ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

ΑΡΧΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κεφάλαιο 3

Επιμέλεια:
Βασίλης Παλιουράς , Αναπληρωτής Καθηγητής
Ευάγγελος Δερματάς , Αναπληρωτής Καθηγητής
Σταύρος Νούσιας , Βοηθός Ερευνητή
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τυπου άδειας χρησης, η άδεια χρησης αναφερεται ρητώς.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου των διδασκόντων καθηγητών.
- Το έργο «Ανοικτά Ακαδήμαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών» έχει χρηματοδοτηθεί μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκη Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικους πόρους.

Ανάπτυξη

Το παρόν εκπαιδευτικό υλικό αναπτύχθηκε στο τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Μέχρι τώρα...

- Οργάνωση Προγράμματος C
- Ενέργειες -> ρήματα (συντακτικό) -> συναρτήσεις
- Δεδομένα -> αντικείμενα(συντακτικό) -> μεταβλητές
- Αφαιρετικότητα (abstraction)
- εργαλείο για την αντιμετώπιση της πολυπλοκότητας
- Διαχωρίζουμε το "Τι γίνεται" => όνομα, δήλωση συνάρτησης
- από το "Πώς γίνεται"
- Χρήση σύνθετων τύπων
- Αυξητική Ανάπτυξη Προγράμματος
- Top-down ανάπτυξη
- Διευκολύνει ανάπτυξη του προγράμματος
- Διευκολύνει τον έλεγχο του προγράμματος
- Ξεκινάμε γράφοντας κάτι εκτελέσιμο.
- Διαδικασία compilation και link για δημιουργία εκτελέσιμου.
- Γίνονται από την εφαρμογή gcc/mingw

Μερικές καλές πρακτικές

- Πρώτα σκέφτομαι μετά γράφω κώδικα.
- Γράφω με μέθοδο, ώστε να είναι αμέσως εκτελέσιμος ο κώδικας.
- Γράφω τμήματα κώδικα και τα δοκιμάζω.
- Όταν σχεδιάζω την υλοποίηση, θα πρέπει να με απασχολεί πώς θα κάνω τις δοκιμές.

Έκδοση 0: θα πρέπει να είναι εκτελέσιμο!

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power =
   ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 printf ("function: diabase\n");
 return 5;
}

int ypologisepower(int x ) {
 printf ("function:
   ypologise\n");
 return x;
}

void typwse (int x ) {
  printf("function: typwse\n");
}
```

Έκδοση 1: πλήρης typwse()

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power =
   ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 printf ("function: diabase\n");
 return 5;
}

int ypologisepower(int x ) {
 printf ("function:
   ypologise\n");
 return x;
}

void typwse (int x ) {
 printf("function: typwse\n");
 printf("apotelesma: %d\n",x);
}
```

Έκδοση 2: πλήρης ypologisepower()

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power =
   ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 printf ("function: diabase\n");
 return 5;
}

int ypologisepower(int x ) {
 printf ("function:
   ypologise\n");
 return x * x * x;
}

void typwse (int x ) {
 printf("function: typwse\n");
 printf("apotelesma: %d\n",x);
}
```

Έκδοση 3: πλήρης diabase()

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power =
   ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 int aninput;
 printf ("function: diabase\n");
 scanf("%d", &aninput);
 return aninput;
}

int ypologisepower(int x ) {
 printf ("function:
   ypologise\n");
 return x * x * x;
}
void typwse (int x ) {
 printf("function: typwse\n");
 printf("apotelesma: %d\n",x);
}
```

Στοιχεία της Γλώσσας C

- Γραμματική και Συντακτικό
- Διαθέσιμοι **τύποι δεδομένων**
  - Απλοί και σύνθετοι τύποι
- Βασική βιβλιοθήκη της C
  - παρέχει ένα σύνολο έτοιμων συναρτήσεων: printf( ), scanf( ), …
- Εκτεταμένη τεκμηρίωση της GNU C library
  - http://www.gnu.org/software/libc/manual/

Δεσμευμένες λέξεις (reserved words)

- Λέξεις κλειδιά (keywords)
- Ονόματα συναρτήσεων της βασικής βιβλιοθήκης
- Ονόματα μακροεντολών που ορίζονται σε αρχεία επικεφαλίδας: EOF, …
- Ονόματα τύπων που ορίζει η βασική βιβλιοθήκη: time_t, …
- Ονόματα εντολών προεπεξεργαστή: include, define
- Ονόματα της μορφής _DATE_, _FILE_, κτλ.

