ΕΠΛ 033: ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΓΙΑ ΜΗΧΑΝΙΚΟΥΣ

Μάριος Belk, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

Email: belk@cs.ucy.ac.cy

Δομές Επανάληψης

Θέματα Διάλεξης

- Εντολές while, for, do while
- Τελεστές postfix/prefix και σύνθετοι τελεστές
  Συναρτήσεις getchar και putchar
- Εφαρμογή επαναλήψεων σε σειρές χαρακτήρων

Δομές Εκτέλεσης

- **Διαδοχική εκτέλεση**
  - Στηρίζεται στην απλή παράθεση εκφράσεων/εντολών, η μια μετά την άλλη
- **Εκτέλεση με επιλογή**
  - Η ροή του προγράμματος “διακόπτεται” για να παρθεί μια απόφαση, να γίνει κάποια επιλογή
  - Το αποτέλεσμα της απόφασης καθορίζει την “κατεύθυνση” της ροής του προγράμματος
- ***Εκτέλεση με επανάληψη***
  - Μια ομάδα εκφράσεων/εντολών εκτελείται περισσότερο από μια φορά

Δομές Επανάληψης – Βρόχοι (repetition – loops)

- while(){}
- for(){}
- do{ } while()
- τελεστές postfix/prefix (++, --, ...) και σύνθετοι τελεστές

Χρήση Βρόχων

- **Για επανάληψη λειτουργικότητας**
  - μετρητής (counter)
    - επανέλαβε εάν ο μετρητής είναι μικρότερος/μεγαλύτερος από μια τιμή
  - συνθήκη/έλεγχος (conditional)
    - επανέλαβε εφόσον ισχύει η συνθήκη
  - σημαία (sentinel, flag)
    - επανέλαβε εαν διάφορο του EOF, -1 κτλ
  - συνδυασμός (με χρήση λογικών τελεστών)

Εντολή while

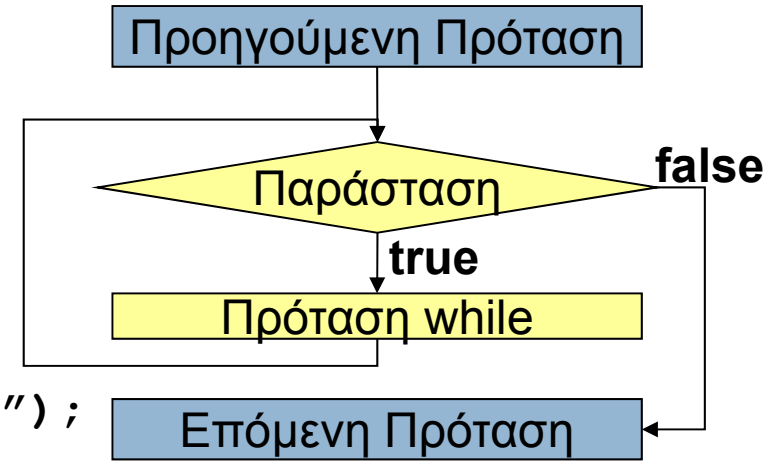
- Η εντολή ***while*** επαναλαμβάνει την εκτέλεση μιας πρότασης ενώσω η τιμή μιας λογικής παράστασης είναι αληθής (1)
- Η σύνταξη της εντολής είναι:

```
while (παράσταση)
   Πρόταση1;
```

```
while (παράσταση) Πρόταση1;
```

```
while (παράσταση)
{
   Πρόταση1_1;
   Πρόταση1_2;
   …
   Πρόταση1_N;
}
```

- Η *πρόταση1* (απλή ή σύνθετη) θα εκτελείται συνέχεια μέχρι η *παράσταση* της ***while*** να πάρει λογική τιμή false (0)



