Δομημένος προγραμματισμός

https://www.needpix.com

Οι εντολές επανάληψης (while, do-while, for)

Περίγραμμα της ύλης που θα καλυφθεί

- Η εντολή while
- Η εντολή do-while
- Βρόγχοι με τη εντολή for
- Ένθετοι βρόγχοι

https://commons.wikimedia.org

Γενικά για τις εντολές επανάληψης

- Συχνά στο προγραμματισμό είναι επιθυμητή η πολλαπλή εκτέλεση μιας ενότητας εντολών, είτε
  - για ένα συγκεκριμένο αριθμό επαναλήψεων, είτε
  - για όσο ισχύει κάποια συνθήκη.
- Τέτοιου είδους εντολές στη C είναι:
  - **while**
  - **do - while**
  - **for**
  - **goto** (δεν την χρησιμοποιούμε…)
- Ο έλεγχος της επανάληψης γίνεται με τη βοήθεια λογικών εκφράσεων/προτάσεων.
- Οι δομές που σχηματίζουν οι εντολές επανάληψης ονομάζονται «βρόγχοι».

Η σύναξη της εντολής while

- Η γενική σύνταξη της εντολής είναι:

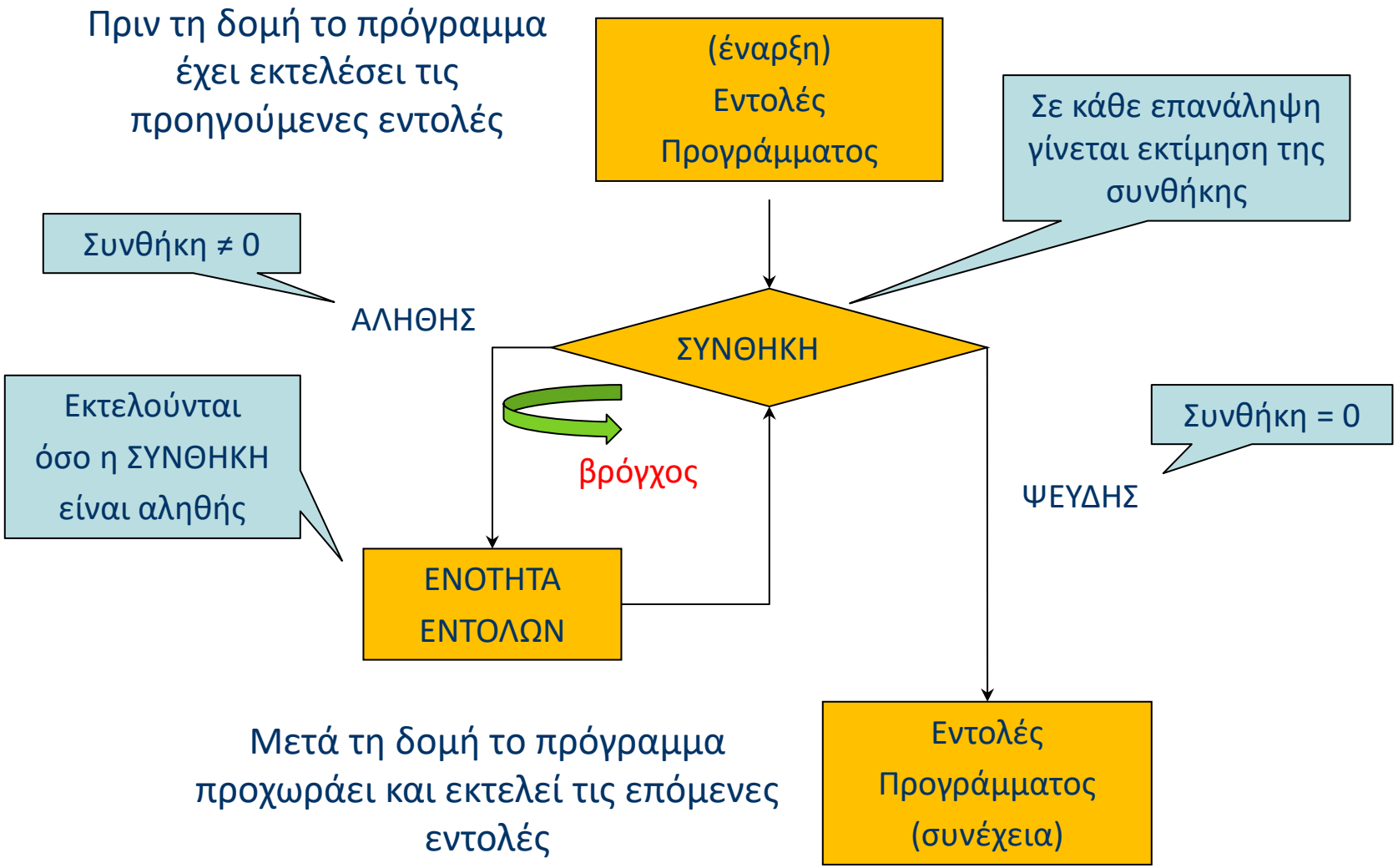
  while (<ΣΥΝΘΗΚΗ>)

  ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ

```
i=1;
while (i<5)
{
    printf("one more loop to go...\n");
    i=i+1;
}
```

- Η *ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ* μπορεί να περιλαμβάνει μια εντολή ή πολλές εντολές (block) που περικλείονται σε άγκιστρα (**{**,**}**).
  - Οι εντολές (ή η εντολή) της *ΕΝΟΤΗΤΑΣ_ΕΝΤΟΛΩΝ* εκτελούνται όσο ισχύει η *<ΣΥΝΘΗΚΗ>* (είναι αληθής).
  - Μια ή περισσότερες εντολές μέσα στη ενότητα εντολών «φροντίζουν» για το πότε να γίνει η συνθήκη ψευδής.
  - Όταν η *<ΣΥΝΘΗΚΗ>* γίνει ψευδής, τότε σταματά η εκτέλεση των εντολών της ενότητας.
  - Θυμηθείτε:
    - *<ΣΥΝΘΗΚΗ>* αληθής: διάφορη του μηδενός
    - *<ΣΥΝΘΗΚΗ>* ψευδής: ίση με μηδέν

Γραφική αναπαράσταση της δομής while



Παρατηρήσεις στη while

- Η *<ΣΥΝΘΗΚΗ>* μπορεί να είναι *λογική πρόταση*, *έκφραση συσχετισμού*, *αποτέλεσμα κάποιας πράξης*, είτε ακόμα και κάποια *μεταβλητή* ή *τιμή*.
- Αν η *ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ* περιέχει μόνο μια εντολή, τότε τα άγκιστρα (**{**,**}**) μπορούν να παραλειφθούν.
- Αν βάζουμε το ελληνικό ερωτηματικό «**;**» στο τέλος της *while* τότε η εντολή θεωρείται ξεχωριστή και ουσιαστικά δεν εκτελείται η *ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ*.
- Αν η *<ΣΥΝΘΗΚΗ>* σε μια *while* είναι πάντα αληθής, τότε ο βρόγχος δεν τερματίζει ποτέ (ατέρμων βρόγχος – infinity loop). Παράδειγμα ατέρμονος βρόγχου:

```
while (1)
```

Στη **while** η *<ΣΥΝΘΗΚΗ>* ελέγχεται στη *αρχή* του βρόγχου. Άρα αν η *<ΣΥΝΘΗΚΗ>* είναι ψευδής πριν ξεκινήσει ο βρόγχος, τότε οι εντολές της *ΕΝΟΤΗΤΑΣ_ΕΝΤΟΛΩΝ* δεν εκτελούνται *ποτέ* (ούτε μια φορά).

Παράδειγμα -1 (while)

Εκτέλεση

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

```
#include <stdio.h>
main()
{
    int i =1;
    while (i<5)
    {
        printf("%d\n",i);
        i=i+1;
    }
    printf(“end“);
getchar(); getchar();
}
```

Δοκιμάστε:

```
while (++i<5)
    printf("%d\n",i);
```

```
while (i++<5)
    printf("%d\n",i);
```

Παραδείγματα -2 (while)

Παρουσίαση αριθμών από 0 έως N. Το N δίδεται.

