Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

Γλώσσα Προγραμματισμού C

4η Διάλεξη

Δημοσθένης Κυριαζής

Τρίτη 23 Οκτωβρίου 2018

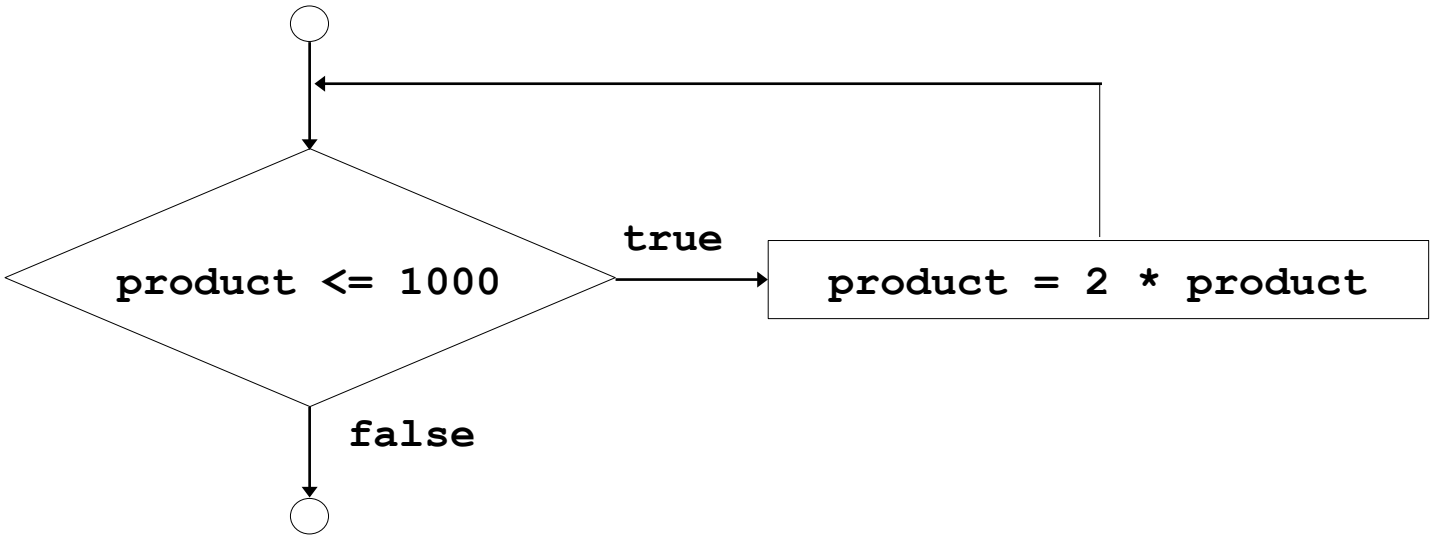
Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Η εντολή επανάληψης while

- Παράδειγμα

```
int product = 2;
while ( product <= 1000 )
        product = 2 * product;
```



Η εντολή επανάληψης do...while

- Παρόμοια με τη while
- Συνθήκη επανάληψης ελέγχεται αφού έχει εκτελεστεί ο κορμός του βρόχου
  - Οι εντολές εκτελούνται τουλάχιστον μια φορά
- Γενική μορφή:

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

- Παράδειγμα (έστω counter = 1):

```
do {
   printf( "%d  ", counter );
} while (++counter <= 10);
```

  - Τυπώνει τους ακέραιους από 1 ως 10

Σύγκριση "επανάληψης" με μετρητή / τιμή «σημαία»

- Επανάληψη ελεγχόμενη από μετρητή
  - Καθορισμένη επανάληψη: ξέρουμε πόσες φορές θα εκτελεστεί ο βρόχος
  - Χρήση μεταβλητής ελέγχου για τη μέτρηση των επαναλήψεων
- Επανάληψη ελεγχόμενη από τιμή "σημαία"
  - Αόριστη επανάληψη
  - Χρησιμοποιείται όταν ο αριθμός των επαναλήψεων δεν είναι γνωστός
  - Η τιμή σημαία σημαίνει το τέλος των δεδομένων