- Π.χ.

```
while (5>0)
{
   printf("Δε θα σταματήσω ποτέ\n");
}
```

Σύνταξη while

```
while  (έκφραση)
    εντολή;
```

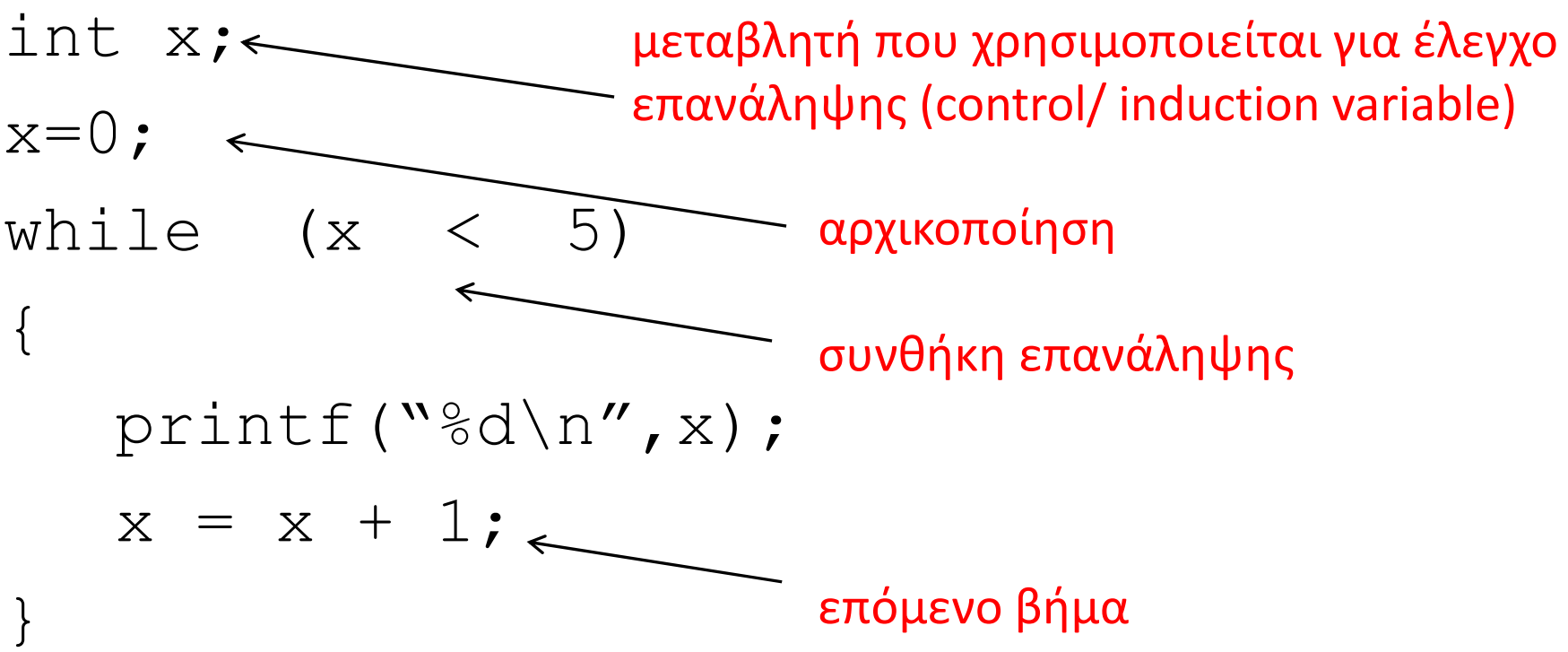
ή

```
while  (έκφραση) {
    εντολή;
    εντολή;
    …
}
```

Σημασία while

- Όσον η τιμή της έκφρασης (συνθήκης) είναι αληθής – δηλαδή διάφορη του μηδέν – εκτέλεσε τις εξαρτώμενες εντολές, *αλλιώς συνέχισε με τις εντολές που ακουλουθούν (μετά) το* while *block*

while με μετρητή



while (συν.)

x	x<5	output
0	1	0
1	1	1
2	1	2
3	1	3
4	1	4
5	0	

Παράδειγμα

Γράψτε έναν κώδικα που τυπώνει τους ακέραιους αριθμούς από το 1 μέχρι το 12. Κάθε αριθμός που διαιρείται με το 3 να τυπώνεται * δίπλα του. Ο κάθε αριθμός να τυπώνεται σε ξεχωριστή γραμμή.

Παράδειγμα – Κώδικας Λύσης

```
int i;   /* loop counter */
i = 1;
while (i <= 12){
    printf("%d",i);
    if( (i%3) == 0)
    {
         printf("*");
    }
    printf("\n");
    i = i + 1;
}
```

Trace Table

I	i<=12	diareitai_me 3	output
1	T	Όχι	1
2	T	Όχι	2
3	T	Ναι	3*
4	T	Όχι	4
5	T	Όχι	5
6	T	Ναι	6*
7	T	Όχι	7
8	T	Όχι	8
9	T	Ναι	9*
10	T	Όχι	10
11	T	Όχι	11
12	T	Ναι	12*
13	F		

Τελική τιμή του i : 13

Τελεστές postfix/prefix

προ-σημειογραφική (prefix)

++i;	<==>	i = i + 1;
--i;	<==>	i = i - 1;

μετα-σημειογραφική (postfix)

i++;	<==>	i = i + 1;
i--;	<==>	i = i - 1;

Description	Operator	Use
scope resolution	::	std::cout
post increment post decrement	++ --	k++ k--
pre increment pre decrement not unary minus unary plus	++ -- ! - +	++k --k ! isPrime() -5 +23
multiply divide modulo (remainder)	* / %	x * y i / j m % n
add subtract	+ -	x + y i - j
put get	<< >>	cout << k; cin >> i >> j;
less than less than or equal greater than greter than or equal	< <= > >=	2 < 3 ch1 <= ch2 i > k x >= z
equal not equal	== !=	5 == num1 age != 65
AND	&&	p && q
OR	||	p || q
conditional operator	?:	(age < 21)? 0 : 1
assignment operators	= += -= . . .	k = k + 1 k += 1 k -= 1 . . .