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

Εύρεση αθροίσματος 1+2+3+…+N

```
int i,N, sum;
printf("Dwste N:");
scanf("%d", &N);
i = 1;
sum=0;
while (i <= N) {
        sum=sum+i;
        i++;
 }
printf("SUM=%d\n", sum);
```

Για την άθροιση των τιμών

Η εντολή do -while

- Η γενική σύνταξη της εντολής είναι:

  do

  ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ

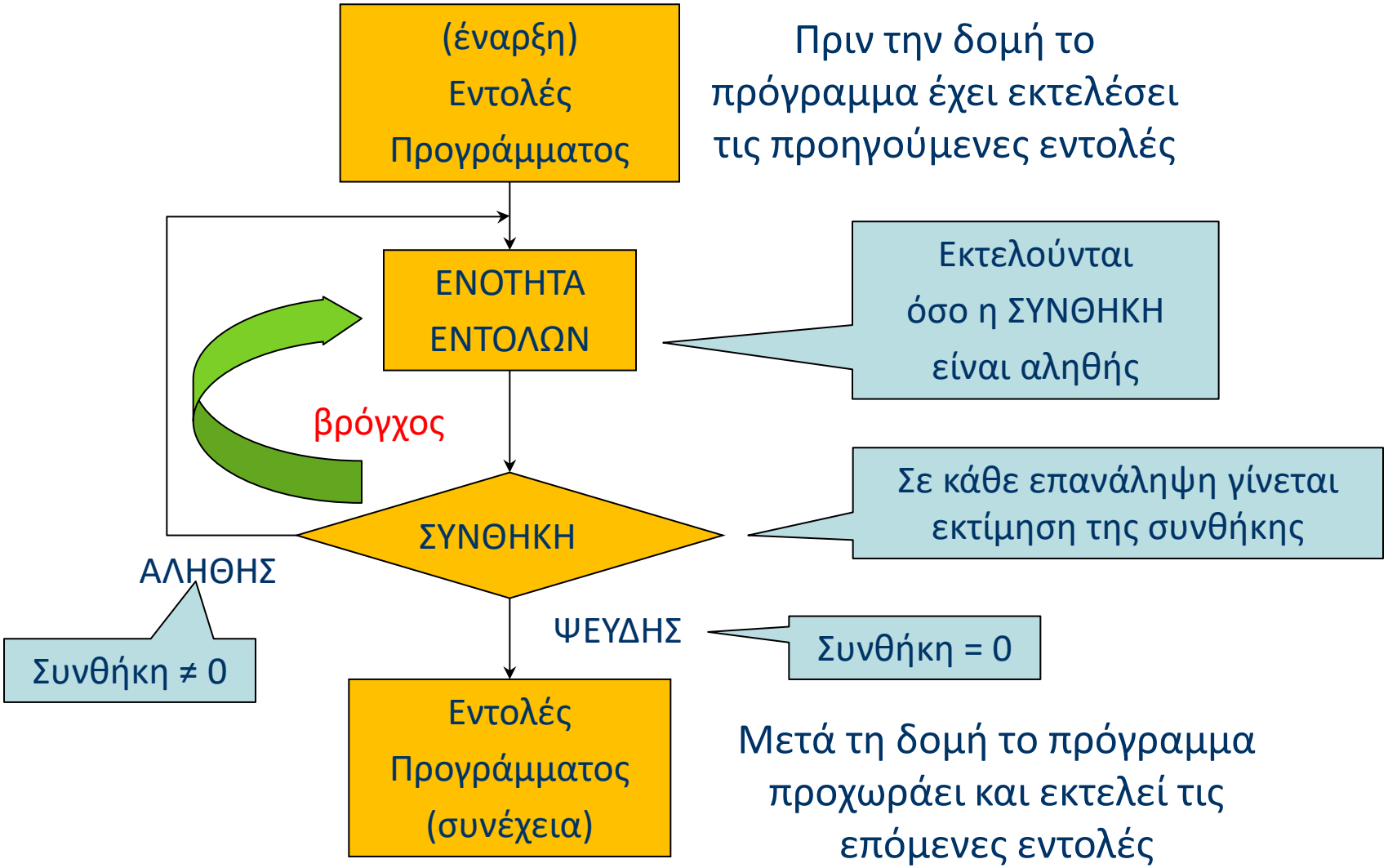
  while (<ΣΥΝΘΗΚΗ>);

```
i=1;
do
{
      printf("one more loop to go...\n");
       i=i+1;
} while (i<5);
```

