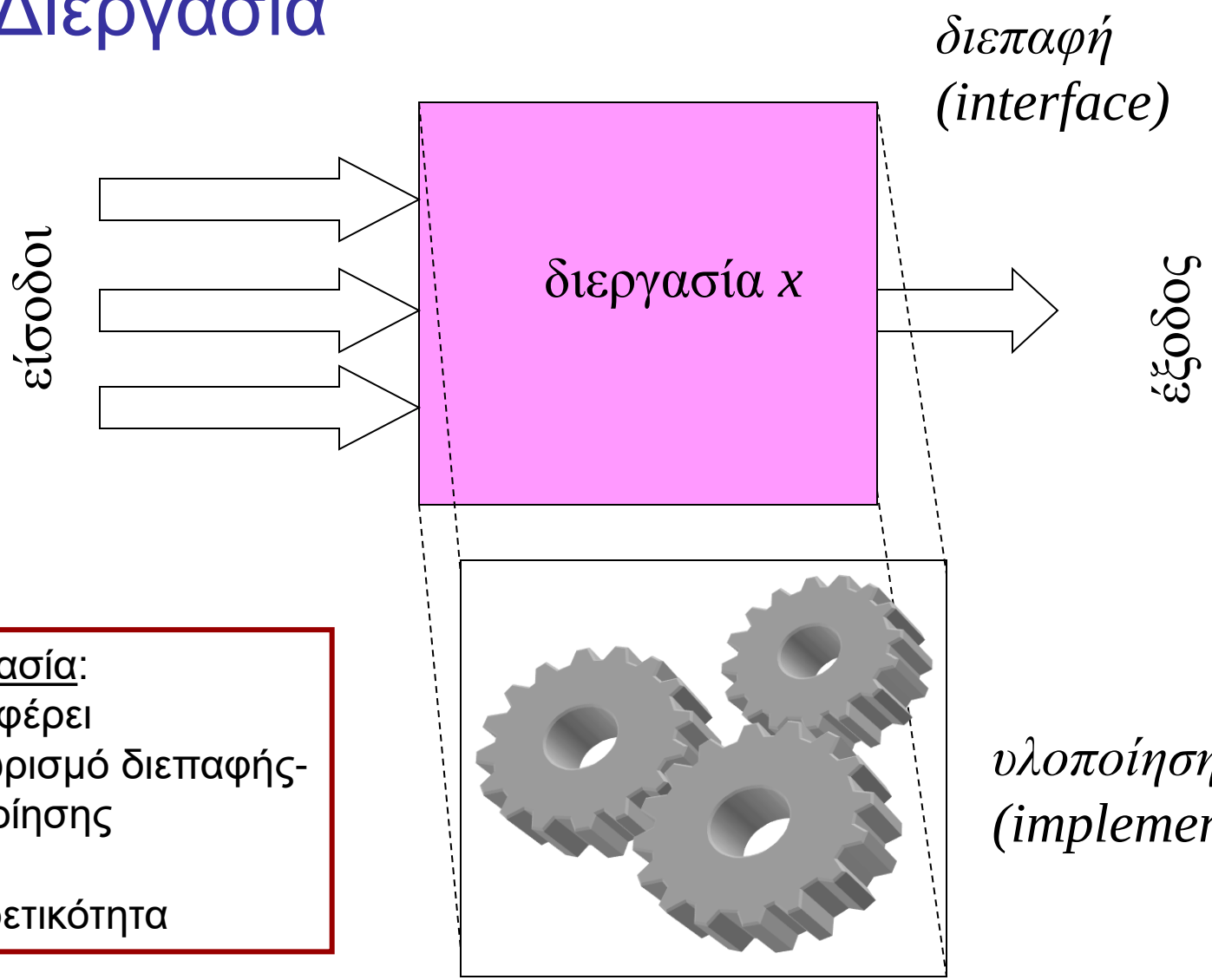
Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς
paliuras@ece.upatras.gr

Διαδικαστικός προγραμματισμός: Η έννοια της διεργασίας

- top down και bottom up
- Τμήμα μιας ευρύτερης λύσης, με σαφώς καθορισμένη σχέση εισόδων εξόδων
  - όταν είναι δυνατόν να εκτελεστεί από υπολογιστή $\Rightarrow$ *υπολογιστική διεργασία*
- Βασικό εργαλείο στο χειρισμό της πολυπλοκότητας $\Rightarrow$ αφαιρετικότητα

