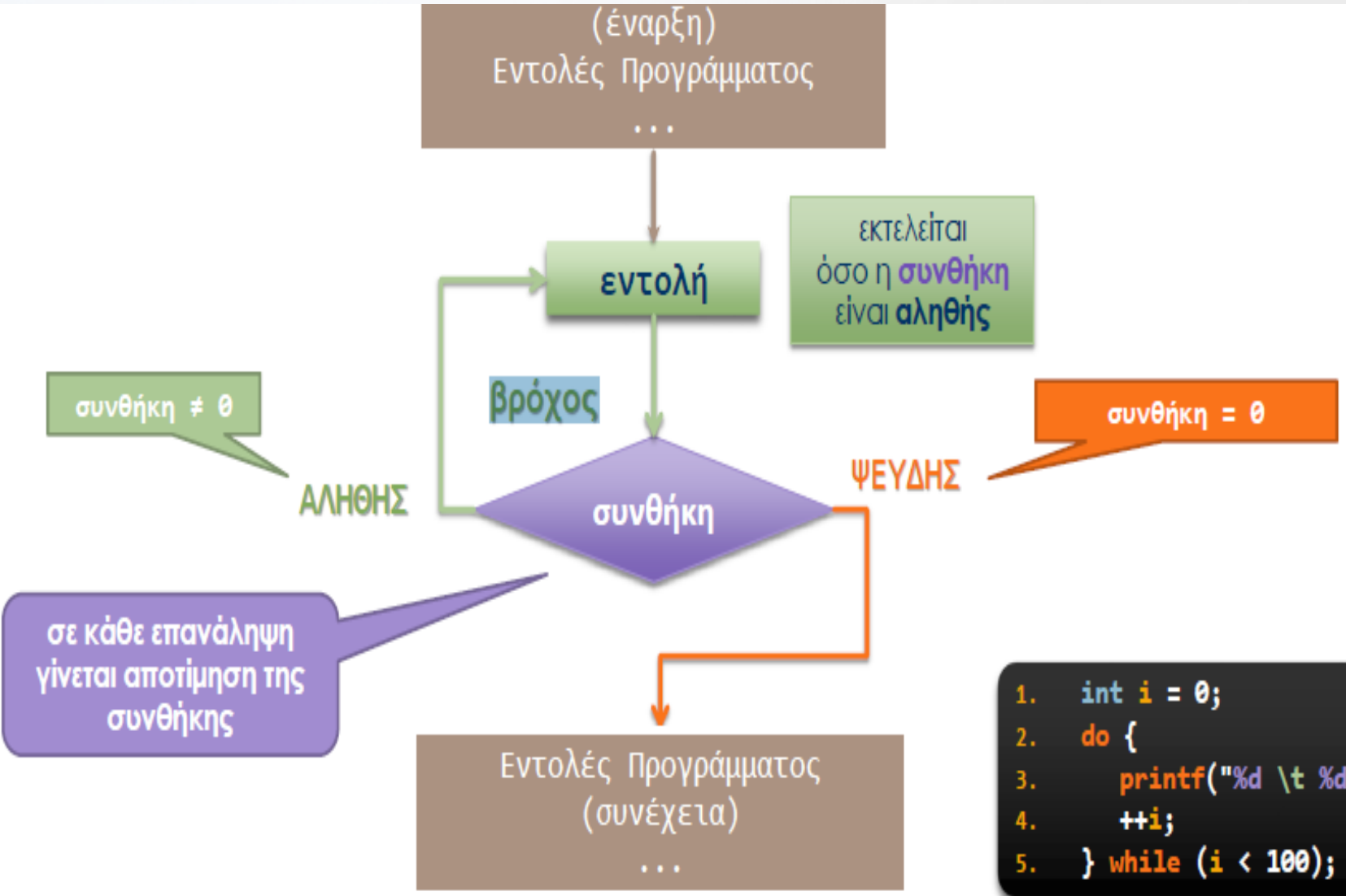
```
do
    statement          // do statement, while condition is true
while (condition);
```

- ο έλεγχος της συνθήκης (condition) γίνεται στο τέλος του βρόχου
- η εντολή (statement) εκτελείται τουλάχιστον μία φορά
- το **while** στο τέλος της εντολής **do…while** πρέπει να τελειώνει με ελληνικό ερωτηματικό (;)

```
1.    int i = 0;
2.    while (i < 100) {
3.       printf("%d \t %d\n", i, i*i);
4.       ++i;
5.    }
```

```
1.    int i = 0;
2.    do {
3.       printf("%d \t %d\n", i, i*i);
4.       ++i;
5.    } while (i < 100);
```

Επανάληψη- do..while - Διάγραμμα ροής




```
1.  int i = 0;
2.  do {
3.     printf("%d \t %d\n", i, i*i);
4.     ++i;
5.  } while (i < 100);
```

Παράδειγμα

- Ας υποθέσουμε ότι γίνεται ένας έρανος στον οποίο προσπαθούμε να μαζέψουμε το ποσό των 1000€. Ζητάμε από διάφορους πολίτες να συνεισφέρουν με κάποιο ποσό και σταματάμε όταν το συνολικό ποσό γίνει ίσο ή υπερβεί τα 1000€. Το πρόγραμμα που ακολουθεί υλοποιεί αυτή την απαίτηση.

```c
#include <stdio.h>
#include <stdlib.h>
main()
{
float poso,sum=0;
int plithos=0,polla=0,liga=0;
do
{
printf("Dwse poso: ");
scanf("%f",&poso);
sum=sum+poso;
plithos++;
if (poso>=50)
polla++;
if (poso<=2)
liga++;
} while (sum<1000);
printf("Synoliko poso: %f\n",sum);
printf("Atoma pou symemteixan ston erano: %d\n",plithos);
printf("Atoma pou edwsasn apo 50 euro kai panw: %d\n",polla);
printf("Atoma pou edwsan mexri 2 euro: %d\n",liga);
return 0;
}
```

Το ποσό που δίνει ο χρήστης προστίθεται κάθε φορά στη μεταβλητή sum.

Η μεταβλητή plithos μετράει το σύνολο των ατόμων που συμμετείχαν στον έρανο. Η polla μετρά το σύνολο των ατόμων που έδωσαν από 50 € και πάνω και η liga μετρά τα άτομα που έδωσαν μέχρι 2 €.

Η επαναληπτική διαδικασία συνεχίζεται όσο η μεταβλητή sum έχει τιμή μικρότερη του 1000.