- Συνεχής εκτέλεση της *ΕΝΟΤΗΤΑΣ_ΕΝΤΟΛΩΝ* μέχρι η *<ΣΥΝΘΗΚΗ>* να μην είναι αληθής.
- Και εδώ κάποιες εντολές μέσα στη *ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ* «φροντίζουν» για το πότε να γίνει η συνθήκη ψευδής.
- Το *while* στο τέλος της εντολής *do-while* πρέπει να τελειώνει με το ελληνικό ερωτηματικό (**;**)

Στη **do-while** η *<ΣΥΝΘΗΚΗ>* ελέγχεται στο *τέλος* του βρόγχου. Άρα ακόμα και αν η *<ΣΥΝΘΗΚΗ>* είναι ψευδής, οι εντολές της *ΕΝΟΤΗΤΑΣ_ΕΝΤΟΛΩΝ* εκτελούνται τουλάχιστον *μια φορά*).

Γραφική αναπαράσταση της δομής do-while



Παραδείγματα -2 (do-while)

Παρουσίαση αριθμών από 0 έως N. Το N δίνεται.

```
int i,N;
printf("Dwste N:");
scanf("%d", &N);
i = 0;
do {
     printf("i is now %d!\n", i);
      i++;
}
while (i<N);
```

Εύρεση αθροίσματος 1+2+3+…+N

```
int i,N, sum;
printf("Dwste N:");
scanf("%d", &N);
i = 1;  sum=0;
do {
        sum=sum+i;
        i++;
} while (i <= N);
printf("SUM=%d\n", sum);
```

Η εντολή for

```
for(i=1;i<5;i++)
{
    printf("one more loop to go...\n");
}
```

- Γενική σύνταξη της *for*:

for (Εντολή1; <ΣΥΝΘΗΚΗ>; Εντολή2)

ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ

- Η *Εντολή1* μπορεί να είναι και περισσότερες από μια εντολές που συνήθως καθορίζουν κάποιες αρχικές τιμές. Εκτελείται μία φορά.
- Η *Εντολή2* είναι μια παράσταση που συνήθως αλλάζει τη τιμή μιας μεταβλητής που βρίσκεται στη <*ΣΥΝΘΗΚΗ*>. 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 <ΣΥΝΘΗΚΗ> είναι αληθής

Παραδείγματα με τη for

Παρουσίαση αριθμών από 0 έως N. Το N δίνεται.

```
int i,N;
printf("Dwste N:");
scanf("%d", &N);
for (i = 0; i<N;i++)
{
    printf("i is now %d!\n", i);
}
```

Εύρεση αθροίσματος 1+2+3+…+N

```
int i,N, sum;
printf("Dwste N:");
scanf("%d", &N);
sum=0;
for (i = 0; i<=N;i++)
        sum=sum+i;
printf("SUM=%d\n", sum);
```

Παρατηρήσεις για τη for

- Χρησιμοποιούμε την *for* όταν γνωρίζουμε εκ των προτέρων τον αριθμό επαναλήψεων.
- Δεν βάζουμε το ελληνικό ερωτηματικό «**;**» στο τέλος της *for*:

```
for (i = 0; i< 1000; i++);  //απλά αυξάνει το i 1000 φορές
```

- Όταν η *Εντολή1* και η *Εντολή2* αποτελούνται από πολλές εντολές, τότε αυτές χωρίζονται με κόμμα (**,**):

```
for (i=0, k=0; (i < 5 && k < 3); i++, k++)
```

- Δεν είναι απαραίτητο να υπάρχουν και τα 3 τμήματα της *for*. Όμως υπάρχει πάντα το διαχωριστικό «**;**» μεταξύ των τμημάτων.

```
for (;;)  { εντολές } // ατέρμων βρόγχος – infinity loop
```

- Φυσικά αν η *<ΣΥΝΘΗΚΗ>* είναι ψευδής, δεν θα εκτελεστεί ποτέ η *ΕΝΟΤΗΤΑ_ΕΝΤΟΛΩΝ*.