Διεργασία



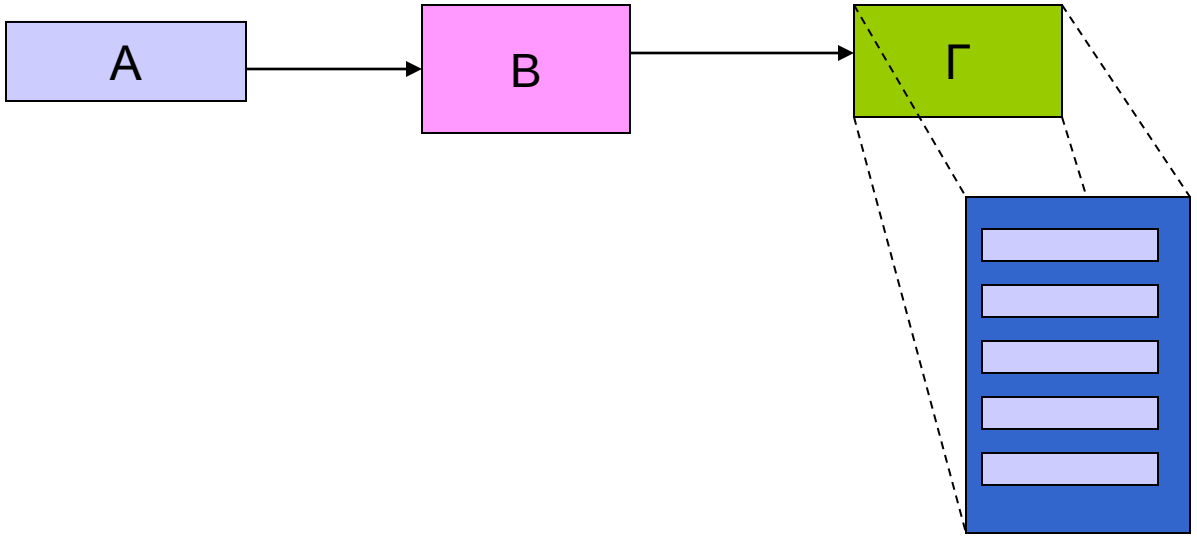
Διεργασία:
προσφέρει διαχωρισμό διεπαφής-υλοποίησης
⇒
αφαιρετικότητα

Συναρτήσεις και Διεργασίες

- Στη γλώσσα C

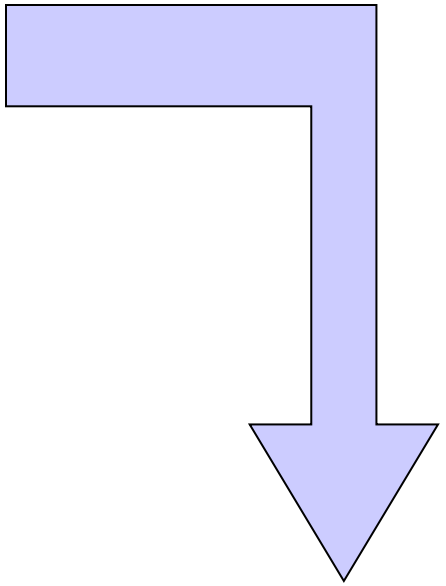
  συναρτήσεις $\leftrightarrow$ υπολογιστικές διεργασίες
  - χρήση της printf στο παράδειγμα

Οργάνωση λύσης



Απεικόνιση σε οδηγίες προς τον υπολογιστη: «προστακτικός» προγραμματισμός

- Εκτέλεσε τη διεργασία Α
- Εκτέλεσε τη διεργασία Β
- Εκτέλεσε τη διεργασία Γ
- Εκτέλεσε τη διεργασία Δ



Πώς γνωρίζει το σύστημα τι πρέπει
να κάνει για να εκτελέσει τη διεργασία Α;

Διεργασία Α:
Εκτέλεσε τη διεργασία Α1
Εκτέλεσε τη διεργασία Α2

Βασική μέθοδος: Αυξητική ανάπτυξη προγράμματος (incremental development)

- Να γραφεί ένα πρόγραμμα που **διαβάζει** έναν αριθμό, να **υπολογίζει** την τρίτη δύναμή του, και στη συνέχεια να **τυπώνει** το αποτέλεσμα.

int number, power;

- Διάβασε έναν αριθμό ⟵ number
- Υπολόγισε την τρίτη δύναμή του ⟵ power
- Τύπωσε το αποτέλεσμα ⟵ power

- **int** number, power;
- Διάβασε number ← readnumber()
- Υπολόγισε power ← computepower()
- Τύπωσε power ← printout()