Επανάληψη - for

- η εντολή **for** είναι παρόμοια με την εντολή **while**, με τη διαφορά ότι η διαχείριση της μεταβλητής ελέγχου επικεντρώνεται στην αρχή → ευκολότερο να καταλάβουμε τι συμβαίνει

```
1.  for (int i = 0; i < 100; ++i)
2.      printf("%d \t %d\n", i, i*i);
```

```
1.  int i = 0;
2.  while (i < 100) {
3.      printf("%d \t %d\n", i, i*i);
4.      ++i;
5.  }
```

- συντακτικό εντολής **for**:

```
for (initialize; condition; increment)
    statement
```

Εντολή **for**: επαναλαμβάνει μια ενότητα (μπλοκ) εντολών
για ένα γνωστό εκ των προτέρων αριθμό φορών

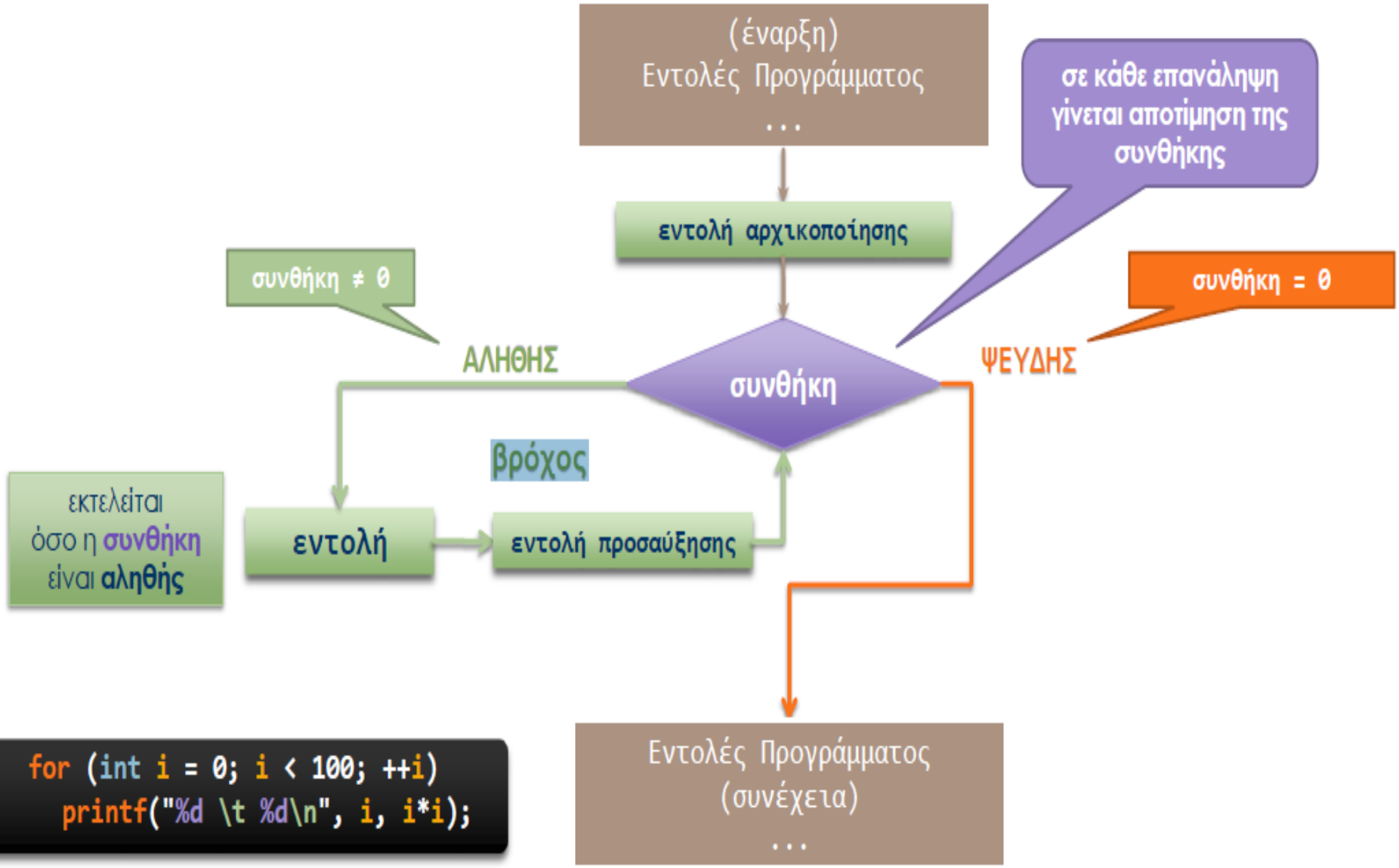
```
for (Εντολή1; συνθήκη; Εντολή2 ) {
    ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ
}
```

Η Εντολή1 μπορεί να είναι και περισσότερες από μια εντολές που συνήθως καθορίζουν κάποιες αρχικές τιμές. Εκτελείται μία και μόνο φορά, στην αρχή.

Η Εντολή2 είναι μια παράσταση που συνήθως αλλάζει τη τιμή μιας μεταβλητής που βρίσκεται στη <συνθήκη >. Μπορεί να είναι και πάνω από μια εντολή.

Η ανάπτυξη της εντολής for είναι η εξής:
Εκτέλεση της Εντολής1 και στη συνέχεια εκτέλεση της ΕΝΟΤΗΤΑΣ_ΕΝΤΟΛΩΝ και της Εντολής2, όσο η <συνθήκη > είναι αληθής

Επανάληψη - for - Διάγραμμα ροής




```
1.  for (int i = 0; i < 100; ++i)
2.    printf("%d \t %d\n", i, i*i);
```

Επανάληψη -for -Παρατηρήσεις

- συντακτικό εντολής **for**:

```
for (initialize; condition; increment)
    statement
```

  - αν η συνθήκη είναι **ψευδής** δε θα εκτελεστεί ποτέ η εντολή
  - η εντολή αρχικοποίησης (**initialize**) μπορεί να είναι περισσότερες από μία εντολές

```
for (i = 0, j = 0, k = 1; i < k; ++i) {…}
```

  - η εντολή προσαύξησης (**increment**) μπορεί να είναι περισσότερες από μία εντολές

```
for (i = 0; i < k; ++i, j+=2) {…}
```

  - όταν οι εντολές αρχικοποίησης και προσαύξησης αποτελούνται από πολλαπλές εντολές τότε αυτές χωρίζονται μεταξύ τους με κόμμα (,)

