ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

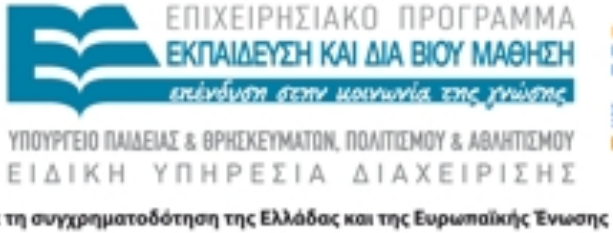
ΑΡΧΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κεφάλαιο 4

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

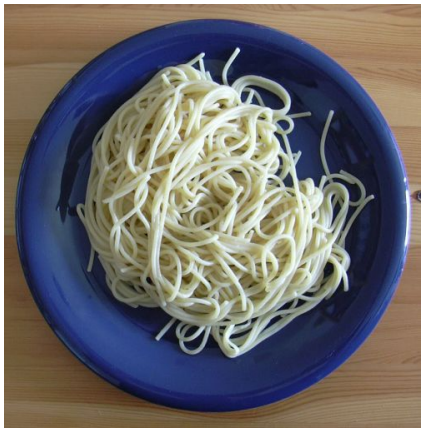
- Οργάνωση προγράμματος στη C
  - συναρτήσεις
- Μεθοδολογίες σχεδιασμού προγραμμάτων
  - Top-down
  - Λεκτική περιγραφή (προστακτικός τρόπος)
  - Αυξητική ανάπτυξη προγράμματος
- Αναγνωριστές, Τελεστές, Εκφράσεις, Προτάσεις
- Βρόχοι επανάληψης στη C
  - `do { σύνθετη εντολή; } while (έκφραση) ;`
  - `while (έκφραση) { σύνθετη εντολή;}`
  - `for (έκφραση1; έκφραση2;έκφραση3)`
    `{σύνθετη εντολή}`

Δομημένος προγραμματισμός

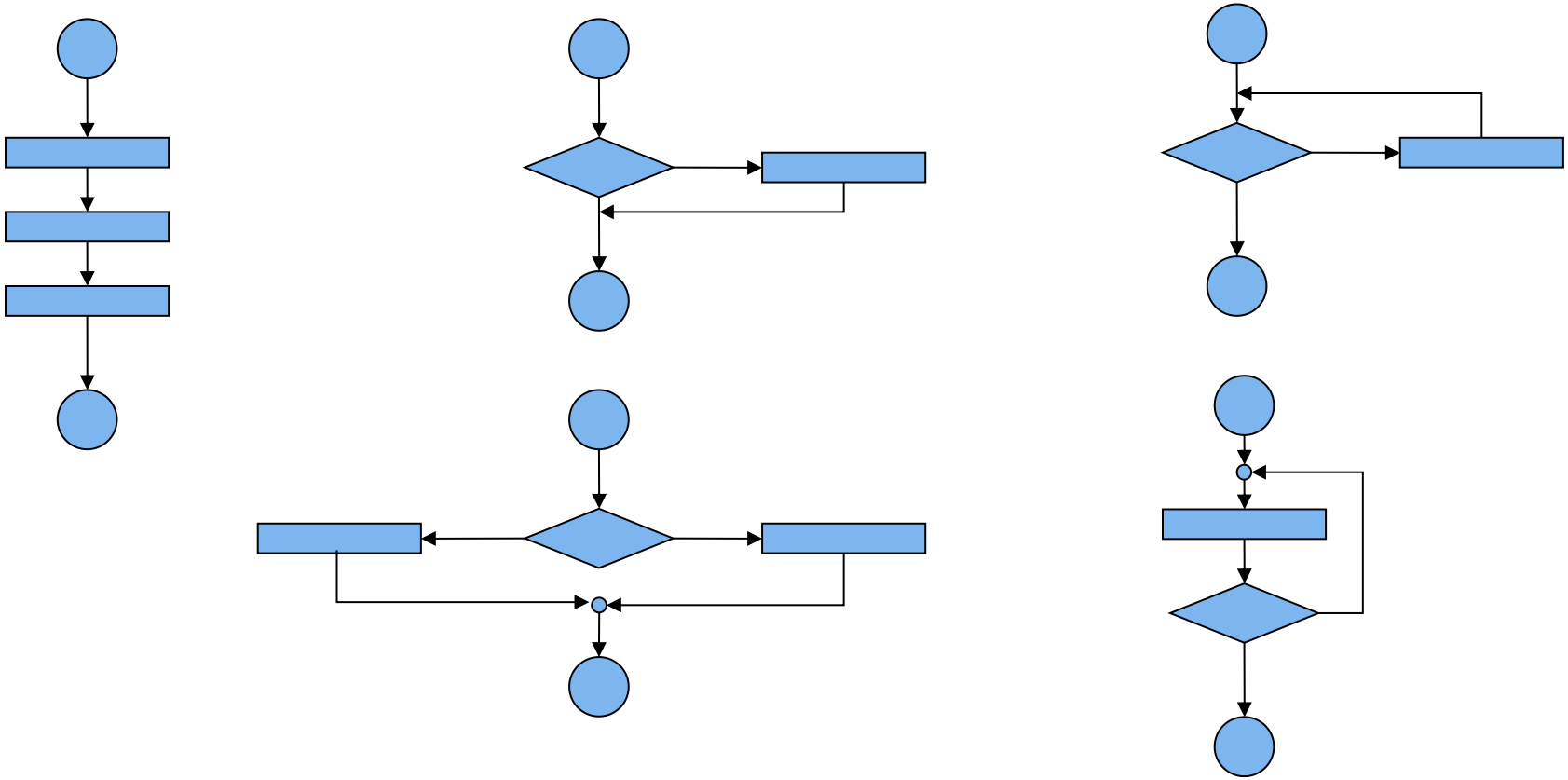
```c
#include <stdio.h>

int main() {

    int i = 0;
    next:
    i = i + 1;
    printf ("%d %d\n", i, i *i);
    if (i>=10) goto end;
    goto next;
    end:
    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int i ;
    for (i=1; i<10; i++)
          printf ("%d %d\n", i, i *i);

    return 0;
}
```