Αναγνωριστές (Identifiers)

- λέξεις που κατασκευάζει ο προγραμματιστής για να ονομάσει
  - μεταβλητές
  - σταθερές
  - συναρτήσεις
  - ...
- Δεν θα πρέπει να είναι **δεσμευμένες**

Τύποι Δεδομένων στη C

- **`char`** – χαρακτήρας
- **`int`** – ακέραιος
- **`float`** – αριθμός κινητής υποδιαστολής απλής ακρίβειας
- **`double`** – αριθμός κινητής υποδιαστολής διπλής ακρίβειας
- απαριθμητικός τύπος
  - `enum boolean {FALSE, TRUE};`
- σύνθετοι τύποι
  - πίνακες και δομές (**`struct`**)

- Τύποι του C90. Το C99 επεκτείνει.

Παραδείγματα Αναγνωριστών - ποιοί είναι σωστοί;

- j
- 5j
- lname
- _Fname
- $amount
- set_password
- int
- MaXveLocity
- find_max#of_lements
- get_word
- isdigit
- get@name

Μετατροπή Fahrenheit σε Celcius

```
#include <stdio.h>
/* print Fahrenheit-Celcius table for
fahr = 0, 20, …, 300 */

main ( ) {
   int fahr, celcius;
   int lower, upper, step;

   lower = 0 ; /* lower limit of temperature */
   upper = 300;
   step = 20 ;
   fahr = lower ;
   while (fahr <= upper) {
      celcius = 5* (fahr – 32) / 9;
      printf (“%d\t%d\n”, fahr, celcius);
      fahr = fahr + step;
    }
}
```

Μετατροπή Fahrenheit σε Celcius

```
#include <stdio.h>
/* print Fahrenheit-Celcius table for fahr = 0, 20, …, 300 */
main () {
int fahr, celcius;
int lower, upper, step;
lower = 0 ; /* lower limit of temperature */
upper = 300;
step = 20 ;
fahr = lower ;
while (fahr <= upper) {
     celcius = 5* (fahr – 32) / 9;
     printf (“%d\t%d\n”, fahr, celcius);
     fahr = fahr + step;
     }
}
```

δήλωση μεταβλητών

Μετατροπή Fahrenheit σε Celcius(1)

```
#include <stdio.h>
/* print Fahrenheit-Celcius table for fahr = 0, 20, …, 300 */

main () {

int fahr, celcius;
int lower = 0, upper = 300, step = 20;

fahr = lower ;

while (fahr <= upper) {
     celcius = 5* (fahr – 32) / 9;
     printf (“%d\t%d\n”, fahr, celcius);
     fahr = fahr + step;
     }
}
```

δήλωση μεταβλητών

Μαζί με αρχικοποίηση

Μετατροπή Fahrenheit σε Celcius(2)

```
#include <stdio.h>
/* print Fahrenheit-Celcius table for fahr = 0, 20, …, 300 */

main () {

int lower = 0, upper = 300, step = 20;
int fahr = lower, celcius;

while (fahr <= upper) {
    celcius = 5* (fahr - 32) / 9;
    printf ("%d\t%d\n", fahr, celcius);
    fahr = fahr + step;
    }

}
```

Χρήση άλλης μεταβλητής

Για αρχικοποίηση

Ομαδοποίηση Τελεστών

Κατηγορία	Ενδεικτικά C	Ενδεικτικά FORTRAN	Ενδεικτικά Python
Αριθμητικοί	* / % + -	* / + -	Όπως C
Λογικοί	&& || !	.AN D. .OR. .NOT.	and, or, not
Συσχετιστικοί	> >= == !=	.G T. .G E. .EQ. .NE.	Όπως C (το διάφορο και <>)
Διαχείρισης δυαδικών ψηφίων μιας λέξης	>> & | ^	έκδοση	Όπως C
Τελεστές διαχείρισης μνήμης	& [ ] . ->		

Εκφράσεις και προτάσεις

```
a  = f(g[3]) + 3*d
```

```
sum = sum + total
```

έκφραση
(expression)

πρόταση
(statement)

```
...
a  = f(g[3]) + 3*d ;
...
```

Προτάσεις και σύνθετες προτάσεις

πρόταση1;

{ πρόταση1;
πρόταση2;
πρόταση3; }

Σύνθετη πρόταση:

Μπλοκ προτάσεων π
ου ορίζεται με άγκισ
τρα.