Prefix vs Postfix

```
i = 5;
x = ++i;
y = i++;
```

- Το **x είναι 6**, το **y είναι 6** και το **i είναι 7**
- Χρησιμοποιείτε -- ++ σε απλές εκφράσεις
  - `n = ++m;` /*first increment m, then assign its value to n*/
  - `n = m++;` /*first assign m's value to n, then increment m*/

Σύνθετοι Τελεστές Ανάθεσης

- `i+=k;` `<==> i = i + k;`
- `i*=k;` `<==> i = i * k;`
- `i`***`operator`*** `= k; <==> i = i` ***`op`*** `(k);`

Παράδειγμα

- Γράψτε κώδικα που υπολογίζει **το άθροισμα** μιας **απροσδιόριστου μεγέθους σειράς θετικών ακεραίων αριθμών**. Η σειρά εισάγεται από μονάδα εισόδου και τερματίζεται με την τιμή 0.

Δεν μπορεί να γίνει χρήση του while με μετρητή

Χρήσιμες Λειτουργικότητες

- **Τι** πρέπει να γίνει
  - Διάβασμα μια σειράς απροσδιόριστου μεγέθους που τερματίζεται με καθορισμένη τιμή
  - Υπολογισμός αθροίσματος μιας σειράς
- Χρήσιμες Λειτουργικότητες
  - Πώς διαβάζουμε μια σειρά απροσδιόριστου μεγέθους που τερματίζεται με καθορισμένη τιμή;
  - Πώς υπολογίζουμε το άθροισμα μιας σειράς;

Ανάγνωση σειράς απροσδιόριστου μεγέθους που τερματίζεται με καθορισμένη τιμή

- Απαιτεί εντολή επανάληψης (βρόχος) - ***σημαία*** η τελευταία τιμή
- Τυπική Δομή:

```
/*διάβασε το πρώτο στοιχείο*/
while(/*το στοιχείο δεν είναι η σημαία*/) {

        /* διάβασε επόμενο στοιχείο */

}
```

Διάβασμα σειράς απροσδιόριστου μεγέθους που τερματίζεται με 0

```
int number;
scanf(“%d”,&number);      /*diabase prwto stoixeio*/

while(number != 0){

scanf(“%d”,&number); /* διαβασε επομενο στοιχειο */

}
```

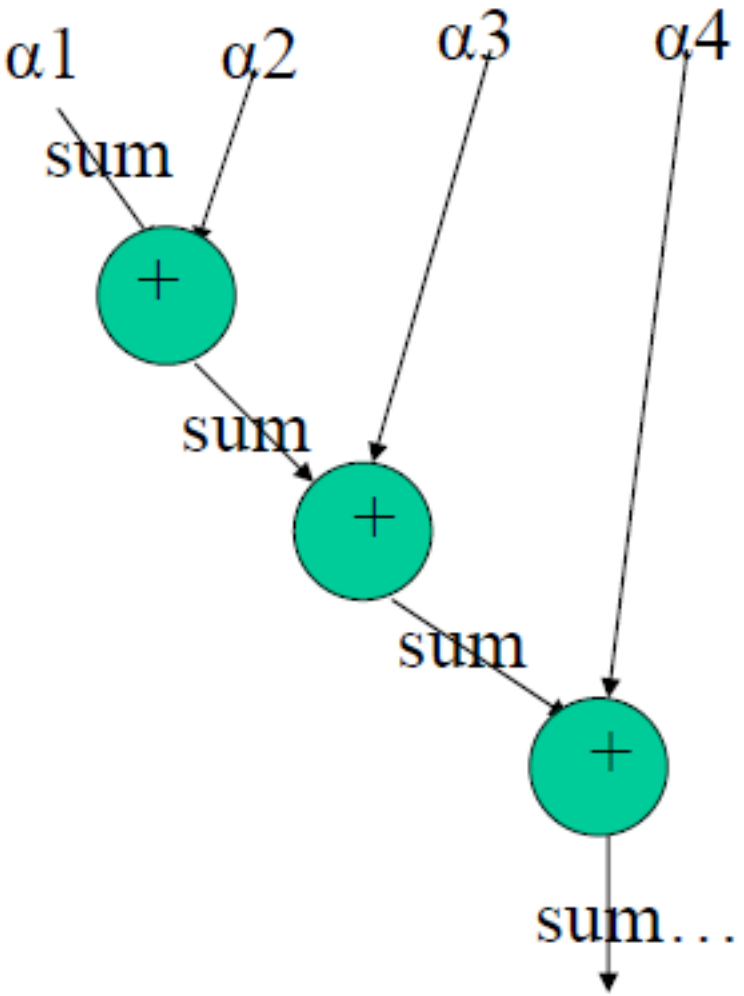
Διάσπαση: Έννοια του μερικού αποτελέσματος

- Όταν δεν έχουμε εντολή που επιλύει ένα πρόβλημα, διασπούμε το πρόβλημα μέχρι να βρούμε ένα υποπρόβλημα που υπάρχουν εντολές που το επιλύουν
  - Π.χ. δεν υπάρχει τελεστής στην C που μπορεί να προσθέσει *n* τιμές μαζί (όπου το n >2).