Spaghetti code



Σύνοψη δομών ελέγχου



C. Bohm and C. Jacopini, “Flow diagrams, Turing machines, and languages with only two formation rules,” *Comms. of ACM*, vol. 9, no. 5, May 1966, pp. 336 – 371.

Στοιχεία ελέγχου ροής προγράμματος

- Προγραμματισμός χωρίς χρήση goto
  - ‘goto-less’ programming
  - γνωστό ότι είναι εφικτό από τη δεκαετία του 60
  - στη βιομηχανία, αποδειγμένος ως αποδοτικός

- Αρκεί η υποστήριξη επιλογών ροής
  - απλή, διπλή, πολλαπλή,
  - βρόχοι υπό συνθήκη (do..while, while)

Κανόνες κατασκευής δομημένου προγράμματος

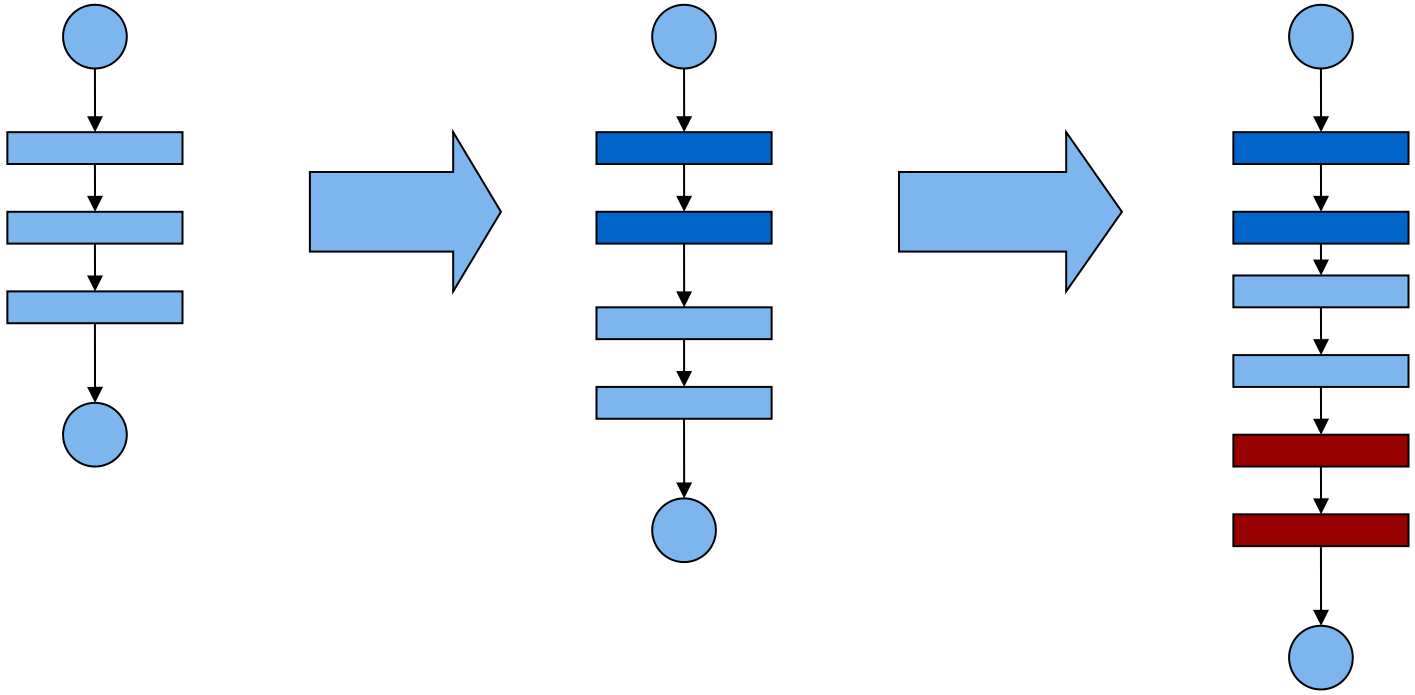
1. Αρχή: Το απλούστερο διάγραμμα ροής
2. Κάθε διεργασία μπορεί να αντικατασταθεί από δύο διεργασίες σε σειρά
3. Κάθε διεργασία μπορεί να αντικατασταθεί από οποιαδήποτε από τις δομές ελέγχου
4. Οι κανόνες 2 και 3 εφαρμόζονται με όποια σειρά και όσες φορές απαιτείται.

Παρατηρήσεις

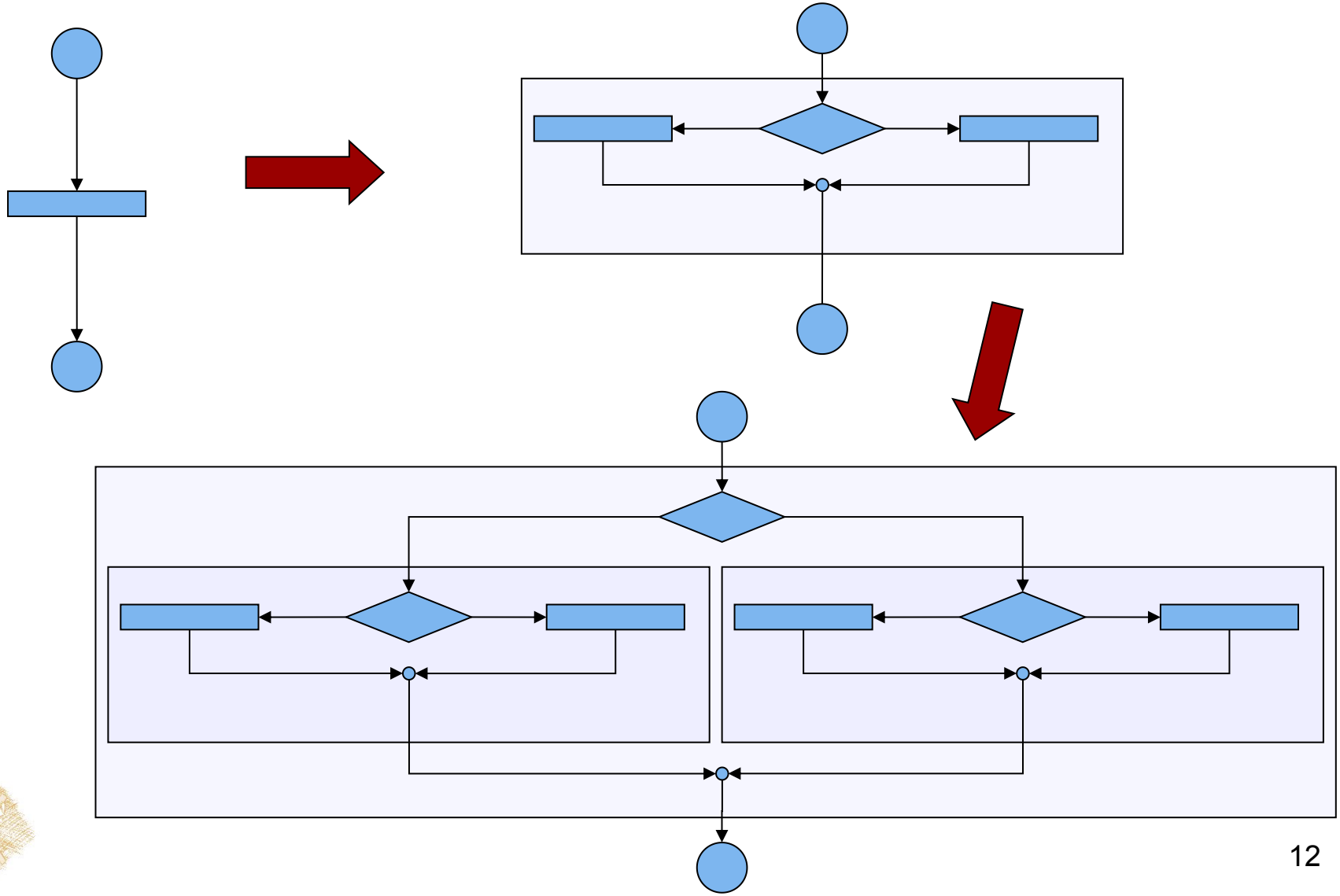
- Δεν περιορίζεται σε διαγράμματα ροής
- Λεκτική περιγραφή – ψευδο-κώδικας
- Progressive refinement: Σταδιακά προσθέτουμε λεπτομέρειες υλοποίησης
- Αυξητική ανάπτυξη κώδικα