Σχεδίαση top-down

```
main( ) {
  int number, power;
  number = readnumber( );
  power = computepower(number );
  printout(power );

}
```

έκδοση 0: θα πρέπει να είναι εκτελέσιμο!

```c
#include <stdio.h>
int readnumber(void);
int computepower(int);
void printout(int);

int main( ) {
 int number, power;
 number = readnumber( );
 power = computepower(number);
 printout(power);

 return 0;
}
```

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

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

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

έκδοση 1: πλήρης printout()

```c
#include <stdio.h>
int readnumber(void);
int computepower(int);
void printout(int);

int main( ) {
 int number, power;
 number = readnumber( );
 power = computerpower(number);
 printout(power);

 return 0;
}
```

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

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

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

 return ;
}
```

έκδοση 2: πλήρης computepower()

```c
#include <stdio.h>
int readnumber(void);
int computepower(int);
void printout(int);

int main( ) {
 int number, power;
 number = readnumber( );
 power = computepower(number);
 printout(power);
 return 0;
}
```

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

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

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

Έκδοση 3: πλήρης printout()

```c
#include <stdio.h>
int readnumber(void);
int computepower(int);
void printout(int);

int main( ) {
 int number, power;
 number = readnumber( );
 power = computepower(number);
 printout(power);

 return 0;
}
```

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

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

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

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 -> υλοποίηση συνάρτησης,***
  - Χρήση σύνθετων τύπων
- Αυξητική Ανάπτυξη Προγράμματος
  - top-down ανάπτυξη
  - Διευκολύνει *ανάπτυξη* του προγράμματος
  - Διευκολύνει τον *έλεγχο* του προγράμματος
  - Ξεκινάμε γράφοντας κάτι εκτελέσιμο
  - C: για να είναι εκτελέσιμο, πρέπει να έχει main() διαφορά από Pytthon.
- Διαδικασία compile και link για δημιουργία εκτελέσιμου.
  - Γίνονται με gcc/mingw

Μερικές καλές πρακτικές που είδαμε μέχρι τώρα

- 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.