Παρουσίαση αριθμών από 0 έως N. Το N δίνεται.

```
printf("Dwste N:");
scanf("%d", &N);
i = 0;
while (i < N) {
    printf("i is now %d!\n", i);
    i++;
}
```

```
printf("Dwste N:");
scanf("%d", &N);
i = 0;
do {
    printf("i is now %d!\n", i);
    i++;
} while (i<N);
```

```
printf("Dwste N:");
scanf("%d", &N);
for (i = 0; i<N;i++)
{
    printf("i is now %d!\n", i);
}
```

Πως δουλεύουν οι βρόγχοι που είδαμε μέχρι τώρα;

- Δίνουμε αρχική τιμή σ' ένα μετρητή βρόγχου και σε κάθε επανάληψη:
  - εξετάζουμε ένα κριτήριο τερματισμού,
  - εκτελούμε κάποιες εντολές,
  - αυξάνουμε τον μετρητή.

Εύρεση αθροίσματος 1+2+3+…+N

```
printf("Dwste N:");
scanf("%d", &N);
 i = 1;
sum=0;
while (i <= N) {
        sum=sum+i;
        i++;
}
printf("SUM=%d\n", sum);
```

```
printf("Dwste N:");
scanf("%d", &N);
 i = 1;  sum=0;
  do {
        sum=sum+i;
        i++;
} while (i <= N);
printf("SUM=%d\n", sum);
```

```
printf("Dwste N:");
scanf("%d", &N);
sum=0;
for (i = 1; i<=N;i++)
          sum=sum+i;
printf("SUM=%d\n", sum);
```

Και εδώ οι βρόγχοι δουλεύουν όπως πριν.

Όμως δεν χρειαζόμαστε πάντα μετρητή για να προξενήσουμε το τέλος του βρόγχου.

Υπάρχουν και άλλοι τρόποι να ικανοποιήσουμε το κριτήριο τερματισμού…

Εύρεση αθροίσματος Ν ακεραίων (for)

```
int i,N,A,sum;
printf("Dwste N:");
scanf("%d", &N); // πόσους αριθμούς ?
sum=0;  //για την άθροιση των αριθμών
for (i = 1; i<=N;i++)
{
        printf("Dwste akeraio:");
        scanf("%d",&A);      //διάβασμα αριθμού
        sum=sum+A;           // αυξάνω το άθροισμα
}
printf("SUM=%d\n", sum);
```

Σε κάθε επανάληψη διαβάζει έναν αριθμό και αυξάνει το άθροισμα

Εύρεση αθροίσματος N πραγματικών (while)

```
int i, N;
float F, sum;
printf("Dwste N:");  scanf("%d", &N);
i=1;
sum=0;
while  (i<=N) {
        printf("Dwste float:");
        scanf("%f",&F);
        sum=sum+F;
         i++;
}
printf("SUM=%f\n", sum);
```

Εύρεση αθροίσματος πραγματικών (do-while)

Σε αυτό το πρόβλημα το πλήθος των αριθμών δεν είναι γνωστό. Διαβάζονται συνεχώς αριθμοί μέχρι να διαβαστεί ένας πραγματικός ίσος με το 0.

```
#include <stdio.h>
main()
{
   double F, sum = 0;
   do  //ο βρόγχος εκτελείται τουλάχιστον 1 φόρα
   {
      printf("Enter a number: ");
      scanf("%f", &F);
      sum += F;
   }
   while(F != 0.0);
   printf("Sum = %.2lf",sum);
}
```

Δεν χρειαζόμαστε πάντα μετρητή βρόγχου.