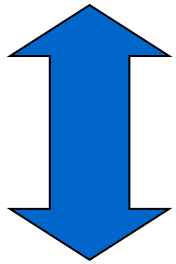
Προτάσεις και σύνθετες προτάσεις

- ; -->Κενή πρόταση:
- Δεν είναι συντακτικό λάθος.
- Δεν κάνει κάτι.
- Εξηγεί συμπεριφορές.
- 13;   Δεν είναι συντακτικό λάθος.

Τελεστής Ανάθεσης =

```
int a, b;
a = 5 ;
b = a ;
```

```
int a, b;
b = a = 5;
```



όλη η έκφραση έχει ως α
ξία την τιμή 5

σημαίνει

```
b = (a = 5);
```

Σχεσιακοί Τελεστές

- Συγκρίσεις
  > , >= , <, <=

  Παράδειγμα

  a < 5

  – αν το a είναι μικρότερο του **πέντε** η έκφραση είναι **αληθής** διαφορετικά είναι **ψευδής**

Έλεγχος Ισότητας

- a == 5 ; /* αληθές αν το a είναι 5 */
- a != 5 ; /* αληθές αν το a δεν είναι 5 */
- άλλος ο ρόλος του = άλλος του ==

if (/*…*/) /*…*/ else /*…*/ ;

if (έκφραση)
 (σύνθετη) εντολή 1;
else
 (σύνθετη) εντολή 2;

```
if ( a == 5)
   printf ("a equals five.\n");
else
   printf("a does not equal five\n");
```

Παράδειγμα 1: if με απλή πρόταση

```
#include <stdio.h>

int main() {
   int a = 3;
   printf("a:%d\n", a);
   if (a==5)
      printf("is five\n");

   printf("a:%d\n",a);
   return 0;
}
```

Παράδειγμα 2: if με σύνθετη πρόταση

```
#include <stdio.h>

int main() {
   int a = 3;
   printf("a:%d\n", a);
   if (a==5)
   {
      printf("is five\n");
      printf("nothing else\n");
       }
   printf("a:%d\n",a);
   return 0;
}
```

(αντί)-παράδειγμα 1: Τι θα τυπώσει; Γιατί;

```c
#include <stdio.h>

int main() {
   int a = 3;
   printf("a:%d\n", a);
   if (a==5)
      printf("is ");
      printf("five\n");

   printf("a:%d\n",a);
   return 0;
}
```

(αντί)-παράδειγμα 2:Τι θα τυπώσει; Γιατί;

```c
#include <stdio.h>

int main() {
  int a = 3;
  printf("a:%d\n", a);
  if (a=5)
     printf("is five\n");

  printf("a:%d\n",a);
  return 0;
}
```

(αντί)-παράδειγμα 3: Τι θα τυπώσει; Γιατί;