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

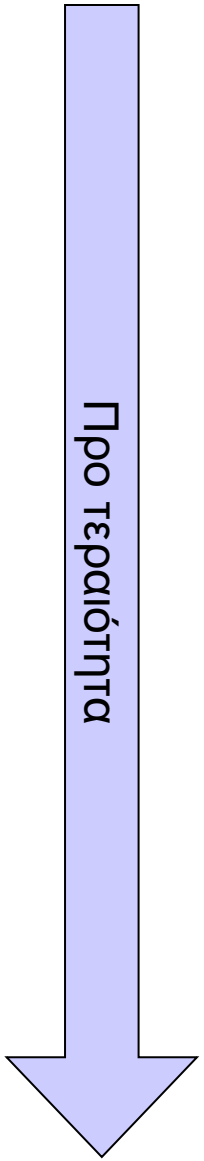
- **`char`** – χαρακτήρας
- **`int`** – ακέραιος
- **`float`** – αριθμός κινητής υποδιαστολής απλής ακρίβειας
- **`double`** – αριθμός κινητής υποδιαστολής διπλής ακρίβειας
- απαριθμητικός τύπος
  - `enum boolean {FALSE, TRUE};`
- σύνθετοι τύποι
  - πίνακες και δομές (**`struct`**)
- Τύποι του C90. Το C99 επεκτείνει με header files
  - `#include <stdbool.h>`
  - `stdint.h`
  - `inttypes.h`

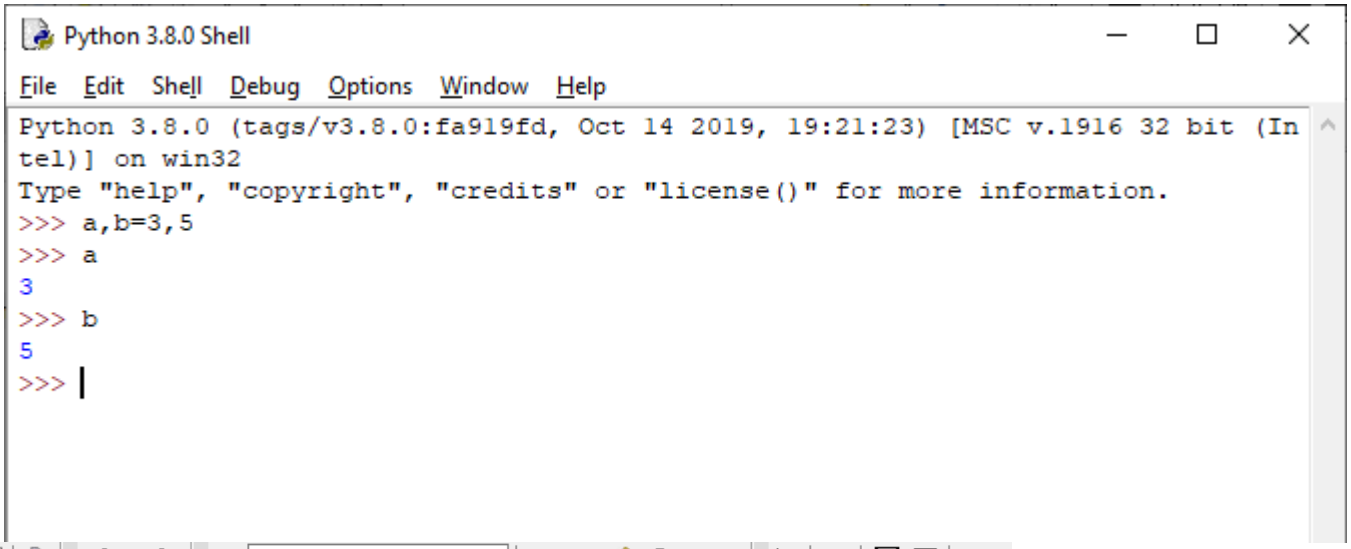
Ομαδοποίηση Τελεστών

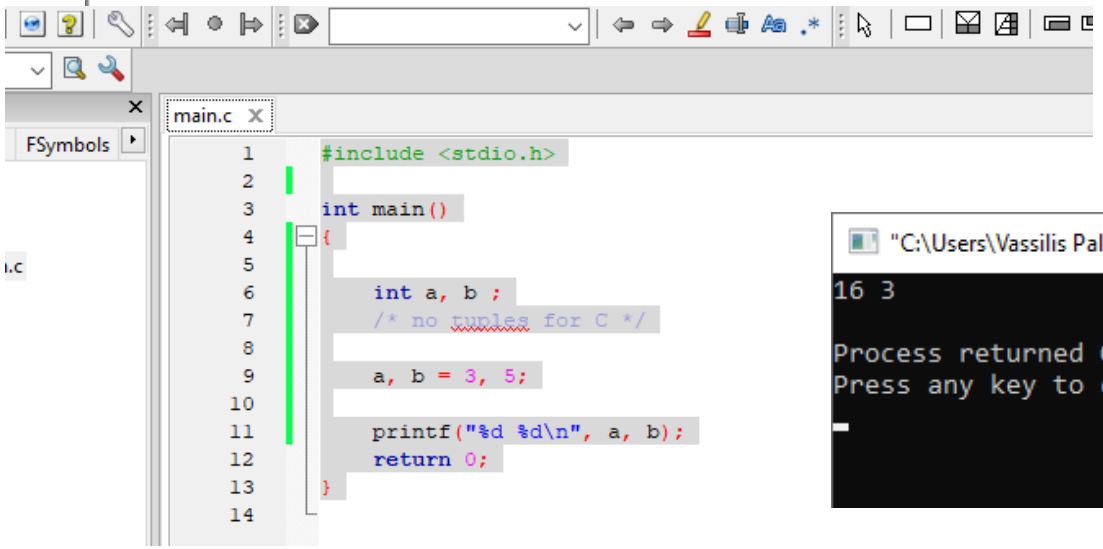
Κατηγορία	Ενδεικτικά C	Ενδεικτικά FORTRAN	Ενδεικτικά Python
Αριθμητικοί	* / % + -	* / + -	Όπως C
Λογικοί	&& || !	.AND. .OR. .NOT.	and, or, not
Συσχετιστικοί	> >= == !=	.GT. .GE. .EQ. .NE.	Όπως C (το διάφορο και <>)
Διαχείρισης δυαδικών ψηφίων μιας λέξης	>> & | ^	έκδοση	Όπως C
Τελεστές διαχείρισης μνήμης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

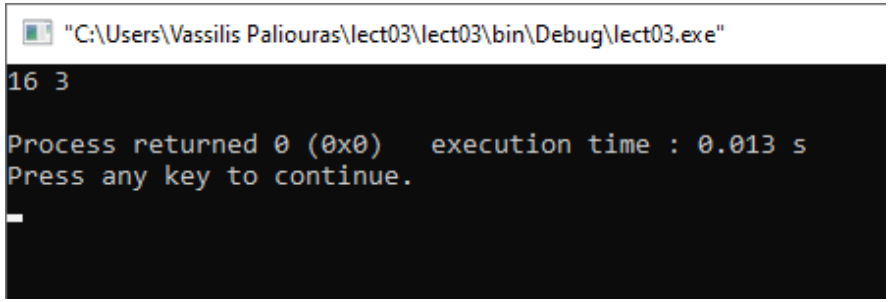
Τελεστές ⇒ Ενέργειες σε δεδομένα

Τελεστές	Προσεταιριστικότητα
( ) [ ] -> .	Αριστερά προς δεξιά
! ~ ++ -- + - * & sizeof	Δεξιά προς αριστερά
* / %	Αριστερά προς δεξιά
+ -	Αριστερά προς δεξιά
<< >>	Αριστερά προς δεξιά
< <= > >=	Αριστερά προς δεξιά
== !=	Αριστερά προς δεξιά
&	Αριστερά προς δεξιά
^	Αριστερά προς δεξιά
|	Αριστερά προς δεξιά
&&	Αριστερά προς δεξιά
||	Αριστερά προς δεξιά
?:	Δεξιά προς αριστερά
= += -= *= /= %= &= ^= |= <<= >>=	Δεξιά προς αριστερά
,	Αριστερά προς δεξιά




Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a,b=3,5
>>> a
3
>>> b
5
>>>


main.c
FSymbols
#include <stdio.h>
int main()
{
int a, b ;
/* no tuples for C */
a, b = 3, 5;
printf("%d %d\n", a, b);
return 0;
}


"C:\Users\Vassilis Paliouras\lect03\lect03\bin\Debug\lect03.exe"
16 3
Process returned 0 (0x0) execution time : 0.013 s
Press any key to continue.

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:

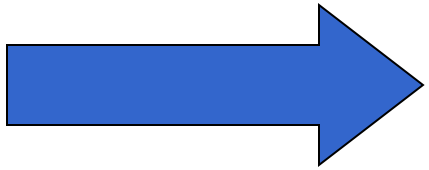
Μπλοκ προτάσεων που ορίζεται με άγκιστρα.

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

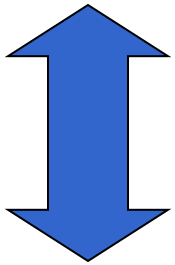
- 13; Δεν είναι συντακτικό λάθος.

Τελεστής Ανάθεσης =

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



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



όλη η έκφραση έχει ως αξία 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 ==

Διαφορές C, python

Το a == b ==c δεν είναι το ίδιο!!!