    Για n τιμές χρειάζονται __;__προσθέσεις.

  - n-1

Διάσπαση



Άθροισμα σειράς – Κώδικας

```
int number;        /* holds input number one at a time */
int sum;                                  /* current sum */

scanf("%d",&number);          /* diabase prwto stoixeio */
sum = 0;                                  /* arxikopoihsh */
while(number != 0){
     sum = sum + number;                  /* epeksergasia*/
     scanf("%d",&number); /* diavase epomeno stoixeio*/
}
printf("To athroisma tis seiras einai %d\n",sum); /*eksodos*/
```

Παράδειγμα

Γράψτε κώδικα που αναγνωρίζει τον πιο μεγάλο αριθμό σε μια απροσδιόριστου μεγέθους σειρά θετικών ακέραιων αριθμών. Η σειρά εισάγεται από μονάδα εισόδου και τερματίζεται με την τιμή 0

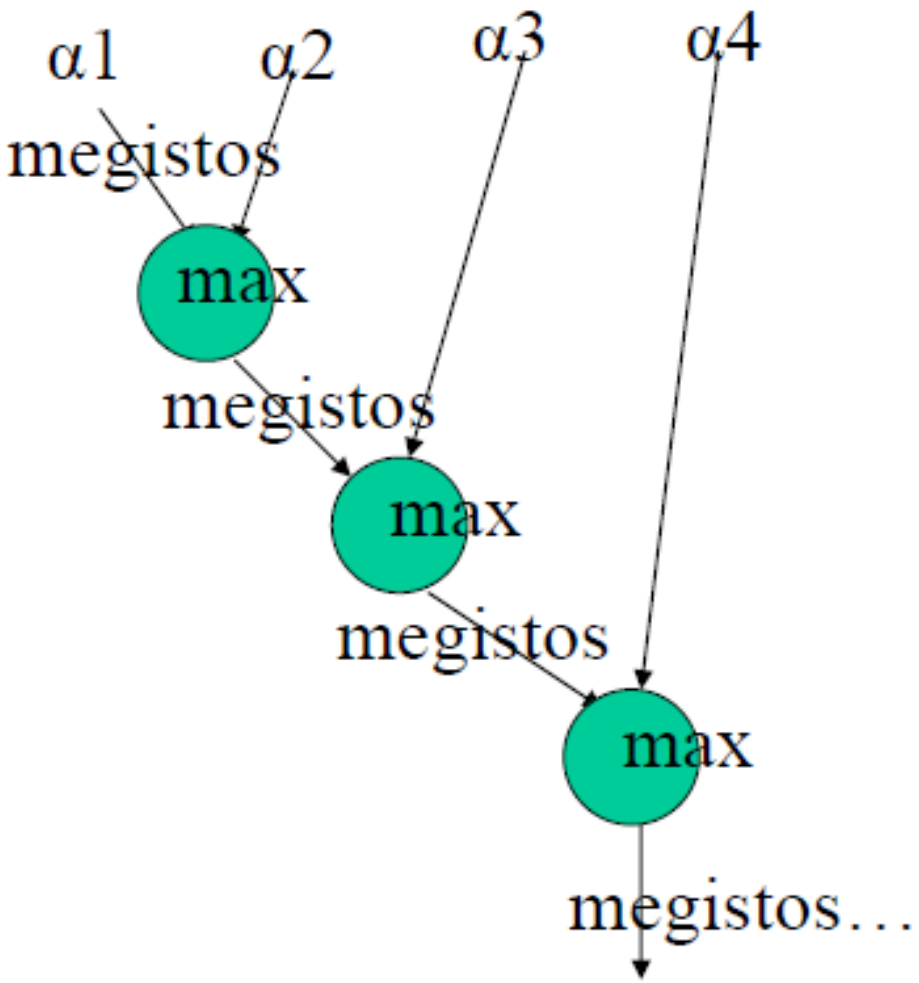
Χρήσιμες Λειτουργικότητες

- Πώς διαβάζουμε μια σειρά απροσδιόριστου μεγέθους που τερματίζεται με καθορισμένη τιμή;
- Πώς υπολογίζουμε τον μέγιστο αριθμό μιας σειράς;

Διάσπαση σε απλά υποπροβλημάτα:

υπολόγισε μέγιστο ανά δύο

Διάσπαση



Μέγιστη τιμή – Κώδικας

```
int number;       /* hold input number one at a time */
int maximum;      /* hold current maximum */

scanf("%d",&number);          /* diabase prwto stoixeio */
maximum = number;             /* arxikopoihsh */

while(number != 0){
     scanf("%d",&number);     /*diavase epomeno stoixeio*/
     if (number>maximum)      /* epeksergasia*/
           maximum = number;
}
printf("To megisto stoixeio tis seiras einai %d\n",
maximum);
```

while(συνθήκη)

Διάβασμα τιμής ενώσω ικανοποιείται η συνθήκη (αργότερα do-while)

```
int number;
printf("Enter a positive value: ");
scanf("%d",&number);          /* diabase timi*/
while(number<= 0){            /*sinthiki epanalipsis*/
    printf("Enter a possitive value: ");
    scanf("%d",&number);      /* diavase timi ksana */
}
```

Nested while (φωλιασμένα)

```
int i, j;
i=0;
while(i<5){
    j=0;
    while(j<5){
        printf("%d-%d", i, j);
        ++j;
    }
    printf("\n");
    ++i;
}
```

0-0 0-1 0-2 0-3 0-4

1-0 1-1 1-2 1-3 1-4

2-0 2-1 2-2 2-3 2-4

3-0 3-1 3-2 3-3 3-4

4-0 4-1 4-2 4-3 4-4

Nested while (φωλιασμένα)

```
int i, j;
i=0;
while(i<5){
    j=0;
    while(j<=i){
        printf("%d-%d", i, j);
        ++j;
    }
    printf("\n");
    ++i;
}
```

0-0

1-0 1-1

2-0 2-1 2-2

3-0 3-1 3-2 3-3

4-0 4-1 4-2 4-3 4-4

Πρόβλημα

Γράψτε κώδικα που υπολογίζει το παραγοντικό ενός ακέραιου αριθμού που δίνει ο χρήστης.

- Δεδομένα εισόδου: αριθμός x
- Δεδομένα εξόδου: το παραγοντικό: 1*2 … *(x-2)*(x-1)*x
- Πρόσθετα δεδομένα: μετρητής counter

Κώδικας 1

```
int i, f;
int n;
printf("please give the number for which you want the factorial\n");
scanf("%d",  &n);

f=1;
i=1;
while(i<=n){          /*  if  n=0  or  1  return  1*/
    f  =  f  *  i;
    ++i;
}
printf("The  factorial  of  %d  is  %d\n",  n,  f);
```

Μπορούσαμε να βάλουμε i=2;

Κώδικας 2α

```
int f;
int n,m;
printf("please give the number for which you want
the factorial\n");
scanf("%d",  &n);
m = n;   //In order to keep the initial value of given n
f=1;
while(n){
   f = f * n;
   n--;
}
printf("The  factorial  of  %d  is  %d\n",  m,  f);
```

Κώδικας 2β

```
int f;
int n,m;

printf("please give the number for which you want
the factorial\n");
scanf("%d",  &n);
m=n
f=n;
n--;
while(n){
    f = f * n;
    n--;
}
printf("The  factorial  of  %d  is  %d\n",  m,  f);
```

Identify these three steps in the pseudocode that follows: the initialization of the loop control variable, the loop repetition condition, and the update of the loop control variable.

```
a.  Get a value for n.
b.  Give p the value 1.
c.  while n is positive
        d.  Multiply p by n.
        e.  Subtract 1 from n.
f.  Print p with a label.
```

Σύνταξη for

- for(αρχικοποίηση; συνθήκη επανάληψης; ενημέρωση)
  εντολή;

ή

- for(αρχικοποίηση; συνθήκη επανάληψης; ενημέρωση){
  εντολή;
  εντολή;
  …
  }

Σημασία for

for (αρχικοποίηση; συνθήκη επανάληψης; ενημέρωση)
    εντολή;

Αρχικοποίηση
Συνθήκη
εντολή
Ενημέρωση
Συνθήκη
εντολή
Ενημέρωση
Συνθήκη

Παράδειγμα με for

```
int   x;
for   (x=0;   x<5;   ++x){
          printf("%d\n",x);
}
```