Διάγραμμα ροής

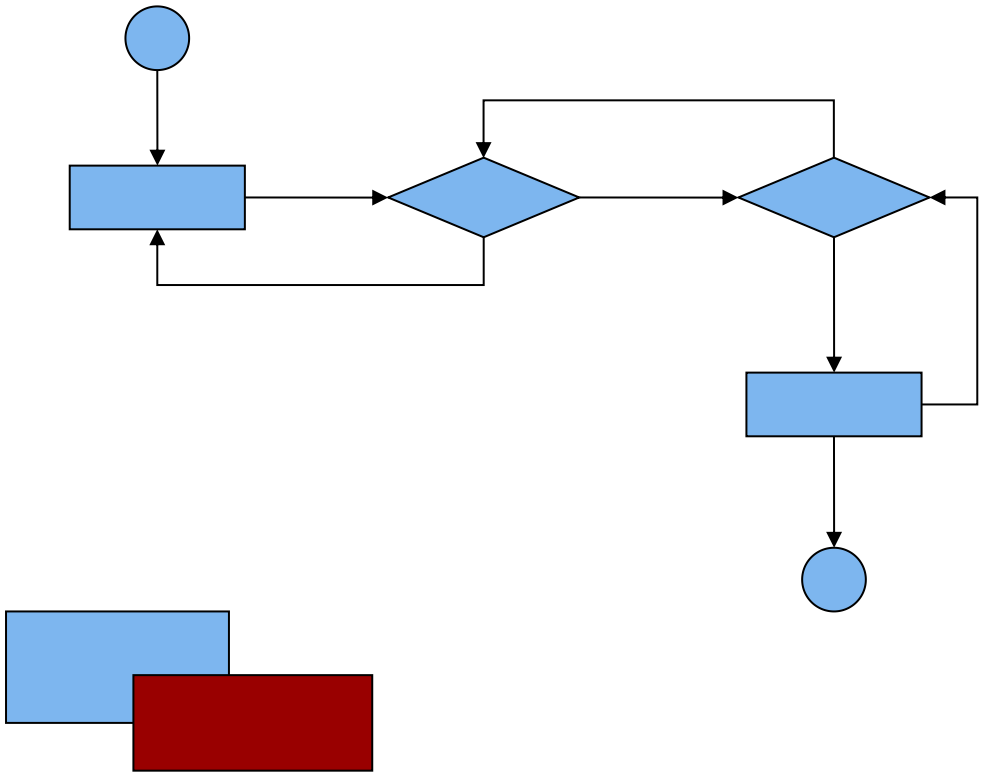
Προσθέτω λεπτομέρεια στο αρχικό διάγραμμα ροής (κανόνας 2)