Κύρια χαρακτηριστικά επανάληψης με μετρητή

- Χρειάζεται
  - Μια μεταβλητή ελέγχου (ή μετρητής βρόχου)
  - Αρχικοποίηση της μεταβλητής ελέγχου
  - Μια αύξηση (ή μείωση) μέσω της οποίας η εντολή ελέγχου μεταβάλλεται κάθε φορά μέσα στο βρόχο
  - Μια συνθήκη που ελέγχει την τιμή της μεταβλητής ελέγχου κι ανάλογα συνεχίζεται ή τερματίζεται ο βρόχος

Κύρια χαρακτηριστικά επανάληψης με μετρητή

- Παράδειγμα

```
int counter = 1;                // αρχικοποίηση
while ( counter <= 10 ) {       // επανάληψη
   printf( "%d\n", counter );
   ++counter;                   // αύξηση
}
```

- Εναλλακτικά

```
int counter = 1;                // αρχικοποίηση
while ( ++counter <= 10 )       // αύξηση και επανάληψη
{
   printf( "%d\n", counter );
}
```

Η εντολή επανάληψης for (1/2)

- Χρησιμοποιείται για την επανάληψη μιας εντολής ή μιας ομάδας εντολών κατά έναν προσδιοριζόμενο αριθμό φορών
- Γενική μορφή εντολής for

```
for (αρχικοποίηση; συνθήκη-ελέγχου; αύξηση/μείωση)
    εντολή;
```

  - Αρχικοποίηση
    - παροχή αρχικής τιμής στη μεταβλητή που ελέγχει το βρόχο
    - εκτελείται μόνο μια φορά πριν ξεκινήσει ο βρόχος
  - Συνθήκη-ελέγχου
    - ελέγχει την τιμή της μεταβλητής ελέγχου και τη συγκρίνει με μια τιμή προορισμού στην αρχή του βρόχου
      - Αν το αποτέλεσμα είναι true ο βρόχος επαναλαμβάνεται
      - Αν το αποτέλεσμα είναι false ο βρόχος σταματά και η εκτέλεση του προγράμματος συνεχίζει με την εντολή που βρίσκεται αμέσως μετά από το βρόχο

Η εντολή επανάληψης for (2/2)

- Οι βρόχοι for μπορούν συνήθως να γραφτούν και σαν βρόχοι while

```
αρχικοποίηση;
while ( συνθήκη-ελέγχου) {
   εντολή;
   αύξηση/ μείωση;
}
```

- Αρχικοποίηση και αύξηση / μείωση
  - Μπορούν να αποτελούνται από μια ομάδα εντολών που διαχωρίζονται με κόμμα (,)
  - Παράδειγμα

```
for (i = 0, j = 0;  j + i <= 10; j++, i++)
   printf( "%d\n", j + i );
```