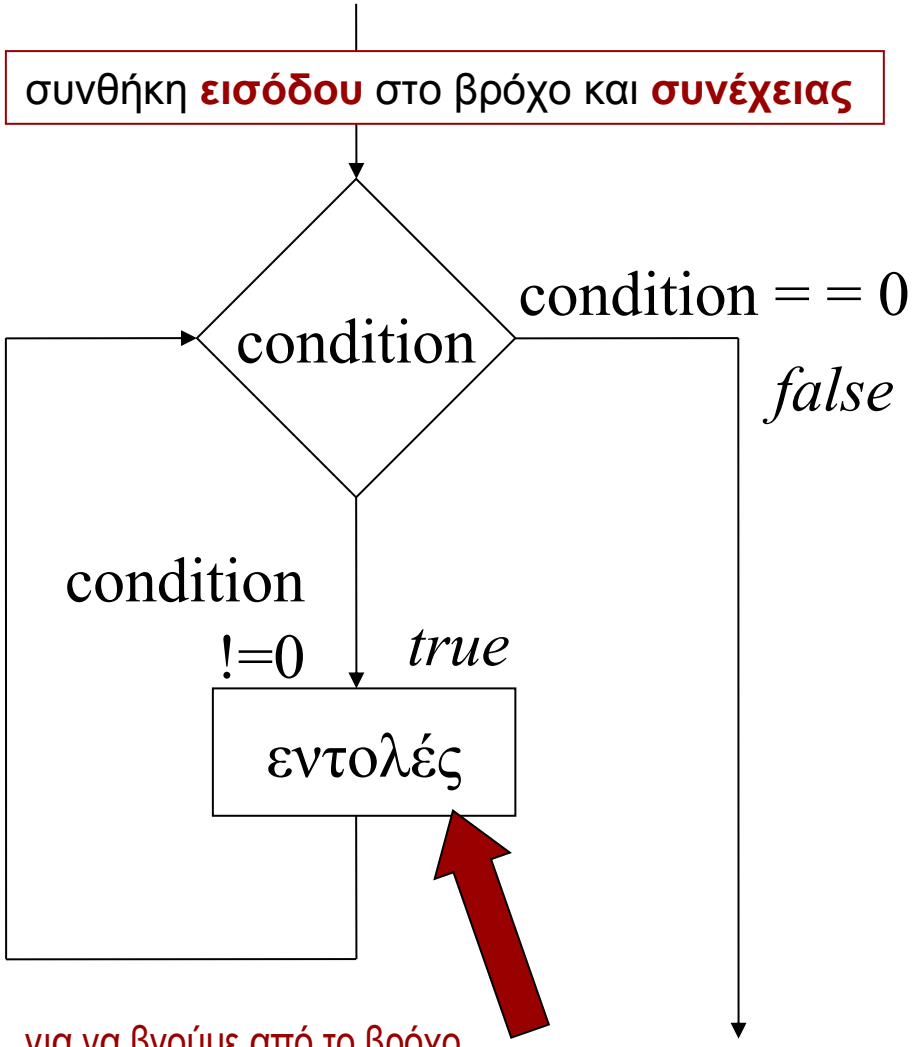
```c
#include <stdio.h>

int main() {
  int a = 3;
  printf("a:%d\n", a);
  if (a==5) ;
     printf("is five\n");

  printf("a:%d\n",a);
  return 0;
}
```

Βρόχος while

```
main ( ) {
   int condition;
   condition = 1;
   while (condition) {
        printf("loop body");
        condition = f();
   }
}
```



Βρόχος while

while (έκφραση)
σύνθετη (ή όχι) πρόταση

```
a = 0 ;
while (a < 5) {
    printf (“value of a is %d\n”, a);
    a ++;
    }
```

Βρόχος while και βρόχος for

```
a = 0 ;
while (a < 5) {
    printf ("value of a is %d\n", a);
    a ++;
    }
```