- χρησιμοποιούμε τη **for** όταν γνωρίζουμε εκ των προτέρων τον αριθμό επαναλήψεων

- δε βάζουμε το ελληνικό ερωτηματικό (;) στο τέλος της **for**

```
for (i = 0; i < 100; ++i); {…} // απλά αυξάνει το i 100 φορές
```

- δεν είναι απαραίτητο να υπάρχουν και τα τρία τμήματα της **for**
  - όμως υπάρχει πάντα το διαχωριστικό ';' μεταξύ των τμημάτων

```
for (;;) {…}      // ατέρμων βρόχος
```

Επανάληψη-for -1ο παράδειγμα

- γράψτε ένα πρόγραμμα το οποίο να ζητά από το χρήστη έναν ακέραιο n και να εμφανίζει όλους τους ακεραίους που είναι μικρότεροι από το n

```
1.  // εμφάνιση αριθμών μικρότερων του n
2.
3.  #include <stdio.h>
4.
5.  main ()
6.  {
7.     int i, n;
8.     printf("Enter integer: ");
9.     scanf("%d", &n);
10.    for (i = 0; i < n; ++i)
11.       printf("i is now: %d\n", i);
12. }
```

- γράψτε ένα πρόγραμμα το οποίο να ζητά από το χρήστη έναν ακέραιο n και να εμφανίζει όλους τους ακεραίους από το **0** έως και το n

```
1.  // εμφάνιση αριθμών από το 0 έως και το n
2.
3.  #include <stdio.h>
4.
5.  main ()
6.  {
7.     int i, n;
8.     printf("Enter integer: ");
9.     scanf("%d", &n);
10.    for (i = 0; i <= n; ++i)
11.       printf("i is now: %d\n", i);
12. }
```

Καταμέτρηση και άθροιση

- Αρκετά συχνά, στα προγράμματά μας πρέπει να μετράμε ένα πλήθος (για παράδειγμα, το πλήθος των αριθμών που είναι μεγαλύτεροι από 10), αλλά και να αθροίζουμε τις διαφορετικές τιμές που λαμβάνει μια μεταβλητή.
- Για την καταμέτρηση ενός πλήθους χρησιμοποιούμε πάντα μια μεταβλητή στην οποία δίνουμε αρχική τιμή 0, και κάθε φορά που ικανοποιείται μια συνθήκη αυξάνουμε τη μεταβλητή κατά 1.
- Για το συνολικό άθροισμα διαφορετικών τιμών χρησιμοποιούμε πάλι μια μεταβλητή, στην οποία δίνουμε επίσης αρχική τιμή 0, και κάθε φορά που δημιουργείται μια νέα τιμή αυξάνουμε τη μεταβλητή κατά την τιμή αυτή.

Υπολογισμός μέγιστου και ελάχιστου

- Το παρακάτω πρόγραμμα διαβάζει δέκα ακέραιους αριθμούς που δίνονται από το πληκτρολόγιο και εμφανίζει τον μέγιστο και τον ελάχιστο από αυτούς.
- Ο μέγιστος αριθμός αποθηκεύεται στη θέση max και ο ελάχιστος στη θέση min.
- Η τεχνική είναι η εξής: Διαβάζουμε τον πρώτο αριθμό και τον θεωρούμε ταυτόχρονα τον μέγιστο αλλά και τον ελάχιστο που έχουμε διαβάσει μέχρι στιγμής, καταχωρίζοντάς τον στις μεταβλητές max και min.
- Κατόπιν, με μια επαναληπτική διαδικασία, διαβάζουμε έναν προς έναν τους υπόλοιπους αριθμούς και τους συγκρίνουμε με τα περιεχόμενα των μεταβλητών max και min. Αν ο αριθμός είναι μεγαλύτερος από τη θέση max τον καταχωρίζουμε στη max, ενώ αν είναι μικρότερος από τη min τον καταχωρίζουμε στη min.
- Τελικά οι μεταβλητές max και min θα περιέχουν τον μεγαλύτερο και τον μικρότερο αριθμό αντίστοιχα.

```c
#include <stdio.h>
main()
{
   int i,ar,min,max;
   for (i=1;i<=10;i++)
   {
      printf("Δώσε αριθμό:");
      scanf("%d",&ar);
      if (i==1)
         max=min=ar;
      if (ar>max)
         max=ar;
      else if (ar<min)
         min=ar;
   }

   printf("max=%d\nmin=%d\n",max, min);
   return 0;
}
```

Διαβάζει έναν προς έναν τους 10 αριθμούς

Αν ο αριθμός είναι ο πρώτος που δόθηκε (i==1), καταχωρίζεται ως αρχική τιμή τόσο στη μεταβλητή **max** όσο και στη μεταβλητή **min**.

Στην περίπτωση που ο αριθμός **ar** έχει τιμή μεγαλύτερη από την τρέχουσα μέγιστη τιμή (**max**), καταχωρίζεται στη θέση **max**, ενώ αν έχει τιμή μικρότερη από την τρέχουσα ελάχιστη τιμή (**min**), καταχωρίζεται στη θέση **min**.

Εμφανίζει τον μεγαλύτερο και μικρότερο αριθμό από αυτούς που δόθηκαν.

Ένθετοι βρόχοι

- Ένας βρόχος μπορεί να περιέχει άλλες εντολές βρόχων.
- Βρόχοι μέσα σε βρόχο -> ένθετοι βρόχοι.
- Σε αυτές τις περιπτώσεις θα πρέπει να είμαστε προσεκτικοί στους μετρητές που χρησιμοποιούνται στους βρόχους.

Προπαίδεια 1-10 (και 1-5)

```
#include <stdio.h>
#include <stdlib.h>
main()
{
int i, j;
for(i= 1; i<= 10; ++i)
{
for(j= 1; j<= 10; ++j)
printf("%d\t", i*j);
printf("\n");
}
}
```

```
1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100
```

```
for(i= 1; i<= 10; ++i){
for(j= 1; j<= 5; ++j)
printf("%d\t", i*j);
```

```
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
6	12	18	24	30
7	14	21	28	35
8	16	24	32	40
9	18	27	36	45
10	20	30	40	50
```

TETRAGONA _2.cpp.

Γράψτε ένα πρόγραμμα το οποίο θα ζητάει από το χρήστη Ν ακεραίους και αφού υπολογίσει το άθροισμα των τετραγώνων τους θα το εμφανίζει στην οθόνη. Ο ακέραιος Ν θα δίνεται επίσης από το χρήστη στην αρχή του προγράμματος.