```
#include <stdio.h>
int main() {
   int a = 5, b =5, c =5;
   if (a==b==c)
        printf("equal?\n") ;
   else
        printf("or not?\n");

   return 0;
}
```

```c
#include <stdio.h>

int main() {

        int a = 5, b =5, c =5;

        if (a==b && b==c)
                        printf("equal\n") ;
        else
                        printf("or not\n");

        return 0;
}
```

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;
}
```

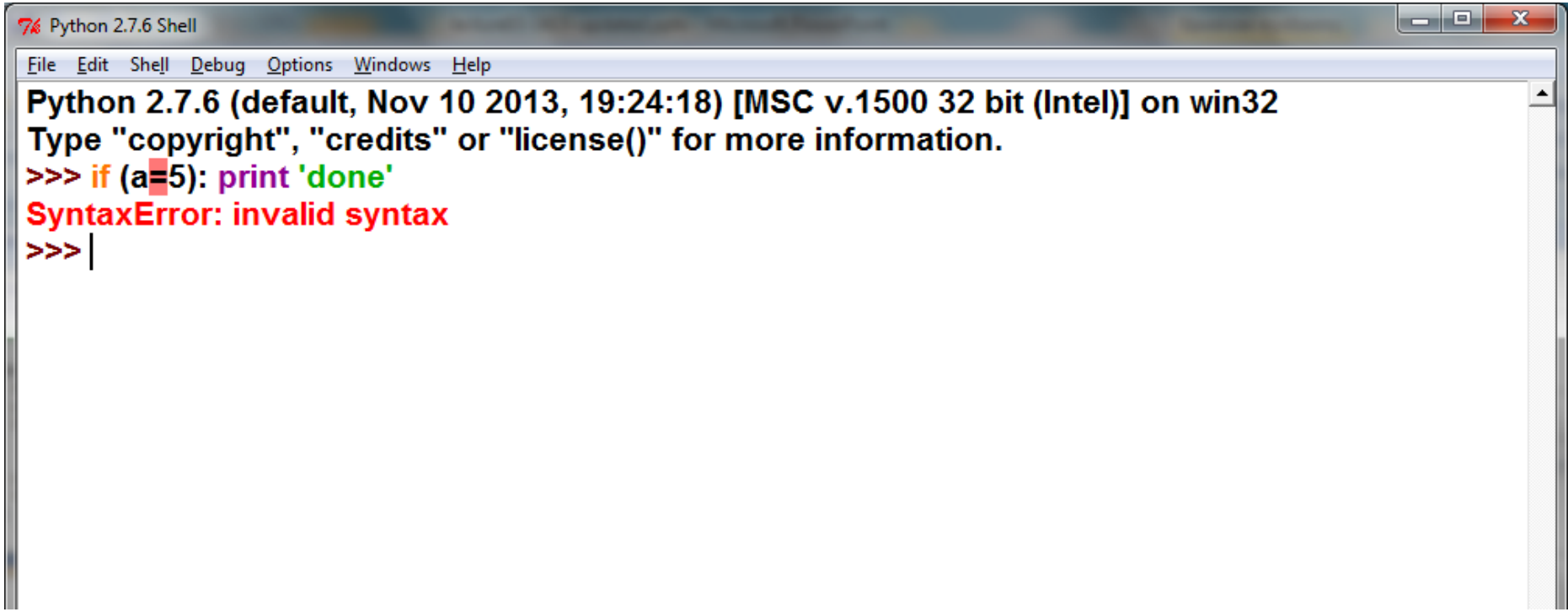
(αντί)-παράδειγμα 2:
Τι θα τυπώσει; Γιατί;

```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;
}
```