Διάγραμμα ροής



Μη-δομημένο διάγραμμα ροής



Μη δομημένος κώδικας

```
#include <stdio.h>
int main() {
        int a = 0, b = 7;
        /* bad style: no need to follow*/
        block1: a = a + 2;
                printf("a: %d\n", a);
        check1: if (a<5 ) goto block1;
                b = 7;
        check2: if (b<=5) goto check1;
                a = 0;
                b = b - 1;
                printf("b: %d\n", b);
                goto check2;

    return 0;
}
```

Εφαρμόζοντας καλές τεχνικές ανάπτυξης **ΔΕΝ** προκύπτει τέτοιος κώδικας!

Αληθείς και ψευδείς εκφράσεις

- τιμή έκφρασης != 0 => η έκφραση είναι αληθής

- τιμή έκφρασης == 0 => η έκφραση είναι ψευδής

Λογικές μεταβλητές στη C

- ISO C90
  - Δεν υπάρχει τύπος λογικής μεταβλητής
    - έκφραση !=0 είναι αληθής
- ISO C99
  - Επεκτείνει το C90
  - Ορίζει **_Bool**
  - Τυποποιεί το header file <stdbool.h>
    - **bool, true, false**

Ομαδοποίηση Τελεστών

Κατηγορία	Ενδεικτικά C
Αριθμητικοί	* / % + -
Λογικοί	&& || !
Συσχετιστικοί	> >= == !=
Διαχείρισης δυαδικών ψηφίων μιας λέξης	>> & | ^
Τελεστές διαχείρισης μνήμης	& [ ] . -> *

iso646.h

- Τροποποίηση C95 του C90

Macro	Ορίζεται ως
and	&&
and_eq	&=
bitand	&
bitor	|
compl	~
not	!
not_eq	!=
or	||
or_eq	|=
xor	^
xor_eq	^=

Παράδειγμα ISO C99

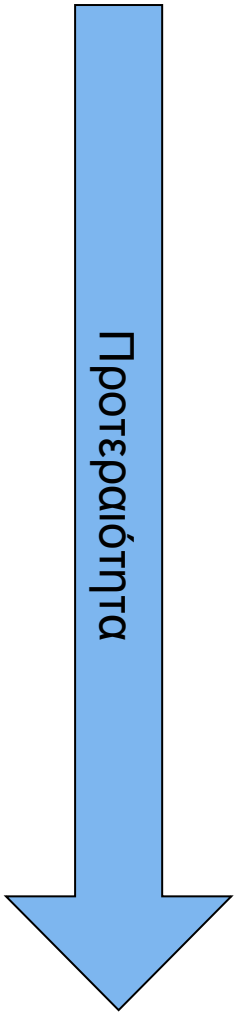
```c
#include <stdio.h>
#include <stdbool.h>
#include <iso646.h>
// C99 style for comments and boolean logic

int main() {
            bool bmorethana;
            bool blessthanc;
            bool between;
            int a = 1, b =2, c =3;

            bmorethana = b>a ;
            blessthanc = b<c ;

            if(bmorethana and blessthanc) {
                printf("b is between 'em\n");
                between = true;
            }
            else {
                printf("b is out of limits\n");
                between = false ;
            }

            if (between)
               printf("between is true");

            return 0;
}
```

Στα πλαίσια του μαθήματος γράφουμε κυρίως ISO C90

Portability

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



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 ==

Έλεγχος Ισότητας ==

```
main () {
int a = 1, b =1;

if ( a == b)
   printf ("equal");
else
   printf ("unequal");

}
```

```
main () {
int a = 1, b =1;

if ( a != b)
   printf ("unequal");
else
   printf ("equal");

}
```

Ανάθεση και Ισότητα

```
main () {
   int a = 1, b =1;
   int condition;

   condition = (a==b);

   if (condition)
       printf("equal");
   else
       printf("unequal");
}
```

```
main () {
   int a = 1, b =1;
   int condition;

   condition = (a==b);

   if (!condition)
       printf("unequal");
   else
       printf("equal");
   }
```