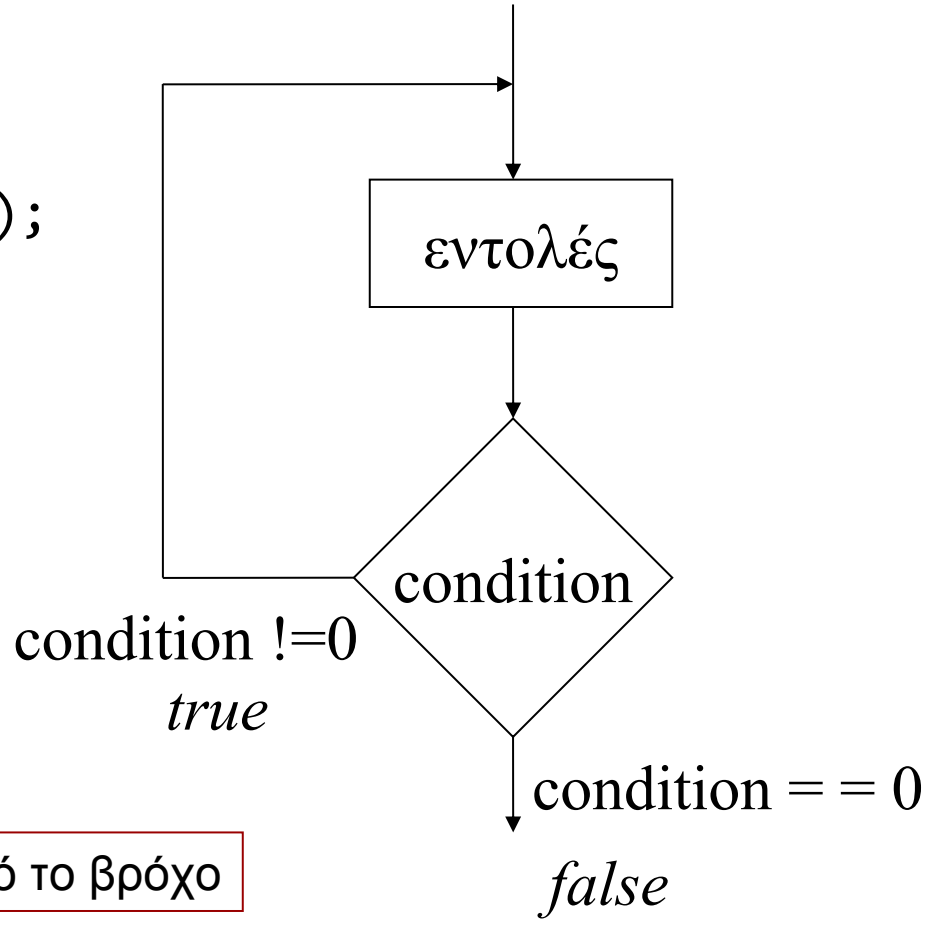
```
for (a = 0; a < 5; a ++) {
    printf ("value of a is %d\n", a);
    }
```

Στη C ο βρόχος **for** ορίζεται ως άλλη γραφή του **while**

```
for (a = 0 ; a < 5; a++)
printf ("value of a is
%d\n", a);
```

Βρόχος do /*… */ while (/*…*/);

```
main ( ) {
   int condition;
   do {
        printf("loop body");
        condition = f();
   } while (condition) ;
}
```



συνθήκη **εξόδου** από το βρόχο

Παραδείγματα χρήσης δομών ελέγχου

- Σταθερός αριθμός επαναλήψεων
- Αριθμός επαναλήψεων εξαρτώμενος από τα δεδομένα
- Ένθετες (nested) δομές ελέγχου

Παράδειγμα 1
Καθορισμένος αριθμός επαναλήψεων

- Να γραφεί ένα πρόγραμμα που διαβάζει δέκα ακεραίους, έναν κάθε φορά και τυπώνει το μερικό άθροισμα.

- Στο τέλος τυπώνεται το συνολικό άθροισμα και το γινόμενό τους.

- (Εδώ λύση χωρίς πίνακες).

Λεκτική περιγραφή

- Διάβασε έναν αριθμό
- Υπολόγισε το μερικό άθροισμα
- Τύπωσε το μερικό άθροισμα
- Έχεις διαβάσει δέκα αριθμούς;
  - Αν όχι, επανάλαβε.

Λεκτική περιγραφή

- Διάβασε έναν αριθμό num
- Υπολόγισε το μερικό άθροισμα sum
- Τύπωσε το μερικό άθροισμα sum
- Έχεις διαβάσει δέκα αριθμούς;
  - Αν όχι, επανάλαβε.

```
int num, sum;
```

Λεκτική περιγραφή

- Διάβασε το num
- Υπολόγισε το sum
- Τύπωσε το sum
- Έχεις διαβάσει δέκα αριθμούς;
- Αν όχι, επανάλαβε.