Υπόδειξη:

Το πρόγραμμα διαφέρει από το προηγούμενο στο ότι σε κάθε επανάληψη θα πρέπει επιπλέον να διαβάζετε έναν ακέραιο από το πληκτρολόγιο, το τετράγωνο του οποίου θα προστίθεται στο άθροισμα.

Παράδειγμα εκτέλεσης προγράμματος – με ***κόκκινο*** *τα δεδομένα από το πληκτρολόγιο*

```
DOSE TO PLHTHOS AKERAION: 3

DOSE TON AKERAIO 1: 3
DOSE TON AKERAIO 2: 2
DOSE TON AKERAIO 3: 7

TO ATHROISMA TON TETRAGONON TON 3 AKERAION POU EDOSES EINAI 62
```

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

main()
{
    int i,N,K, athroisma;

    N=0;
    while (N<=0){
        printf("DOSE TO PLHTHOS AKERAION: ");
        scanf("%d", &N);
        if (N<=0) printf("LATHOS! TO PLHTHOS PREPEI NA EINAI THETIKO \n\n");
    }

    athroisma=0;
    for (i = 1; i<=N;i++)
    {
        printf("DOSE TON AKERAIO %d : ",i);
        scanf("%d", &K);
        athroisma = athroisma+K*K;
    }
    printf("\nTO ATHROISMA TON TETRAGONON TON %d AKERAION POU EDOSES EINAI %d\n", N, athroisma);
  system("PAUSE");  /* pause */
}
```

Τι κάνει το επόμενο πρόγραμμα; Ποιο το νόημα του περιεχομένου των μεταβλητών num1 και num2;

```
#include <stdio.h>
main()
{
int a,num,num1,num2;
num1=num2=0;
for (a=1;a<=100;++a)
     {
     printf("Dwse enan arithmo: ");
     scanf("%d",&num);
     switch (num%2)
          {
          case 0:
               ++num2;
               break;
          case 1:
               ++num1;
               break;
          }
     }
printf ("Edwses %d perittous kai %d artious", num1, num2);
}
```

- Με απλά λόγια, το πρόγραμμα ζητάει να δώσουμε 100 αριθμούς και στο τέλος εμφανίζει πόσους 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.

Πόσο είναι το k στο παρακάτω πρόγραμμα;

```c
#include <stdio.h>
main()
{
    int i,j, k=4;
    for(i=1;i<=10;i=i+2)
    {
        k++;
        printf("k=%d\t", k);
        for(j=1;j<5;j++)
        k=k+2;

    }
    printf("k= %d", k);
}
```

1η for	k++	2η for
i=1	k=5	7,9,11,13
i=3	k=14	16, 18, 20, 22
i=5	k=23	25, 27, 29, 31
i=7	k=32	34, 36, 38, 40
i=9	k=41	43, 45, 47, 49

```
k=5     k=14    k=23    k=32    k=41    k= 49
--------------------------------
Process exited after 2.994 seconds with return value 0
Press any key to continue . . .
```

ARTIOI.cpp.

Να γραφεί πρόγραμμα το οποίο να διαβάζει συνεχώς ακέραιους αριθμούς μέχρι να διαβάσει Ν θετικούς άρτιους. Τότε να σταματά και να εμφανίζει το πλήθος των αριθμών που διαβάστηκαν και τον μικρότερο από αυτούς. Το Ν θα δίνεται από το πληκτρολόγιο.

Υπόδειξη:

Θα χρησιμοποιήσετε -μεταξύ άλλων- και 3 μεταβλητές. Μία για το πλήθος των *αρτίων*, μία για το πλήθος *όλων των αριθμών* (μετρητής) και μία για την αποθήκευση του *μικρότερου*. Οι μεταβλητές αυτές θα αλλάζουν τιμές μέσα στον βρόχο της επανάληψης.

Παράδειγμα εκτέλεσης προγράμματος – με **κόκκινο** τα δεδομένα από το πληκτρολόγιο

```
DOSE PLHTHOS ARTION: 3

DOSE ENAN AKERAIO ARITHMO: 8
DOSE ENAN AKERAIO ARITHMO: -5
DOSE ENAN AKERAIO ARITHMO: 1
DOSE ENAN AKERAIO ARITHMO: -7
DOSE ENAN AKERAIO ARITHMO: -2
DOSE ENAN AKERAIO ARITHMO: 4

DIAVASTHKAN SYNOLIKA 6 ARITHMOI
O MIKROTEROS AKERAIOS POU DOTHIKE HTAN -7
```

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

main()
{
    int artios, plithos_arithmwn, plithos_artiwn, min=32767, N, K;

    printf("DOSE PLHTHOS ARTION: ");
    scanf("%d", &plithos_artiwn);



    N=0;
    while (N<plithos_artiwn)
    {

        printf("DOSE ENAN AKERAIO ARITHMO: ");
        scanf("%d", &K);

        plithos_arithmwn++;
        artios=K%2;
        if ((K>0)&&(artios==0))  N++; //Metratme to plithos twn thetikwn artiwn
        if(min>K) min=K;

    }
    printf ("\nDIAVASTHKAN SYNOLIKA %d ARITHMOI\n", plithos_arithmwn);
    printf ("O MIKROTEROS AKERAIOS POU DOTHIKE HTAN %d\n", min);


  system("PAUSE");  /* pause */
}
```

Εντολή (statement) - break

- έχει ως αποτέλεσμα την έξοδο του προγράμματος από τον (πιο εσωτερικό) βρόχο ή switch στο οποίο αυτή περιέχεται
  - το πρόγραμμα συνεχίζει με τις εντολές που ακολουθούν μετά το βρόχο

```
1.    int i, N, sum;
2.
3.    printf("Enter integer (N): ");
4.    scanf("%d", &N);
5.
6.    for (i = 0, sum = 0; i < N; ++i) {
7.      if (i == 5) break;
8.      sum = sum + i;
9.    }
10.
11.   printf("Sum = %d\n", sum);
```

Θα υπολογίσει το πολύ το άθροισμα των αριθμών 1+2+3+4, ανεξάρτητα από την τιμή του N (καθώς ο βρόχος σταματά όταν το i πάρει την τιμή 5)

- επιστρέφει τη ροή του προγράμματος στον έλεγχο της συνθήκης σε ένα βρόχο (τον πιο εσωτερικό)
  - δε διακόπτεται ο βρόχος, αλλά η τρέχουσα επανάληψη

```
1.    int i, N, sum;
2.
3.    printf("Enter integer (N): ");
4.    scanf("%d", &N);
5.
6.    for (i = 0, sum = 0; i < N; ++i) {
7.      if (i == 5) continue;
8.      sum = sum + i;
9.    }
10.
11.   printf("Sum = %d\n", sum);
```

Θα υπολογίσει το άρθοισμα: 1+2+...+n-5 (όταν το i πάρει την τιμή 5, η γραμμή 8 δε θα εκτελεστεί, όμως ο βρόχος δε θα τερματιστεί, αλλά θα συνεχίσει για i = 6, ...)

