Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς

Αληθείς και Ψευδείς Εκφράσεις

- `τιμή έκφρασης != 0` $\Rightarrow$ η έκφραση είναι **αληθής**

- `τιμή έκφρασης == 0` $\Rightarrow$ 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

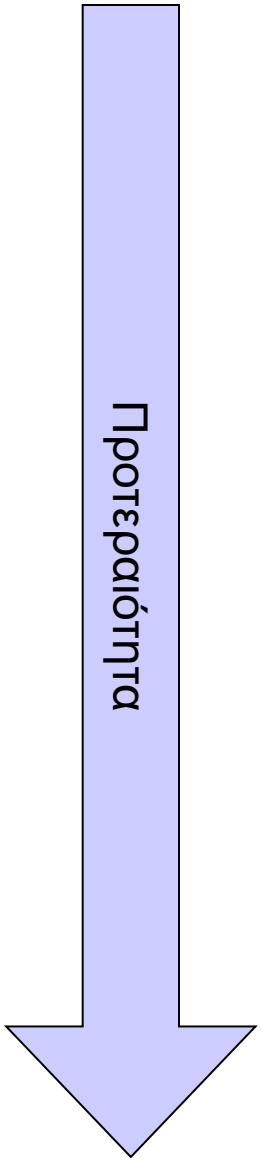
```c
#include <stdio.h>
#include <stdbool.h>
#include <iso646.h>
// C99 style for comments and boolean logic

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 ==

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

if ( a == b) {
   printf ("equal");
}
else {
   printf ("unequal");
}
return 0;
}
```

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

if ( a != b) {
   printf ("unequal");
}
else {
   printf ("equal");
}
return 0;
}
```

Ανάθεση και Ισότητα

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

  condition = (a==b);

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

return 0;
}
```

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

  condition = (a==b);

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

return 0;
}
```

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");

return ;
}
```

Συμβολισμός Προθέματος-Επιθέματος Prefix – Postfix notation

- `a++;  /* postfix */`
- `++a;  /* prefix */`
- `f(a++); /*  function called, then a updated */`
- `f(++a); /* a updated, then function called */`

Τι τυπώνεται;

```c
#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

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

```
#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;
}
```

Short-circuit evaluation

Ισοδύναμος κώδικας για || στη λογική συνθήκη

```
#include <stdio.h>

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

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

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

  return 0;
}
```

Στην περίπτωση συνθήκης με &&, σε τι θα διέφερε;

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

- 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;
}
```

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

```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;
}
```

Παράδειγμα 3.1

- Να γραφεί ένα πρόγραμμα που διαβάζει **ακεραίους** έναν 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 έκδοση 1

```c
#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

```c
#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;
}
```

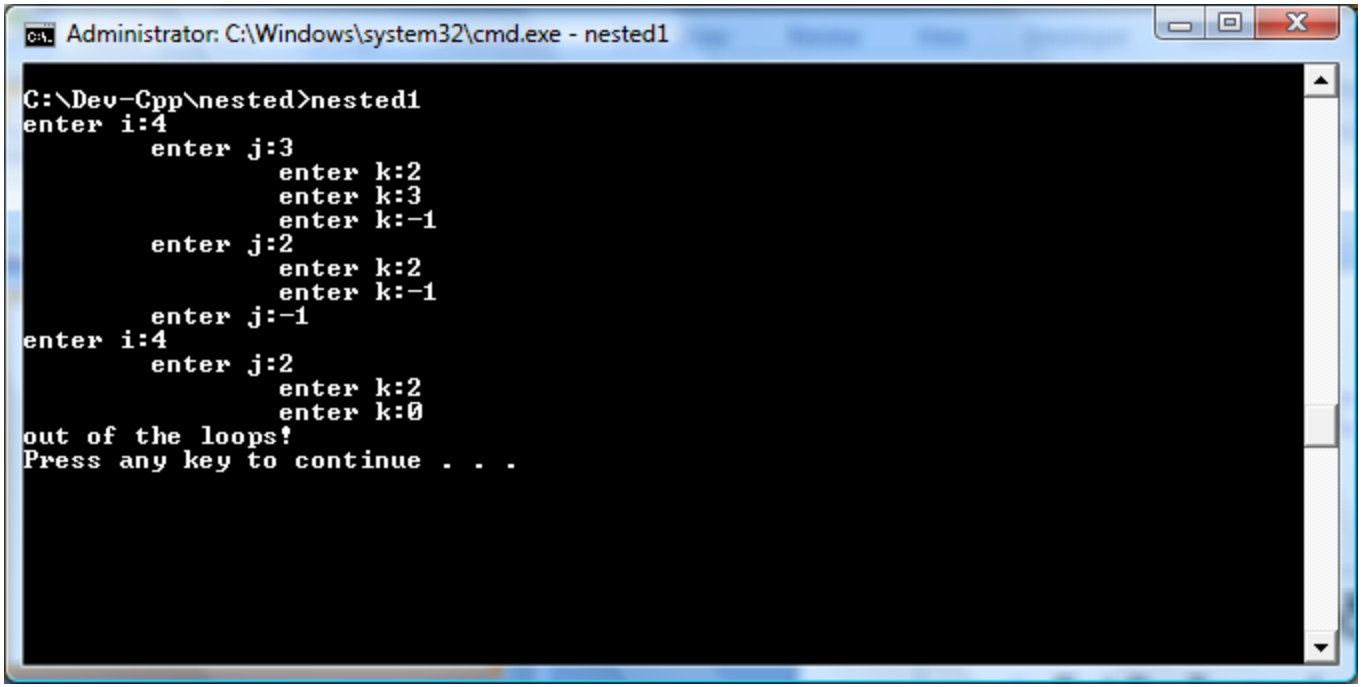
λογική σύζευξη (ΚΑΙ, AND)

Παράδειγμα 3.1 έκδοση 2

```c
#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;
}
```

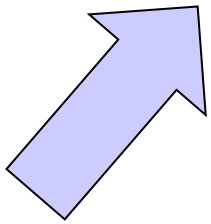
Παράδειγμα 4: Ένθετοι βρόχοι επανάληψης

- 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


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

```
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;

```c
#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;
}
```

έκδοση 1

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;
}
```