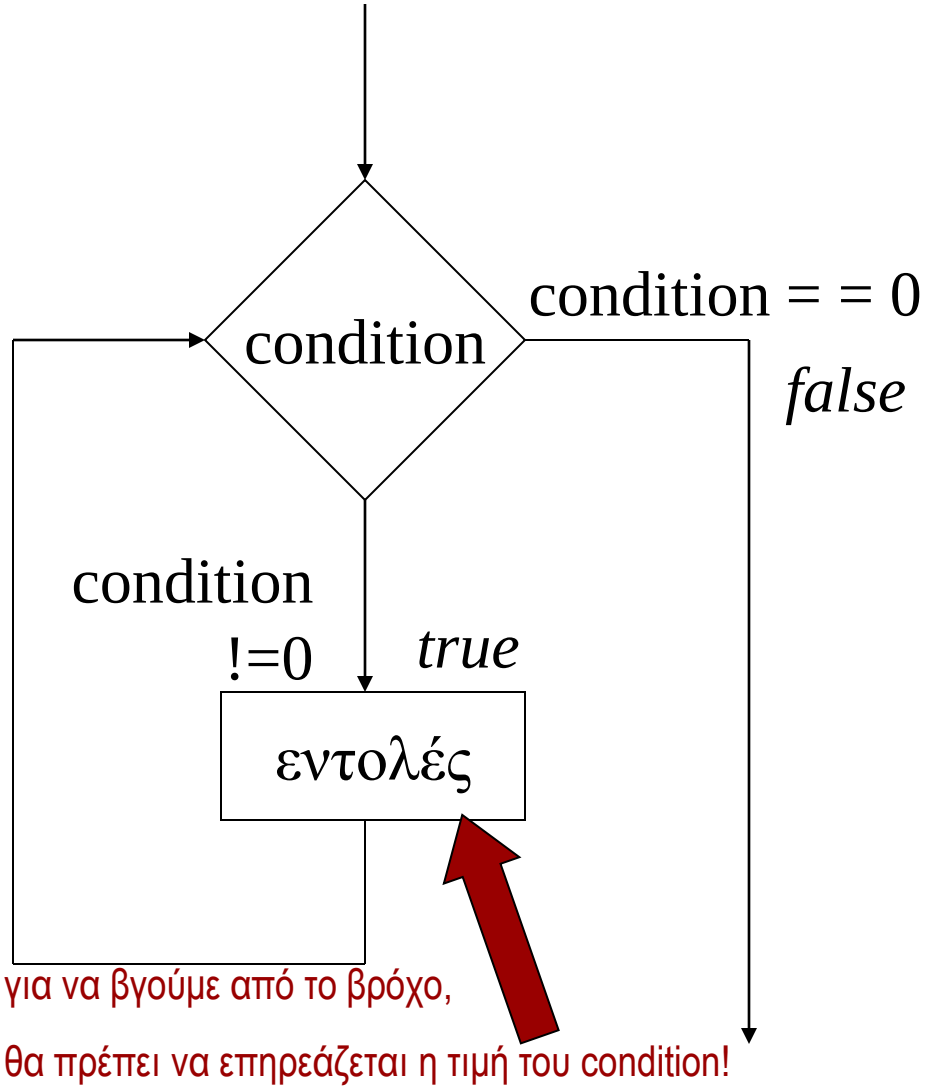
SyntaxError για την Python, όχι για τη C