Χαρακτηριστικά στοιχεία πινάκων

- Βασικά χαρακτηριστικά πινάκων:
  - Ονομασία
  - Διάσταση (1, 2, …πολλών διαστάσεων)
  - Τύπος δεδομένων που φιλοξενούνται (int, float, char,…)
  - Αριθμός των στοιχείων σε κάθε διάσταση του.
- Ένα στοιχείο του πίνακα διακρίνεται από τα υπόλοιπα με τις συντεταγμένες του (εντοπισμός θέσης).
- Η διάσταση προσδιορίζει και τον αριθμό των ακεραίων δεικτών θέσης που χρειάζονται για γίνει διακριτό κάποιο στοιχείο του πίνακα (δηλαδή, οι συντεταγμένες του στοιχείου):
  - για πίνακες 1 διάστασης απαιτείται 1 δείκτης θέσης: Ai
  - για πίνακες 2 διαστάσεων απαιτούνται 2 δείκτες θέσης: Ai,j κ.ο.κ

Μονοδιάστατοι πίνακες

- Ένας πίνακας μιας διάστασης είναι ουσιαστικά ένα διάνυσμα.

  Ένας πίνακας μιας διάστασης με όνομα Α που φιλοξενεί 10 πραγματικούς αριθμούς (βαθμούς φοιτητών) έχει τη δομή:

A =

5.4	3.2	4.5	6.7	6.5	8.2	9.3	4.3	7.3	5.6

Βαθμός 1ου φοιτητή

Βαθμός 2ου φοιτητή

Βαθμός 10ου φοιτητή

- Αναφορά σε κάποιο στοιχείο του πίνακα (διάκριση) γίνεται με τη βοήθεια του δείκτη θέσης του στοιχείου:

  A_1: 1ο στοιχείο *(5.4),* ***A_5: 5ο στοιχείο*** *(6.5)*

- Μαζική αναφορά:

  A_j : το jο στοιχείο, όπου j=1,..,10 *(με εντολές επανάληψης)*

Δήλωση μονοδιάστατων πινάκων στη C

- Η δήλωση γίνεται ως εξής:

  Τύπος_δεδομένων Όνομα_πίνακα [Πλήθος_στοιχείων]

- Παραδείγματα:

  float vathmoi[10]; *//για φιλοξενία 10 πραγματικών*

  int foitites[20]; *// για φιλοξενία 20 ακεραίων*

- Συχνά για το πλήθος των στοιχείων του πίνακα χρησιμοποιείται η οδηγία **#define**:

  #define N 10

  float vathmoi [N];

- Το μέγιστο πλήθος στοιχείων εξαρτάται από τον compiler.
- Βρίσκουμε το μέγεθος του πίνακα με τον τελεστή sizeof():

  sizeof(vathmoi)/sizeof(float) *// πλήθος στοιχείων N*

Αναφορά σε στοιχεία ενός μονοδιάστατου πίνακα

- Με το όνομα του πίνακα και τον δείκτη θέσης του στοιχείου μέσα σε αγκύλες [ ]: ***vathmoi [j]***
  - Το πρώτο στοιχείο: ***vathmoi [0]***
  - Το δεύτερο στοιχείο: ***vathmoi [1]***
  - Το τελευταίο στοιχείο: ***vathmoi [9]***
- Ο δείκτης θέσης εκτός από ακέραιος μπορεί να είναι μια μεταβλητή ή μια αριθμητική έκφραση:

```
j=2; i=4;
vathmoi [i] = 6.5; // το 5ο στοιχείο
vathmoi [i+j] = 9.3; // το 7ο στοιχείο
```

float vathmoi[10];

vathmoi[0]	5.4
vathmoi[1]	3.2
vathmoi[2]	4.5
vathmoi[3]	6.7
vathmoi[4]	6.5
vathmoi[5]	8.2
vathmoi[6]	9.3
vathmoi[7]	4.3
vathmoi[8]	7.3
vathmoi[9]	5.6

Το όνομα του πίνακα χωρίς τις αγκύλες δείχνει στο 1ο στοιχείο του πίνακα (δείκτης μνήμης).

Απόδοση τιμών (μονοδιάστατοι πίνακες)

- Κατά την διάρκεια της εκτέλεσης του προγράμματος:

pinakas [j] =8;

- Με την δήλωση του πίνακα (απόδοση αρχικών τιμών):

int A[5] = {5,8,2,6,3};

(A[0]=5; A[1]=8; A[2]=2; A[3]=6; A[4]=3;)

- Αν δεν υπάρχουν όλες οι τιμές:

int A[5] = {5,8,2};

(A[0]=5; A[1]=8; A[2]=2; A[3]=0; A[4]=0;)

- Απόδοση αρχικών μηδενικών τιμών:

int A[5] = {0};

- Είναι δυνατόν να μη δηλωθεί το πλήθος των στοιχείων όταν αποδίδονται αρχικές τιμές:

int A[] = {5,8,2,6,3};

Διάβασμα ενός μονοδιάστατου πίνακα (πληκτρολόγιο)

```c
#include <stdio.h>
#define N 5 // N ο αριθμός των στοιχείων του πίνακα pin
main()
{
int i, pin[N];
for (i=0;i<N; i++) { //ανάγνωση πίνακα
printf("give me the no. %d element:",i+1);
scanf("%d",&pin[i]);
}
printf("\nYou have just read:\n");
for (i=0;i<N; i++) // εμφάνιση στη οθόνη
printf("%d\n", pin[i]);
getchar();getchar();
}
```

```
give me the no. 1 element:22
give me the no. 2 element:8
give me the no. 3 element:99
give me the no. 4 element:0
give me the no. 5 element:1

You have just read:
22
8
99
0
1


--------------------------------
Process exited after 23.9 seconds with return value 0
Press any key to continue . . .
```