Λογικοί τελεστές

```
#include <stdio.h>

void main () {

int a=1, b=2, c=3;

if (b>a && b<c)
   printf("b is between 'em");
else
   printf("b is out of limits");

}
```

Prefix – postfix notation

- a++;
- ++a;
- f(a++);
- f(++a);

Τι τυπώνεται;

```
#include <stdio.h>

int main() {

        int a = 4;

        printf("%d\n", a);
        a++;
        printf("%d\n", a);
        ++a;
        printf("%d\n", a++);
        printf("%d\n", a);
        printf("%d\n", ++a);
        printf("%d\n", a);

        return 0;
}
```

Σύνδεση εκφράσεων με λογικούς τελεστές

```c
#include <stdio.h>

int main()
{
  int a = 1, b = 0;

  if (a==1 || ++b ==1)
          printf("hello\n");

  printf("value of b after if: %d\n", b);

   return 0;
}
```

Τι αλλάζει στη συμπεριφορά, αν το a αρχικοποιηθεί στο 0;

Short-circuit evaluation

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

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

```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 2.1:Αριθμός επαναλήψεων εξαρτώμενος από τα δεδομένα

- Να γραφεί ένα πρόγραμμα που διαβάζει ακεραίους, έναν κάθε φορά και τυπώνει το μερικό άθροισμα, 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 5

```
#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;
```

Παράδειγμα 3.1

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

Παράδειγμα 3.1
΄Εκδοση 1

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

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

  while ( input > 0) {
     scanf("%d", &input) ;
     if (input > 0 ) {
              if ( input % 2 == 0 ) {
                  sum = sum + input ;
                  printf("input even, partial sum: %d\n", sum);
                  prod = prod * input ;
                }
        }
 }
  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);
  return 0;
```

Παράδειγμα 3.1
Έκδοση 1.1

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

int main( )
{

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

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

     if (input > 0 &&  input % 2 == 0 ) {
                    sum = sum + input ;
                    printf("input even, partial sum: %d\n", sum);
                    prod = prod * input ;
                  }
   }
  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);
  return 0;
```

λογική σύζευξη (ΚΑΙ, AND)

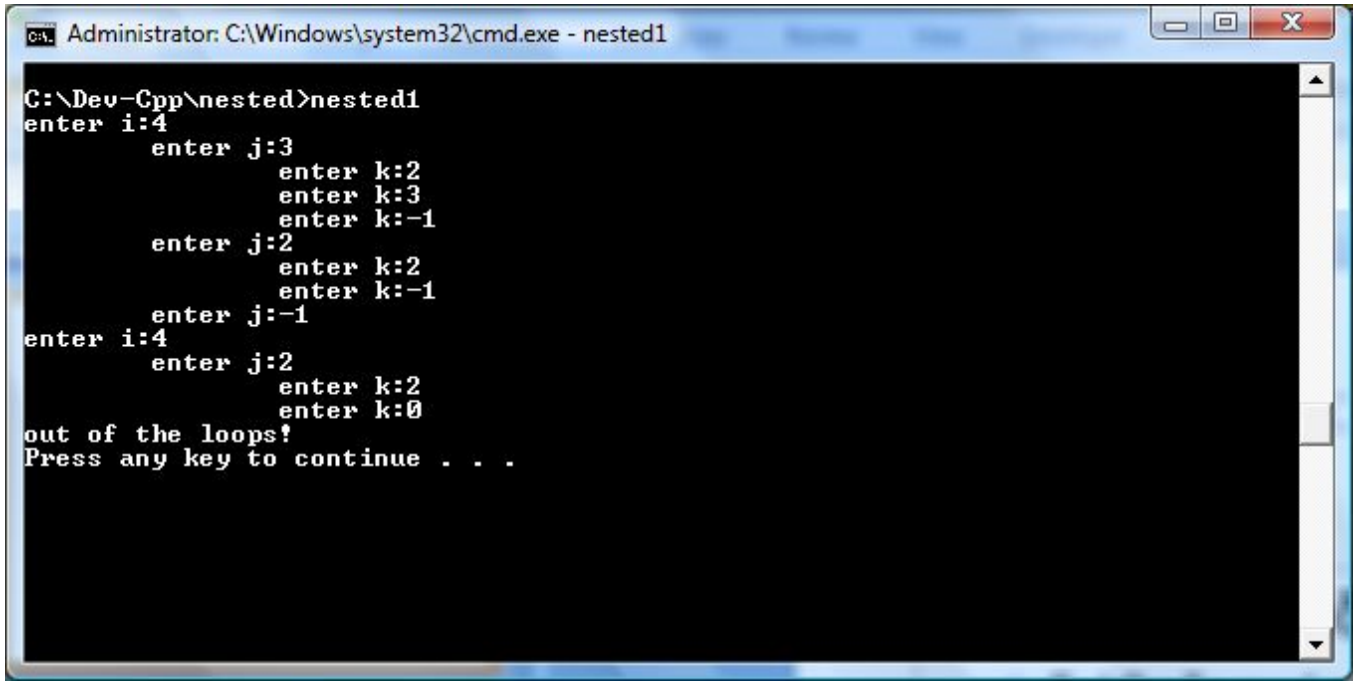
Παράδειγμα 3.1
έκδοση 2