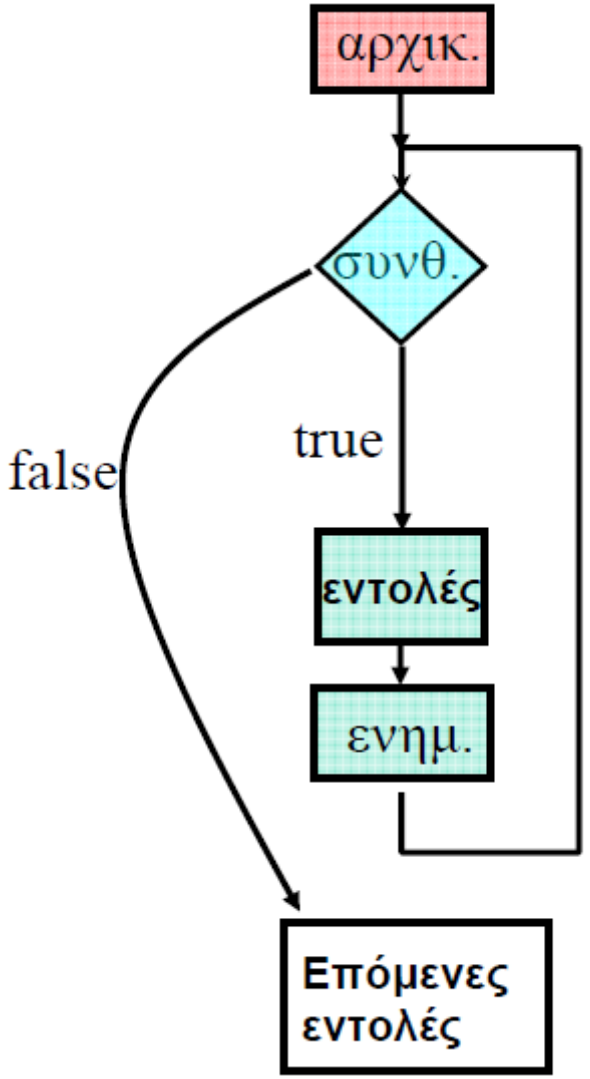
x	x<5	έξοδος
0	1 True	0
1	1	1
2	1	2
3	1	3
4	1	4
5	0 False	

Ροή Ελέγχου for

```
int  x,y;

for  (x=0;  x<5;  ++x){
        printf("%d\n",x);
}

y = x*x + 10;
printf("%d\n",y);
```



Ομοιότητα for με while

```
int x;
x=0;
while (x < 5) {
    printf("%d\n",x);
    ++x;
}
```

```
int x;
for (x=0; x<5;++x) {
    printf("%d\n",x);

}
```

Οποιοδήποτε for μπορεί να γραφεί με while και οποιοδήποτε while με for

Σύνταξη do-while

```
do
    εντολή;
while(συνθήκη);
```

```
do{
    εντολή;
    εντολή;
}while(συνθήκη);
```

Σημασία do-while

- Εκτέλεσε το σώμα της δομής do-while
- **Εφόσον η συνθήκη ισχύει** επανέλαβε την εκτέλεση των εντολών στο σώμα του βρόχου

- Στα for και while το σώμα μπορεί να μην εκτελεστεί εάν η συνθήκη δεν ικανοποιείται, ενώ στο do-while εκτελείται *τουλάχιστον μια φορά!*

Παράδειγμα

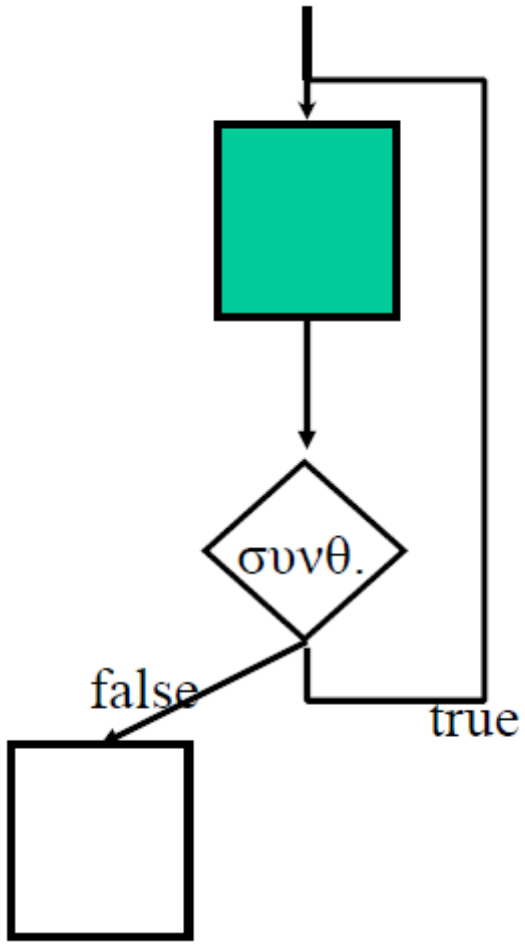
```
int a,b;
do{
  printf("Enter values for A and B where
   A < B: ");
  scanf("%d%d", &a, &b);
 } while (b <= a);
```

Παράδειγμα – Διαφορές

```
int number;
printf("Enter a possitive value: ");
scanf("%d",&number);
while(number<= 0){
    printf("Enter a possitive value: ");
    scanf("%d",&number);
}
```

```
int number;
do{
    printf("Enter a possitive value: ");
    scanf("%d",&number);
} while(number<= 0);
```

Ροή Ελέγχου do-while



Άσκηση προς εξάσκηση

Γράψτε κώδικα που δέχεται από τον χρήστη δύο ακέραιους αριθμούς και τον τελεστή μίας από τις 4 βασικές αριθμητικές πράξεις, και τυπώνει το αποτέλεσμα. Ο κώδικας να τερματίζει όταν και οι δύο αριθμοί είναι μηδέν.