Αντίστροφη εμφάνιση των στοιχείων του πίνακα

```
#include <stdio.h>
#define N 5
main()
{
int i, pin[N];
for (i=0;i<N; i++)  {     //ανάγνωση του πίνακα
   printf("give me the no. %d element:",i+1);
    scanf("%d",&pin[i]);
}
printf("\nYou have just read in reverse:\n");
for (i=N-1;i>=0; i--)       // εμφάνιση (αντίστροφα) στη οθόνη
        printf("%d\n", pin[i]);
}
```

```
give me the no. 1 element:514
give me the no. 2 element:58
give me the no. 3 element:9
give me the no. 4 element:1
give me the no. 5 element:25

You have just read in reverse:
25
1
9
58
514
```

Αποδίδοντας τυχαίες τιμές στα στοιχεία ενός πίνακα

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 10  // ο αριθμός των στοιχείων
#define SIZE 50
main()
{
   int i, pin[N];
//αρχικοποίηση της rand()
   srand(time(NULL));
   for(i=0;i<N;i++)
         pin[i] = 1+rand()%SIZE; //τυχαίοι 1-50
   for(i=0;i<N;i++)
        printf("%d ", pin[i]);
   printf("\n");
   system("PAUSE");
}
```

Δημιουργούμε τυχαίους αριθμούς με την συνάρτηση **rand()** η οποία επιστρέφει έναν ακέραιο από 0 έως μια σταθερά που βρίσκεται στο αρχείο κεφαλίδας **stdlib.h** (RAND_MAX που συνήθως είναι 32767).

Πριν τη χρήση της rand() πρέπει να κληθεί στο πρόγραμμα σας η συνάρτηση **srand()** η οποία αρχικοποιεί την γεννήτρια χαρακτήρων rand().

Για να είναι η ακολουθία των τυχαίων αριθμών διαφορετική, κάθε φορά που την καλούμε την srand(), της δίνουμε σαν παράμετρο (όρισμα) τον χρόνο του συστήματος **time(NULL)**.
– αρχείο κεφαλίδας: **time.h**

MinMaxPin.cpp.

Δημιουργείστε ένα πρόγραμμα το οποίο να διαβάζει Ν τυχαίους θετικούς ακέραιους και να τους βάζει κάποιο μονοδιάστατο πίνακα. Το Ν θα δηλώνεται σαν σταθερά. Στην συνέχεια το πρόγραμμα να βρίσκει τον μικρότερο και τον μεγαλύτερο από αυτούς και να τοποθετεί τον μικρότερο στην πρώτη θέση του πίνακα και τον μεγαλύτερο στην τελευταία θέση του πίνακα. Οι αριθμοί που βρισκόταν στην πρώτη θέση και στην τελευταία θέση του πίνακα να καταλαμβάνουν τις θέσεις που βρισκόταν ο μικρότερος και ο μεγαλύτερος αριθμός αντίστοιχα.

*Υπόδειξη: Οι τυχαίοι ακέραιοι του πίνακα θα δημιουργούνται με την συνάρτηση rand(). Δείτε πως στο παράδειγμα της σελίδας **10** («Αποδίδοντας τυχαίες τιμές στα στοιχεία ενός πίνακα») στην εισήγηση με τίτλο «6. Οι πίνακες στη C - 1ο μέρος» (στα Έγγραφα >> ΘΕΩΡΙΑ >> Παρουσιάσεις για τη C). Στην ίδια παρουσίαση στη σελίδα 16, το παράδειγμα με τίτλο «Ποια είναι η θέση του μικρότερου στοιχείου» θα σας βοηθήσει να βρείτε τις θέσεις του μεγαλύτερου και του μικρότερου από όλους του ακεραίους. Ο μικρότερος θα τοποθετηθεί στη θέση του πρώτου στοιχείου του πίνακα και ο μεγαλύτερος στην τελευταία με χρήση μιας προσωρινής μεταβλητής.*

```
O arxikos pinakas einai:

  25  40  76  48  98  83  28   2  92  94

Meta tis antimetatheseis o neos pinakas pou prokyptei einai o:

   2  40  76  48  94  83  28  25  92  98

Press any key to continue . . .
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 10
#define SIZE 100
main()
{

   int i, pin[N];
   int Max=-1, Max_Position, Min=999, Min_Position, temp; // to temp xreiazetai gia thn antimetathesi

   srand(time(NULL));// Arxikopoihsh ths rand()


   // Edw apodisoume tyxaies times sta stoixeia tou pinaka kai ta ektypwnoume
   printf("O arxikos pinakas einai: \n\n");
   for (i=0; i<N; i++){
       pin[i] = 1 + rand() % SIZE; // Edw ekxvreitai mia tyxaia timi apo to 1 ews to 100
       printf("%4d", pin[i]);
   }

   // Edw ypologizoume to megisto kai to elaxisto stoixeio
       for (i=0; i<N; i++)
    {
      if (pin[i] > Max)  {
                  Max = pin[i];
                  Max_Position = i;
                  }
     if (pin[i] < Min)  {
                  Min = pin[i];
                  Min_Position = i;
                  }
   }

   //Edw antimetathetoume to proto stoixeio tou pinaka me to elaxisto stoixeio
     temp=pin[0];
     pin[0]= Min;
     pin[Min_Position]= temp;

   //Edw antimetathetoume to teleftaio stoixeio tou pinaka me to megisto stoixeio
     temp=pin[N-1];
     pin[N-1]= Max;
     pin[Max_Position]= temp;

   printf("\n\nMeta tis antimetatheseis o neos pinakas pou prokyptei einai o: \n\n");
   for (i=0; i<N; i++) printf("%4d", pin[i]);
   printf("\n\n");