Εδώ το κριτήριο τερματισμού είναι να διαβαστεί ένα 0

Εύρεση Μέσου Όρου Ν ακεραίων (do-while)

```
int i, N, A, sum;
float MO;
printf("Dwste N:");  scanf("%d", &N);
i=1;
sum=0;
do {
        printf("Dwste akeraio:");
        scanf("%d",&A);
        sum=sum+A;
         i++;
} while  (i<=N) ;
MO=(float)sum/N;
printf("Mesos Oros=%f\n", MO);
```

Ο μέσος όρος είναι πάντα πραγματικός (αποτέλεσμα διαίρεσης)

Μετατρέπουμε τη sum προσωρινά σε float (casting) για να έχουμε float αποτέλεσμα.

Η εντολή break

- Χρησιμοποιείται για ηθελημένο τερματισμό ενός βρόχου (*for, while* ή *do-while*). Επίσης για το τερματισμό της *switch*.
- Η εκτέλεση του *break* μέσα στο βρόγχο μεταφέρει τον έλεγχο του προγράμματος στην εντολή που ακολουθεί τον βρόγχο.

```
int i,N, sum;

printf("Dwste N:");  scanf("%d",
&N);

sum=0;

 for (i = 0; i<=N;i++) {

          if (i==5) break;

         sum=sum+i;

 }

printf("SUM=%d\n", sum);
```

Θα βρει το πολύ το άθροισμα των 1+2+3+4, ανεξάρτητα από την τιμή του N.
(ο βρόγχος σταματά όταν i=5)

Η εντολή continue

- Στο σημείο που χρησιμοποιείται στους βρόγχους διακόπτεται η εκτέλεση της *ΕΝΟΤΗΤΑΣ_ΕΝΤΟΛΩΝ* και ξεκινά την επόμενη επανάληψη.
- Δεν διακόπτεται ο βρόγχος, αλλά η τρέχουσα επανάληψη.

```
int i,N, sum;
printf("Dwste N:");  scanf("%d", &N);
sum=0;
 for (i = 0; i<=N;i++) {
          if (i ==5) continue;
         sum=sum+i;
 }
printf("SUM=%d\n", sum);
```

Θα βρει το άθροισμα:

1+2+… N -5

Για i=5 δεν θα εκτελεστεί ο βρόγχος.

Όμως ο βρόγχος θα συνεχιστεί, δεν θα σταματήσει! (ακόμα και αν i=5)

Διαφορά break-continue

```
#include <stdio.h>
int main()
{
 int x;
 for(x=0;x<10;x++)
 {
        if(x==5)
        {
            break;
        }
        printf("%d\n",x);
 }
 return 0;
}
```

```
0
1
2
3
4
```

```
#include <stdio.h>
int main()
{
 int x;
 for(x=0;x<10;x++)
 {
        if(x==5)
        {
            continue;
        }
        printf("%d\n",x);
 }
 return 0;
}
```

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

Πόσοι αριθμοί είναι μικρότεροι του 100

```
int i,N,A,M; //M: ο μετρητής –πάντα ακέραιος
printf("Dwste N:");
scanf("%d", &N);
i=1;
M=0; //Μηδενίζω τον μετρητή
while(i<=N)
{
        printf("Dwste akeraio:");
        scanf("%d",&A);
        if (A<100) M++; //Αυξάνω τον μετρητή
         i++;
}
printf("Diavasa %d arithmous <100\n", M);
```

Γενικότερο πρόβλημα:
Να βρεθεί το πλήθος των αριθμών που χαρακτηρίζεται από κάποια ιδιότητα

Βρείτε τον μικρότερο από Ν αριθμούς

```
int i,N,A,min;
printf("Dwste N:");
scanf("%d", &N);
printf("Dwste akeraio:");
scanf("%d",&A); //διαβάζω τον 1ο αριθμό –πριν τον βρόγχο
min=A;              //θεωρούμε σαν μικρότερο  τον 1ο αριθμό
for (i = 2; i<=N;i++)
{
        printf("Dwste akeraio:");
        scanf("%d",&A);
        if (A<min) min=A;
}
printf("To min einai: %d \n", min);
```

Κάθε φορά που βρίσκουμε ένα μικρότερο από τον min, αλλάζουμε τον min

Ένθετοι βρόγχοι

- Ένας βρόγχος μπορεί να συμπεριλαμβάνει και άλλες εντολές, καθώς επίσης και άλλους βρόγχους (βρόχοι μέσα σε βρόγχο).