Λεκτική περιγραφή

- Επανάλαβε για δέκα φορές {
- Διάβασε το num → scanf()
- Υπολόγισε το sum → computeSum() ή sum = sum + num
- Τύπωσε το sum → printf()
- }

Υλοποίηση

```c
#include <stdio.h>

main() {

  int i, num, sum=0;

  for (i=0; i<10; i++) {
   scanf("%d", &num);
   sum = sum + num;
   printf("partial sum: %d\n", sum);
   }

  printf("total: %d", sum);

}
```

```c
#include <stdio.h>
#define N 10

main() {

  int i, num, sum=0;

  for (i=0; i<N; i++) {
   scanf("%d", &num);
   sum = sum + num;
   printf("partial sum: %d\n", sum);
   }

  printf("total: %d", sum);

}
```

Παράδειγμα

- Να γραφεί ένα πρόγραμμα που διαβάζει ακεραίους, έναν κάθε φορά και τυπώνει το μ ερικό άθροισμα και το μερικό γινόμενο, όσο ο χρήστης δίνει ως είσοδο αριθμούς > 0.
- Αριθμοί ≤ 0 δεν λαμβάνονται υπόψη στους υπολογισμούς.
- Στο τέλος τυπώνεται το συνολικό άθροισμα και το γινόμενό τους.

Έκδοση 1

```
#include <stdio.h>
int main( )
{

  int input, sum = 0 , prod = 1;

  scanf("%d", &input);

  while (input>0) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input ;
        scanf("%d", &input);
   }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

Έκδοση 2

```
#include <stdio.h>

int main( )
{

  int input, sum = 0 , prod = 1;

  for (scanf("%d", &input); input>0; scanf("%d", &input) ) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input ;
      }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

Έκδοση 3

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

int main()
{

  int input, sum = 0 , prod = 1;

  do {
     scanf("%d", &input);
     if (input >0 ) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input ;
        }
     } while (input>0) ;

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

Έκδοση 4

```c
#include <stdio.h>

int main()
{

  int input, sum = 0 , prod = 1;

  do {
     scanf("%d", &input);
     if (input <=0 )
        break;
     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     } while (input>0) ;

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

return 0;
}
```

Έκδοση 4a

```
#include <stdio.h>

int main()
{

  int input, sum = 0 , prod = 1;

  do {
     scanf("%d", &input);
     if (input <=0 )
        break;
     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     } while (input>0) ;

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

return 0;
}
```

Έξοδος από βρόχο
Ούτως ή άλλως
μόνο με **break**
=>

Απλή συνθήκη στο **while**

Έκδοση 5

```c
#include <stdio.h>

int main( )
{

  int input=1, sum = 0 , prod = 1;

  while ( input > 0) {
     scanf("%d", &input) ;


     if (input <=0 )
        break;

     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

Αρχικοποίηση του `input` για εξασφάλιση εισόδου στο βρόχο `while`

Έκδοση 6

```c
#include <stdio.h>
int getinput(void);
int partialsums(int);
void report(int);

int main( )
{
  int input, sum ;

  sum = partialsums(1);
  report(sum);

  return 0;
}

int getinput(void) {
    int a;
    scanf("%d", &a);
    return a;
}
```

```c
void report(int sum) {
  printf("sum: %d\n", sum);
}

int partialsums(int input) {
  int sum = 0 ;
  while ((input=getinput())>0) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);

   }
  return sum;
}
```

Έκδοση 6a

(απλούστερη διεπαφή της partialsums)

```c
#include <stdio.h>
int getinput(void);
int partialsums(void);
void report(int);

int main( )
{
  int input, sum ;

  sum = partialsums();
  report(sum);

  return 0;
}

int getinput(void) {
    int a;
    scanf("%d", &a);
    urn a;
```

```c
void report(int sum) {
  printf("sum: %d\n", sum);
}

int partialsums(void) {
  int input ;
  int sum = 0 ;
  while ((input=getinput())>0) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);

   }
  return sum;
}
```

Σημείωμα αναφοράς

- Copyright Πανεπιστήμιο Πατρών, Παλιουράς Βασίλειος , Δερματάς Ευάγγελος «Αρχές Προγραμματισμού ». Έκδοση: 1.0. Πάτρα 2015
- Διαθέσιμο από τη δικτυακιακή διέυθυνση https://eclass.upatras.gr/modules/