  system ("Pause");
}
```

MaxPin.cpp.

Να γραφεί ένα πρόγραμμα, το οποίο θα δημιουργεί ένα μονοδιάστατο πίνακα Ν ακεραίων και αφού τον «γεμίσει» με τυχαίους τριψήφιους θετικούς ακεραίους (δηλαδή μεταξύ 100 και 999) θα βρίσκει και θα εκτυπώνει το πλήθος όσων από αυτούς βρίσκονται εκτός του διαστήματος [450, 550]. Το Ν θα δίδεται από το χρήστη στην αρχή του προγράμματος και θα γίνεται σχετικός έλεγχος ότι δεν υπερβαίνει το 150.

*Υπόδειξη: Οι τυχαίοι ακέραιοι του πίνακα θα δημιουργούνται με την συνάρτηση rand(). Δείτε πως στο παράδειγμα της σελίδας **10** («Αποδίδοντας τυχαίες τιμές στα στοιχεία ενός πίνακα») της εισήγησης με τίτλο «6. Οι πίνακες στη C - 1ο μέρος» (στα Έγγραφα >> ΘΕΩΡΙΑ >> Παρουσιάσεις για τη C). Διαβάστε επίσης το αρχείο με τίτλο «Δημιουργία τυχαίων αριθμών με στη C» (στα Έγγραφα >> ΕΡΓΑΣΤΗΡΙΟ και βρείτε πως θα δημιουργείτε τριψήφιους αριθμούς. Στην ίδια εισήγηση στη σελίδα 14 το παράδειγμα με τίτλο «Πόσα στοιχεία του πίνακα είναι μικρότερα του 100» θα σας βοηθήσει να βρείτε πόσοι από αυτούς βρίσκονται εκτός του διαστήματος [450, 550]. Δείτε επίσης το παράδειγμα με τίτλο «Αν δεν γνωρίζουμε πόσα στοιχεία έχει ο πίνακας...» στην σελίδα 12.*

```
Dose enan arithmo mexri to 150: 50


O pinakas pou dimiourgeitai einai o:

 588 926 284 724 585 658 975 955 557 137 484 164 490 240 202 144 867 291 951 510 493 558 294 180 553 863 941 119 655 956 685 239 766 149
703 418 227 603 420 104 646 700 168 905 538 313 599 311 246 426


To plithos twn stoixeion pou vriskontai ektos tou diastimatos [450-550]: 45


Press any key to continue . . .
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 150
main()
{

   int arithmos, i, pin[N];
   int M=0;

    srand(time(NULL)); //Edw arxikopoioume thn rand()

    do
    {
        printf("Dose enan arithmo mexri to 150: ");
        scanf("%d", &arithmos);
    }   while ((arithmos > 150) || (arithmos<1));


  // Edw apodidei tyxaies times sta stoixeia tou pinaka
    printf("\n\nO pinakas pou dimiourgeitai einai o: \n\n");
    for (i=0; i<arithmos; i++){
       pin[i] = 100 + rand() %899; // Edw ekxorei mia tyxaia timi apo to 100 ews to 999
       printf("%4d", pin[i]);
    }

  // Edw vriskei to megisto kai to elaxisto stoixeio
     for (i=0; i<arithmos; i++)   {
          if ((pin[i] < 450) || (pin[i] > 550)) {
            M++;
        }

     }

   printf("\n\n\nTo plithos twn stoixeion pou vriskontai ektos tou diastimatos [450-550]: %d \n\n\n", M);

  system ("Pause");
}
```

Συμβολοσειρές

- Συμβολοσειρά ονομάζουμε μια οποιαδήποτε ακολουθία αλφαριθμητικών χαρακτήρων:
- ***«Προγραμματισμός», «Ελληνικό Μεσογειακό Πανεπιστήμιο»***
- Στην C δεν υπάρχει ιδιαίτερος τύπος για τον χειρισμό των συμβολοσειρών.
- Για την αποθήκευση τους χρησιμοποιούμε την δομή του πίνακα τύπου char και φυσικά τις διαχειριζόμαστε όπως ένα πίνακα:
- ***char line[80]; // δήλωση συμβολοσειράς με 80 χαρακτήρες***
- ***line[i] = getchar(); //απόδοση τιμής σε στοιχείο της σειράς***
- Αν πρέπει λοιπόν να διαχειριστούμε Ν χαρακτήρες ο πίνακας της συμβολοσειράς θα πρέπει να δηλωθεί με Ν+1 χαρακτήρες.
- Υπάρχουν ειδικές συναρτήσεις για τη διαχείριση των συμβολοσειρών (#include <string.h>)

Παρατηρήσεις για την αποθήκευση συμβολοσειρών

- Ο τελευταίος χαρακτήρας μιας συμβολοσειράς είναι ο «**\0**» **(μηδενικός χαρακτήρας – null character).**
- Ο μηδενικός χαρακτήρας «**\0**» είναι διαφορετικός από τον χαρακτήρα 0 (μηδέν). Ο μηδενικός χαρακτήρας έχει την τιμή 0 (εσωτερική αναπαράσταση:00000000), ενώ ο χαρακτήρας «0» έχει την τιμή 48 (00110000). Δηλαδή η εντολή:
- ***printf("%d, %d",'\0','0'); //*** *θα δώσει σαν αποτέλεσμα:* ***0, 48***
- Οι τιμές που δίνουμε στις μεταβλητές που έχουν δηλωθεί σαν συμβολοσειρές, οριοθετούνται με λατινικά εισαγωγικά διπλά (**") ή απλά ('). Προσοχή το 'A' δεν είναι ίσο με το "A" :**
  - ***Το 'A' χρειάζεται 1 byte μνήμης (01000001)***
  - ***Το "A" χρειάζεται 2 bytes (‘A’,’\0’ ==> 0100000100000000)***
- Το "A" είναι μία συμβολοσειρά που περιέχει το γράμμα 'A' και στο τέλος τον χαρακτήρα ‘\0' .

Αρχικές τιμές (με τη δήλωση)

- Όπως στους μονοδιάστατους πίνακες:

  char str1[6] = {'h', 'e','l','l','o','\0'};

  (Το str1[0] είναι h, το str1[1] είναι e και το str1[5] είναι ’\0’)

- Εναλλακτικά:

  char str2[8] = "hello"; //μπαίνει αυτόματα το '\0' στο τέλος

- Επίσης:

  char str2[] = "hello"; //το πλήθος υπολογίζεται αυτόματα

- Αν δώσουμε: **char str[20] = "hello";** δεσμεύουμε 20 θέσεις μνήμης ενώ χρειαζόμαστε μόνο 6. Οι θέσεις που περισσεύουν παίρνουν τη τιμή ’\0’ .
- Η εντολή **char str[20] = {'\0'};** δίνει την τιμή '**\0**' σε όλα τα στοιχεία της συμβολοσειράς str.
- Η **sizeof(str)** επιστρέφει το μέγεθος όλου του πίνακα.

Διάβασμα από το πληκτρολόγιο – scanf(), gets()

- Με την γνωστή **scanf() με χρήση του προσδιοριστή %s:**

  printf ("What's your name :"); scanf("%s", str);

  - Προσοχή: δεν βάζουμε τον χαρακτήρα & μπροστά από το str διότι το str είναι η διεύθυνση του 1ου στοιχείου της συμβολοσειράς.
  - Επίσης προσοχή: Η **scanf() σταματά τη ανάγνωση μόλις συναντήσει τον κενό χαρακτήρα (space) ή το χαρακτήρα ελέγχου για τη αλλαγή γραμμής ('\n').**

- Εναλλακτικά, για το διάβασμα μιας συμβολοσειράς από το πληκτρολόγιο μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **gets():**

  printf("What's your name:"); gets(name);

- Η **gets()** διαβάζει συνεχώς από το πληκτρολόγιο μέχρι να συναντήσει τον χαρακτήρα '\n' (ακόμα και τους κενούς χαρακτήρες). Αντικαθιστά δε τον '\n' με τον '\0' .

Εμφάνιση συμβολοσειρών στη οθόνη

- Με την συνάρτηση ***printf() και τον προσδιοριστή %s:***
- ***printf("Το onoma sou einai: %s\n", onoma);***
- Η printf() εμφανίζει όλους τους χαρακτήρες μέχρι να συναντήσει το μηδενικό χαρακτήρα '\0'.
- Με την printf() μπορούμε να εμφανίσουμε μέρος της συμβολοσειράς:

  printf("Το onoma sou einai: %s\n", onoma+2);

  (εμφανίζει την συμβολοσειρά από τον 3ο χαρακτήρα και μετά)
- Με τη συνάρτηση ***puts():***

  printf("What's your name:"); gets(name);

  printf("Hello "); puts(name);
- Η puts() μετά την εμφάνιση της συμβολοσειράς, τυπώνει τον χαρακτήρα '\n'.

Τρεις διαφορετικοί τρόποι να διαβάσουμε μια συμβολοσειρά