Εμφανίζει τη προπαίδεια 1-10

```
main()
{
int i,j,n;

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

```
for( i=1; i<=4; i++ ){
        for( j=1; j<=5; j++ )
                printf("*");
}
```

Θα τυπωθούν 20 (4x5) αστερίσκοι.

Είναι απαραίτητος ο ένθετος βρόγχος;

Τι θα άλλαζε για τη προπαίδεια 1-5;

- Σε αυτές τις περιπτώσεις θα πρέπει να είμαστε προσεκτικοί στους δείκτες που χρησιμοποιούνται στους βρόγχους.

Ένθετοι βρόγχοι

- Ένας βρόγχος μπορεί να συμπεριλαμβάνει και άλλες εντολές, καθώς επίσης και άλλους βρόγχους (βρόχοι μέσα σε βρόγχο).

Εμφανίζει τη προπαίδεια 1-10

```
main()
{
int i,j,n;

for( i=1; i<=10; i++ ){
```

```
for( i=1; i<=4; i++ ){
        for( j=1; j<=5; j++ )
                printf("*");
}
```

(4x5) αστερίσκοι.

αι απαραίτητος ο
θετος βρόγχος;

```
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
```

Τι θα άλλαζε για τη προπαίδεια 1-5;

```
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
```

- Σε αυτές τις περιπτώσεις θα πρέπει στους δείκτες που χρησιμοποιούνται στους

Ένθετοι βρόγχοι -παραδείγματα

```
int i,j,n;
printf("Dwste n: "); scanf("%d", &n);
for( i=1; i<=n; i++ ){
        for( j=1; j<=i; j++ )
                printf("*");
        printf("\n");
}
```

Αποτέλεσμα:

```
Dwste n: 5
*
**
***
****
*****
```

Αν αντί για * βάλουμε i;
Αν αντί για * βάλουμε j;

```
for (i = 1; i <= n; i++) {
    for (j=1;j<=n-i;j++)
                printf (" ");
    for (j=1;j<=i;j++)
                printf ("*");
    printf("\n");
}
```

```
    *
   **
  ***
 ****
*****
```

Ένθετοι βρόγχοι -και άλλο παράδειγμα

```
printf("Please give n: ");
scanf("%d", &n);
for (i = 1; i <= n; i++)
{
    for( j=1; j<=i; j++ )
                printf("*");
    for (j=i;j<=2*n-i;j++)
                printf (" ");
    for (j=1;j<=i;j++)
               printf ("*");
    printf("\n");
}
```

Αποτέλεσμα:

```
*          *
**        **
***      ***
****    ****
***** *****
```

Ένθετοι βρόγχοι -και άλλο παράδειγμα

```
printf("Please give n: ");
scanf("%d", &n);
for (i = 1; i <= n; i++) {
    for (j=1;j<=n-i;j++)
                printf (" ");
    for (j=1;j<=i;j++)
                printf ("*");
     if (i != 1)
        for( j=2; j<=i; j++ )
                printf("*");
    printf("\n");
}
```

Αποτέλεσμα (n=5):

```
    *
   ***
  *****
 *******
*********
```

```
for (i = 1; i <= n; i++)  {
    for (k = n; k >= i; k--)
                printf(" ");
    for (j = 1; j <= i; j++)
                printf("%c", '*');
    for (j = 1; j <= i-1; j++)
                printf("%c", '*');
    printf("\n");
  }
```

Το ίδιο αποτέλεσμα

Λάθη με τους ένθετους βρόγχους

```
printf("Please give n: ");
scanf("%d", &n);
for( i=1; i<=n; i++ ){
        for( j=1; i<=n/2; j++ )
                printf("j");
        printf("\n");
}
```

```
printf("Please give n: ");
scanf("%d", &n);
for( i=1; i<=n; i++ ){
        for( j=1; j<=n/2; i++ )
                printf("j");
        printf("\n");
}
```

Περισσότερη εξάσκηση

- Λύστε τη σειρά των ασκήσεων που έχουν αναρτηθεί στο δικτυακό τόπο του μαθήματος.