Ανάλυση

- **Δεδομένα εισόδου:** ακέραιοι αριθμοί a, b, και ένας τελεστής (+, -, *, /) praxi
- **Δεδομένα εξόδου:** αποτέλεσμα result
- **Πρόσθετα δεδομένα:** στη διαίρεση πρέπει ο διαιρέτης να είναι διαφορετικός του μηδενός (*σωστός προγραμματισμός*)

Ατέρμονες Βρόχοι

- Θεληματικά
  - `while(1) {}`
  - `for (;;){}` (εδώ η συνθήκη είναι 1)
- Από λάθος
  - `while(x=1){}` ή `do{}while(x=1);` ή `for(;i=1;){}`
  - Μη ενημέρωση ή λανθασμένη ενημέρωση της μεταβλητής ελέγχου

```
int i = 0, n= 10;
while(i<n){
    printf("%d ", i);
}
```

Εντολή **break** (πρόωρη έξοδος)

- Η εντολή break μας **βγάζει από την επανάληψη**

```
while(count<n){
        scanf("%d", &number);
        if (number == -1)
            break;
        ++count;
}
printf("%d  %d\n",  count,  n);
```

Εντολή **continue**

Παράλειψη των υπολοίπων εντολών σε μια εντολή επανάληψης και εκ νέου εκτέλεση του σώματος εντολών

```
int count = 0, n_students = 27;
float number, sum = 0.0;
while(count < n_students){
      scanf("%d",&number);
      if (number < 0 || number > 100){
         printf("wrong entry\n");
         continue;
      }
      sum += number;
      ++count;
}
printf("The average is %f\n", sum/n_students);
```

getchar και putchar

- Συναρτήσεις εισόδου/εξόδου χαρακτήρων
- Διεπαφή
  - **int getchar()**, διάβασε τον επόμενο χαρακτήρα από την είσοδο, κίνησε δρομέα διαβάσματος στον επόμενο χαρακτήρα (διαβάζει χαρακτήρες μιας γραμμής μετά το enter)
  - **void putchar(int)**, τύπωσε χαρακτήρα στην μονάδα εξόδου

Παράδειγμα

- Γράψτε ένα πρόγραμμα που μετράει τον αριθμό χαρακτήρων σε μια απροσδιόριστου μεγέθους σειρά χαρακτήρων. Η σειρά εισάγεται από μονάδα εισόδου και τερματίζεται με την τιμή **EOF**
- **foo.txt::: hello world**
- **π.χ.** εισόδου/εξόδου: foo < foo.txt

  το μέγεθος της σειράς είναι 11

Χρήσιμες Λειτουργικότητες

- **Τι** πρέπει να γίνει;
  - διάβασμα μιας σειράς χαρακτήρων απροσδιορίστου μεγέθους
  - υπολογισμός μεγέθους σειράς
- Χρήσιμες Λειτουργικότητες
  - Πώς διαβάζουμε μια σειρά χαρακτήρων που τερματίζεται με καθορισμένη τιμή;
  - Πώς υπολογίζουμε το μέγεθος μιας σειράς;

Ανάγνωση σειράς απροσδιόριστου μεγέθους χαρακτήρων

- Όπως προηγουμένως:
  - Απαιτεί εντολή επανάληψης (βρόχος)
  - Ίδια δομή

```
/*διάβασε το πρώτο στοιχείο*/
while(/*το στοιχείο δεν σημαδοτεί τέλος*/){

      /* διάβασε επόμενο στοιχείο */
}
```

Διάβασμα σειράς χαρακτήρων

```
int c;
c = getchar();    /* diabase prwto xaraktira */

while(c != EOF){ /* oxi telos tou file */
        c = getchar();/*diavase epomeno xaraktira*/
}
```

Τύπωση σειράς χαρακτήρων

```
int c;
c = getchar();              /* diabase prwto xaraktira */
while(c != EOF){
        putchar(c);         /* typwse xaraktira */
        c = getchar();      /*diavase epomeno xaraktira*/
}
```

Μέγεθος σειράς χαρακτήρων

```
int c;
int size;
size = 0;                    /* arxikopoihsh */
c = getchar();              /* diabase prwto xaraktira */
while(c != EOF){

        size = size + 1; /* metra akomi ena xaraktira */
        c = getchar();   /* diavase epomeno xaraktira */
}
```

Απλοποίηση;

```
int c;
int size;
size = 0;                    /* arxikopoihsh */
while((c = getchar() )!= EOF){/*diabase kai elegxe xaraktira*/
size = size + 1;             /* metra akomi ena xaraktira */
}
```