```c
#include <stdio.h>
main() {      // με scanf() & printf()
   char line [20];
   printf("Enter line: ");
   scanf("%s", line);
   printf("Line: %s\n", line);
}
```

```c
#include <stdio.h>
main() {      // με gets() & puts()
  char line[30];
   printf("Enter line: ");
   gets(line);
   printf("Line: ");
   puts(line);
}
```

```c
#include <stdio.h>
main() {       // με την getchar();
   char line[30], ch;
   int i = 0;
   printf("Enter line: ");
   while(ch != '\n') {    //μέχρι το newline
         ch = getchar();
         line[i] = ch;
         i++;
   }
   line[i] = '\0'; // για να γίνει συμβολοσειρά
   printf("Line: %s", line);
}
```

ANASTROFH_LEKSEON.cpp:

Γράψτε ενα πρόγραμμα που να διαβάζει από το πληκτρολόγιο μια λέξη και να την τυπώνει ανάστροφα πχ "HELLO" -> "OLLEH". Θεωρείστε ότι η λέξη που δίνεται δεν υπερβαίνει τους 20 χαρακτήρες.

<u>Υπόδειξη</u>: Χρησιμοποιήστε ένα μονοδιάστατο πίνακα τύπου char 20 θέσεων `char L[20]` *για να αποθηκεύσετε την λέξη. Για το διάβασμα και την εκτύπωση μιας λέξης χρησιμοποιήστε τον προσδιοριστή* `%s` *στην αντίστοιχη σειρά ελέγχου Π.χ.* `scanf("%s", &L)` `printf("... %s ...", L)`

Παράδειγμα εκτέλεσης προγράμματος – με **ΚΟΚΚΙΝΟ** τα δεδομένα από το πληκτρολόγιο

```
DOSE MIA LEKSH MEXRI 20 XARAKTHRES: COMPUTER
H ANASTROFH LEKSH EINAI: 'RETUPMOC'
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{

    int i, N;
    char L[20], CH;

    printf("DOSE MIA LEKSH MEXRI 20 XARAKTHRES GIA NA STHN TYPWSW ANASTROFA:   ");
    scanf("%s", L);

    N= strlen(L);

    for (i=0; i<N/2; i++){
        CH = L[i];
        L[i] = L[N-i-1];
        L[N-i-1] = CH;
    }

     printf("\nH ANASTROFH LEKSH EINAI: '%s'\n\n", L);

system ("Pause");
}
```

Για τη λέξη COMPUTER: N=8			
i=0	i=1	i=2	i=3
CH = L[0]; C L[0] = L[7]; L[7] = CH;	CH = L[1]; O L[1] = L[6]; L[6] = CH;	CH = L[2]; M L[2] = L[5]; L[5] = CH;	CH = L[3]; P L[3] = L[4]; L[4] = CH;
L[0] = R; L[7] = C;	L[1] = E; L[6] = O;	L[2] = T; L[5] = M;	L[3] = U; L[4] = P;

PALINDROMES_LEKSEIS.cpp:

Γράψτε ενα πρόγραμμα που να διαβάζει από το πληκτρολόγιο μια λέξη και να ελέγχει αν είναι ***παλίνδρομη***, δηλαδή αν ταυτίζεται με την ανάστροφή της (παραδείγματα: LEVEL", "MADAM", "ANNA"). Θεωρείστε ότι η λέξη που δίνεται δεν υπερβαίνει τους 20 χαρακτήρες.

Παραδείγματα εκτέλεσης προγράμματος – με **κόκκινο** τα δεδομένα από το πληκτρολόγιο

```
1) DOSE MIA LEKSH MEXRI 20 XARAKTHRES: LEVEL
   H LEKSH 'LEVEL' EINAI PALINDROMH

2) DOSE MIA LEKSH MEXRI 20 XARAKTHRES: HELLO
   H LEKSH 'HELLO' DEN EINAI PALINDROMH
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{

   int i, N, check=0;

   char L[20];

   printf("DOSE MIA LEKSH MEXRI 20 XARAKTHRES GIA NA ELEGKSW EAN EINAI PALINDROMH:  ");
   scanf("%s", L);

   N= strlen(L);

   for (i=0; i<N/2; i++){
      if (L[i] != L[N-i-1]) {
          check=1; // O PRWTOS ME TON TELEYTAIO XARAKTHRA THS LEKSHS DIAFEROUN
          break;
          }
   }

    if (check == 1)
      printf("\nH LEKSH '%s' DEN EINAI PALINDROMH.", L);
    else
     printf("\nH LEKSH '%s' EINAI PALINDROMH.", L);

      printf("\n\n");

system ("Pause");
}
```