```
#include <stdio.h>

int main( )
{
    int input=1, sum = 0 , prod = 1;

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

     if (input <=0 )
                 break ;

     if (input % 2 != 0)
                 continue;

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

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

  return 0;
}
```

Παράδειγμα 4:Ένθετοι βρόχοι επανάληψης

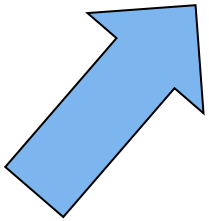
- Διάβασε τριάδες ακεραίων i, j, k ως εξής
- διάβαζε τιμές i, όσο i>0. Για κάθε i:
- αν i <=0, σταμάτα αλλιώς
- διάβαζε τιμές του j, όσο j>0. Για κάθε j
- αν j <=0, διάβασε νέα τιμή του i αλλιώς
- διάβαζε τιμές του k, όσο k>0. Για κάθε k
- αν k ==0 σταμάτα το διάβασμα όλων
- αν k <=0 διάβασε νέα τιμή του j

Αποτέλεσμα


Administrator: C:\Windows\system32\cmd.exe - nested1
C:\Dev-Cpp\nested>nested1
enter i:4
enter j:3
enter k:2
enter k:3
enter k:-1
enter j:2
enter k:2
enter k:-1
enter j:-1
enter i:4
enter j:2
enter k:2
enter k:0
out of the loops!
Press any key to continue . . .

Έκδοση 0 Λεκτική περιγραφή

```
Αρχικοποίηση i
Όσο (i>0) {
     Διάβασε i
     Αν (i > 0) {
          Αρχικοποίηση j
          Όσο (j > 0) {
               Διάβασε j
               Αν (j > 0) {
                    Αρχικοποίηση k
                    Όσο (k>0) {
                         Διάβασε k
                         Αν (k == 0) {
                                        βγες εκτός των βρόχων
                                        }
                    }
               }

          }
     }
     }
```



Πώς θα γίνει αυτό;

40

Έκδοση 1

```
#include <stdio.h>
int main() {
  int i ,j, k, sum;
  int exitall = 0 ;
  i = 1;
  while(i>0 && !exitall) {
            printf("enter i:");
            scanf("%d", &i);
            if ( i > 0) {
              j = 1;
              while (j>0 && !exitall) {
                 printf("\tenter j:");
                 scanf("%d", &j);
                 if ( j > 0) {
                    k = 1;
                    while (k > 0 ) {
                       printf("\t\tenter k:");
                       scanf("%d", &k);
                       if ( k == 0)
                             exitall = 1;
                       else {
                          if (k > 0 ) {
                             sum = sum + k;
                             }
                          }
                       }
                    }
                 }
              }
  }
printf("out of the loops!\n");
return 0;
}
```

Δομημένο στυλ

exitall

Ελέγχει την έξοδο από

Βρόχους επανάληψης.

Έκδοση 2

```c
#include <stdio.h>
int main( )  {
  int i ,j, k, sum;
  i = 1;
  while (i>0) {
            printf("enter i:");
            scanf("%d", &i);
            if ( i > 0) {
               j = 1;
               while (j>0 ) {
                  printf("\tenter j:");
                  scanf("%d", &j);
                  if ( j > 0) {
                     k = 1;
                     while (k > 0 ) {
                        printf("\t\tenter k:");
                        scanf("%d", &k);
                        if ( k == 0)
                              goto EXITLOOPS;
                        else {
                            if (k > 0 ) {
                               sum = sum + k;
                               }
                            }
                     }
                  }
               }
            }
   }
  EXITLOOPS:
  printf("out of the loops!\n");
  return 0;
}
```

Πολλαπλές επιλογές: switch