Απαρίθμηση συγκεκριμένου γεγονότος

- Πόσες φορές παρουσιάστηκε ο χαρακτήρας Α;
- **Τι** πρέπει να γίνει;
  - διάβασμα μιας σειράς χαρακτήρων απροσδιόριστου μεγέθους
  - έλεγχος για χαρακτήρα Α
    - αύξησε μετρητή κάθε φορά που διαβάζεις τον χαρακτήρα Α

Κώδικας

```
int c;
int count;
count = 0; /* arxikopoihsh */
while((c = getchar() )!= EOF){/* diabase xaraktira */
      if (c=='A')
            ++count;   /* metra akomi ena xaraktira */
}
```

Απαρίθμηση γραμμών

- Πόσες γραμμές υπάρχουν στα δεδομένα;
- **Τι** πρέπει να γίνει;
  - διάβασμα μιας σειράς χαρακτήρων απροσδιόριστου μεγέθους
  - έλεγχος για χαρακτήρα “επόμενης γραμμής”
    - αύξησε μετρητή αν χαρακτήρας είναι το \n

Κώδικας

```
int c;
int count;
count = 0;                                  /* arxikopoihsh */
while((c = getchar() )!= EOF){/* diabase xaraktira */
      if (c=='\n')      /* elegxe gia epomeni grammi */
    ++count;                  /* metra akomi mia grammi */
}
```

Αριθμός Λέξεων

- Γράψτε ένα πρόγραμμα που διαβάζει ένα κείμενο από την είσοδο και υπολογίζει και τυπώνει τον αριθμό λέξεων.

Χρήσιμες Λειτουργικότητες

- Τι πρέπει να γίνει;
  - Διάβασμα ακολουθίας χαρακτήρων
  - **Απαρίθμηση γεγονότος:** Λέξη
    - Τι είναι λέξη: συνεχόμενη σειρά χαρακτήρων
    - Τα άλλα τι είναι; `'\n', '\t', ' '`
    - Μετασχηματισμός σε διάστημα ή χαρακτήρα
      - **Μετρούμε:** μετακίνηση από άσπρο διάστημα σε λέξη (ή από λέξη σε άσπρο διάστημα)
    - Πού είμαστε προηγουμένως/τώρα: διάστημα ή λέξη-έννοια σημαίας (flag)

Δομή / Αλγόριθμος

```
int  main(){
/* χρήση δύο σημαίων -πριν, τώρα (διάστημα ή λέξη)*/
/* πριν = διάστημα --- αρχικοποίηση*/

/* διάβασε κάθε χαρακτήρα από είσοδο μέχρι EOF
κάθε φορά επανέλαβε τα πιο κάτω*/

/* που είμαστε τώρα (λέξη ή διάστημα) */
/* εαν πριν σε διάστημα και τώρα σε λέξη τότε μέτρα
λέξη*/

/*πριν παίρνει τιμή του τώρα */

return 0;
}
```

Κώδικας

```
int  main(){
int  c,  nw  =  0;
int before, now; /* simaies */
before = WHITE_SPACE; /* prepei na dilothei san stathera */

while((c=getchar())  !=  EOF){
      if (c == ‘ ‘ || c == ‘\t’ || c == ‘\n’)
            now = WHITE_SPACE;
      else
            now  =  NON_WHITE_SPACE; /* stathera */
      if ( before == WHITE_SPACE && now == NON_WHITE_SPACE)
            nw = nw + 1;
      before  =  now;
}
printf(“Number of words in input is %3d.\n”,nw);
return 0;
}
```

Κλασσικά λάθη σε βρόχους

- Σύγχυση ανάμεσα στις εντολές if και while:
  - `if (συνθήκη) εντολή`
  - `while (συνθήκη) εντολή`
- while, do-while: παράλειψη παρενθέσεων γύρω από τις συνθήκες
- for: παράλειψη του χαρακτήρα ; ή χρήση , αντί ;
- Σώμα δομής:

```
while(x<0)
        count +=x;
         ++x;
printf("%d\n",x);
```

Κλασσικά λάθη σε τελεστές

- Σύνθετοι τελεστές: a*=b+c;

  σημαίνει a = a*(b + c); όχι a = a * b + c;
- Χρήση ++, −−, και σύνθετης ανάθεσης (+= κτλ) σε σύνθετες εκφράσεις
- Χρήση του τελεστή != με τιμές τύπου float και double πρέπει να αποφεύγεται, π.χ.
  - while (balance != 0.0){}

Περίληψη

- Δομές Επανάληψης
  - while(){}
  - for(){}
  - do{ }while()
  - Άπειροι βρόχοι
  - Εντολές break και continue
- Τελεστές postfix/prefix
- Σύνθετοι τελεστές
- Συναρτήσεις getchar/putchar
- Εφαρμογές βρόχων για διαχείριση σειρών χαρακτήρων