Βρόχος **while**

συνθήκη **εισόδου** στο βρόχο και **συνέχειας**

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



Βρόχος **while**

while (έκφραση)
    σύνθετη (ή όχι) πρόταση

όσο η έκφραση είναι αληθής, εκτελείται η (σύνθετη) 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 ++;
    }
```

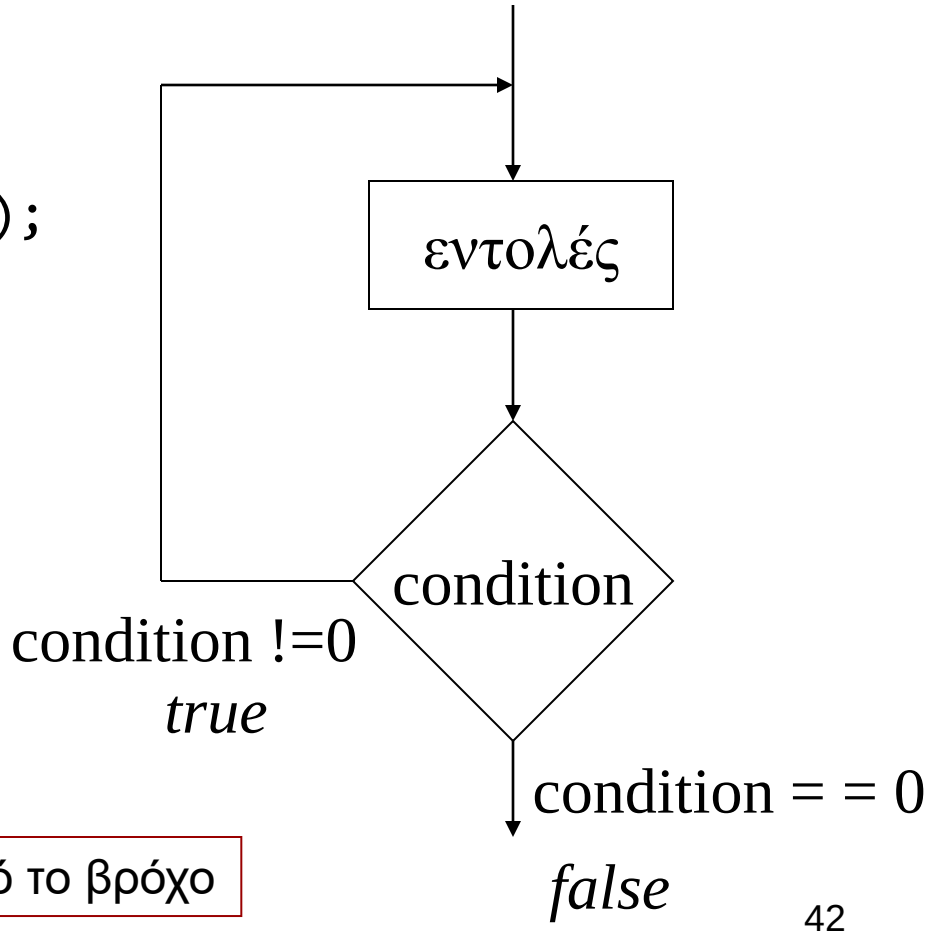
```
for (a = 0; a < 5; a ++) {
    printf ("value of a is %d\n", a);
    }
```

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

1. Σταθερός αριθμός επαναλήψεων
2. Αριθμός επαναλήψεων εξαρτώμενος από τα δεδομένα
3. 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()
- }

```c
#include <stdio.h>

int 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);

  return 0;