```
switch (έκφραση)  {
   case τιμή1: εντολές ; break;
   case τιμή2: εντολές ; break;
   case τιμή3: εντολές ; break;
   /* … */
   default: εντολές ; break;
   }
```

case τιμή1: λειτουργεί ως label

break: μεταφέρει τον έλεγχο εκτός του switch () {}

τι γίνεται χωρίς **break** ;

Παράδειγμα

- Να γραφεί πρόγραμμα τέτοιο ώστε το σύστημα να ζητά από το χρήστη **να επιλέξει μεταξύ τριών επιλογών**:
  - να ξεκινήσει μια συγκεκριμένη διεργασία,
  - να σταματήσει η διεργασία,
  - να λήξει η εκτέλεση του προγράμματος.
- Θα ζητείται είσοδος από το χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

Παράδειγμα – Λεκτική περιγραφή λύσης

- Ζήτησε από το χρήστη **να επιλέξει μεταξύ τριών επιλογών**: int userchoice()
  - να ξεκινήσει μια συγκεκριμένη διεργασία,
  - να σταματήσει η διεργασία,
  - να λήξει η εκτέλεση του προγράμματος.
- Συνέχισε να ζητάς επιλογή από χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

Παράδειγμα – Λεκτική περιγραφή λύσης (2)

- Ζήτησε από το χρήστη **να επιλέξει μεταξύ τριών επιλογών**:
- Ανάλογα με την userchoice
  - αν είναι 1, ξεκίνησε τη διεργασία, `getchoice( )`
  - αν είναι 2, σταμάτα τη διεργασία, `start( )`
  - αν είναι 3, να λήξει η εκτέλεση του προγράμματος. `stop()`
- Συνέχισε να ζητάς επιλογή από χρήστη **έως ότου επιλε γεί η λήξη** του προγράμματος.

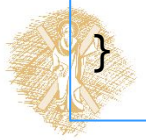
Λεκτική περιγραφή λύσης

userchoice = getchoice();
while (δεν επιλέχθηκε η λήξη) {
    Ανάλογα με την userchoice
        αν είναι 1, start();
        αν είναι 2, stop();
        αν είναι 3, να λήξει η εκτέλεση του προγράμματος.
    userchoice = getchoice();
    }

Οργάνωση βασικού βρόχου (1)

```
main ( ) {
    int userchoice ;

    userchoice = getchoice ( ) ;

    while (userchoice != 3 ) {
        switch (userchoice) {
           case 1: start( ); break;
           case 2: stop( ); break;
           default: break;
         }
        userchoice =  getchoice() ;
  }
```

```
main ( ){
int userchoice ;

userchoice = getchoice();

while (userchoice != 3) {
   switch (userchoice) {
        case 1: start( );
                break;
        case 2: stop( );
                break;
        default:break;
       }

   userchoice=getchoice();

   }

}
```



```
main ( ) {
int userchoice ;

while((userchoice=getchoice
   ())!= 3) {
       switch (userchoice)
        {
        case 1: start();
   break;
        case 2: stop();
   break;
        default: break;
        }

   }
}
```

Υλοποίηση συνάρτησης getchoice( )

```
int getchoice (void) {
    int choice ;

    printf("1: start\n2: stop\n3: quit\n");
    printf("enter choice:\n");
    scanf("%d", &choice);

    return choice;
}
```

κλήση (χρήση)

(πχ στην υλοποίηση της main())

userchoice = getchoice( );

Παραδείγματα

```c
#include <stdio.h>
int getchoice (void) ;
void start (void) ;
void stop (void);

main (){

    int userchoice ;

    while ((userchoice = getchoice()) !=
    3){
      switch (userchoice) {
          case 1: start() ;
                  break;
          case 2: stop();
                  break;
          default: break;
      }
    }

}
```

```c
int getchoice (void ) {
 int choice ;

 printf("1: start\n2: stop\n3:quit\n");
 printf("enter choice:\n");
 scanf("%d", &choice);

 return choice;
}

void start (void) {
    printf("Start...");
}

void stop (void) {
    printf("Stop...");
}
```

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/