}
```

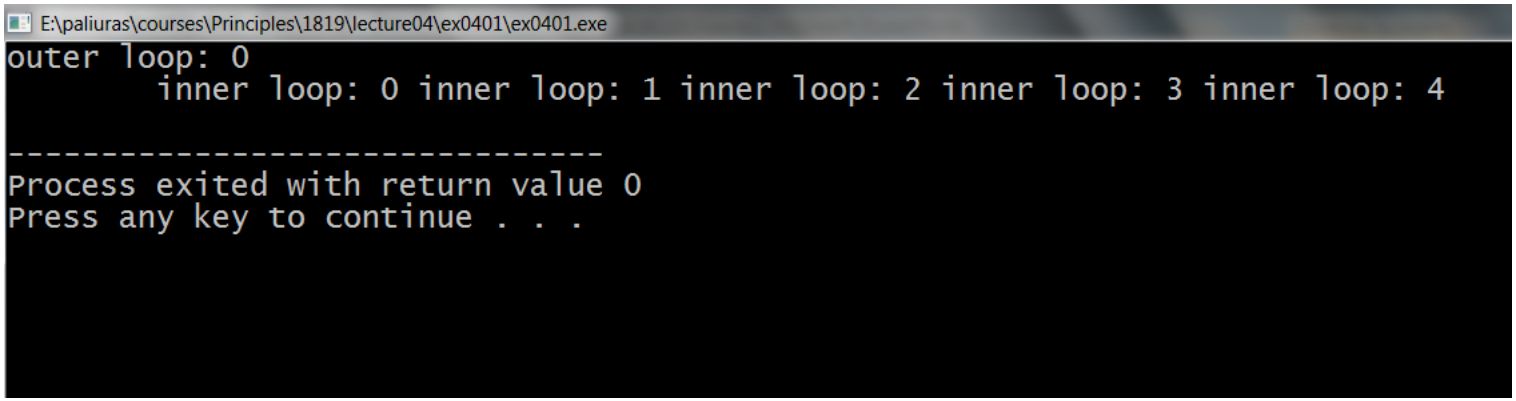
```c
#include <stdio.h>
#define N 10

int 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);

  return 0;
}
```

Nested for σε python

```
for i in range(5):
    print('outer loop: %d\n\t' % i, end='')
    for i in range(5):
        print('inner loop: %d ' % i, end ='')
    print('')
```

```
C:\Python34\python.exe C:/Users/user/PycharmProjects/testnestedfor/testnestedfor.py
outer loop: 0
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 1
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 2
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 3
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 4
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4

Process finished with exit code 0
```

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

int main() {

        int i;
        for (i=0; i< 5; i++) {
                printf("outer loop: %d\n\t", i);
                for (i=0; i<5; i++) {
                                printf("inner loop: %d ", i);
                }
                printf("\n");
        }

        return 0;
}
```



```c
#include <stdio.h>

int main() {

        int i;
        for (i=0; i< 5; i++) {
                printf("outer loop: %d\n\t", i);
                {
                int i;
                for (i=0; i<5; i++) {
                        printf("inner loop: %d ", i);
                        }
                }
                printf("\n");
        }
        return 0;
}
```

C90: δήλωση μεταβλητής μόνο σε αρχή block

```
E:\paliuras\courses\Principles\1819\lecture04\ex0401\ex0401.exe
outer loop: 0
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 1
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 2
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 3
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4
outer loop: 4
        inner loop: 0 inner loop: 1 inner loop: 2 inner loop: 3 inner loop: 4

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

Μπλοκ εντολών με άγκιστρα

C90: δήλωση μεταβλητής μόνο σε αρχή block

```
{
Δηλώσεις ;

Εντολές ;
}
```

Μια διαφορά C90 και C99

```
#include <stdio.h>

int main() {

        for (int i=0; i<5; i++) {
                printf("outer loop: %d\n\t", i);
                for (int i=0; i<5; i++) {
                                printf("inner loop: %d ", i);
                }
                printf("\n");
        }

        return 0;
}
```

Συνήθης πρακτική σε C90: Διαφορετική μεταβλητή σε nested loops

```c
#include <stdio.h>

int main() {

        int i, j;
        for (i=0; i<5; i++) {
                printf("outer loop: %d\n\t", i);
                for (j=0; j<5; j++) {
                                printf("inner loop: %d ", j);
                }
                printf("\n");
        }

        return 0;
}
```

Παράδειγμα

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

```c
#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

```c
#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;
}
```

```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 (1) ;

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

return 0;
}
```

έκδοση 4a

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

Απλή συνθήκη στο **while**

Αρχικοποίηση του input για εξασφάλιση εισόδου στο βρόχο 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;
}
```

```c
#include <stdio.h>

int main( )
{

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

  while ( 1 ) {
     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;
}
```

Η έκφραση πάντα αληθής while

έκδοση 5.1

Σημείο εξόδου από το βρόχο while

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

int main() {
        int a;
        while ( (a = getinput()) > 0) {
                printf("say something\n");
                printf("%d\n", a);
        }

        return 0;
}

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

Σε τι διαφέρουν οι εκφράσεις

```c
a = getinput() > 0

(a = getinput()) > 0

a = (getinput() > 0)
```