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

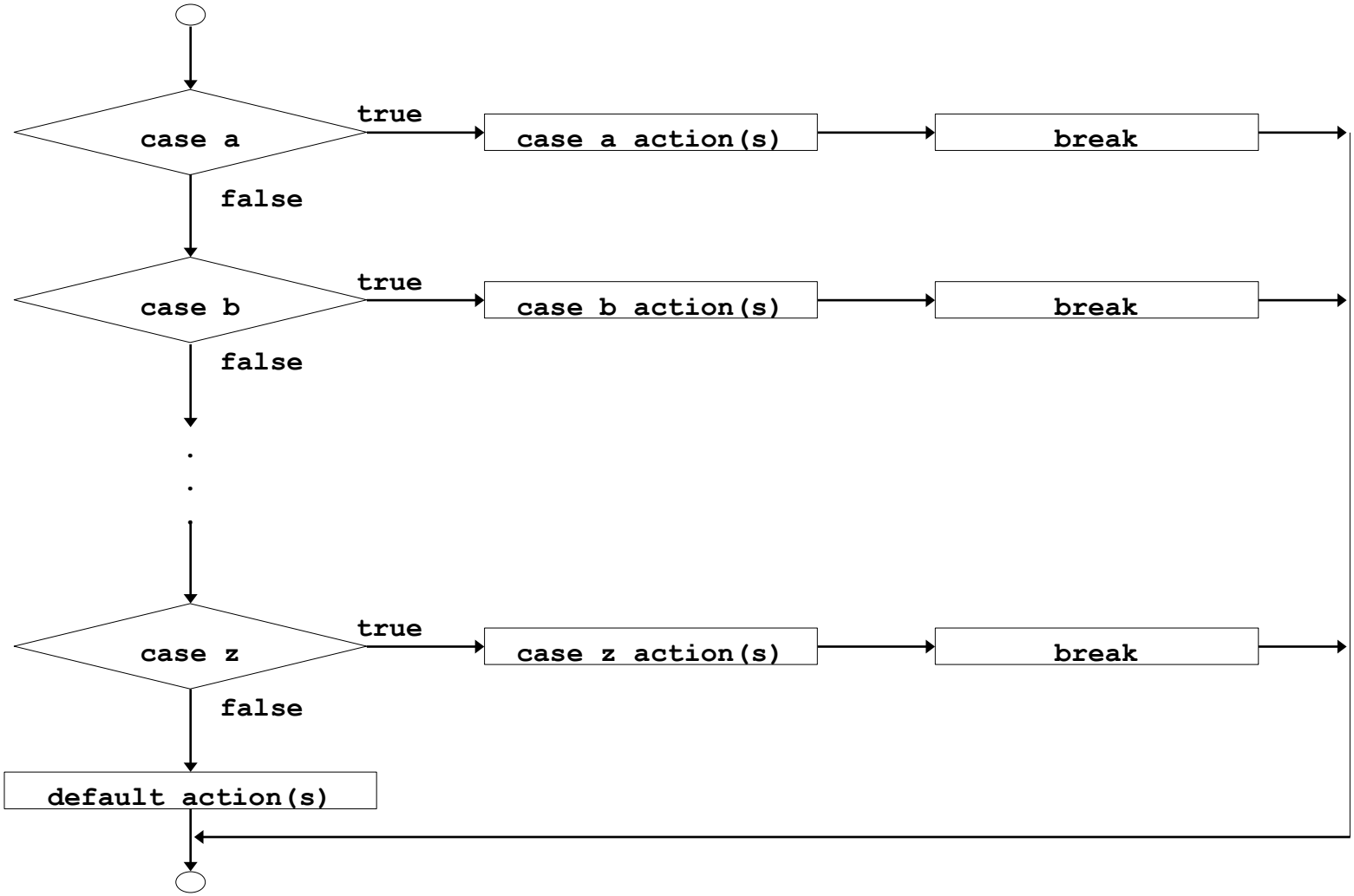
Η εντολή πολλαπλής επιλογής switch (1/2)

- switch
  - Χρήσιμη όταν μια μεταβλητή ή έκφραση ελέγχεται για όλες τις δυνατές τιμές που μπορεί να πάρει και γίνονται διαφορετικές ενέργειες
- Η τιμή ελέγχεται διαδοχικά έναντι μιας λίστας σταθερών (ακέραιων ή χαρακτήρων)
  - Όταν βρεθεί ένα «ταίριασμα» εκτελείται η αλληλουχία εντολών που σχετίζεται με αυτή την περίπτωση - case - μέχρι να βρεθεί μια break
  - default
    - Η αλληλουχία εντολών της εκτελείται αν δεν βρεθεί κανένα «ταίριασμα»
    - Προαιρετική: αν δεν υπάρχει δεν εκτελείται καμία εντολή

Γενική μορφή

```
switch ( τιμή ){
    case σταθερά1:
        εντολές
        break;
    case σταθερά2:
        εντολές
        break;
    .
    .
    .
    default:
        εντολές
        break;
    }
```

Η εντολή πολλαπλής επιλογής `switch` (2/2)



Παράδειγμα (1/4)

```
1  /* Counting letter grades */
2  #include <stdio.h>
3
4  /* function main begins program execution */
5  int main()
6  {
7     int grade;      /* one grade */
8     int aCount = 0; /* number of As */
9     int bCount = 0; /* number of Bs */
10    int cCount = 0; /* number of Cs */
11    int dCount = 0; /* number of Ds */
12    int fCount = 0; /* number of Fs */
13
14    printf(  "Enter the letter grades.\n"  );
15    printf(  "Enter the EOF character to end input.\n"  );
16
17    /* loop until user types end-of-file key sequence */
18    while ( ( grade = getchar() ) != EOF ) {
19
20       /* determine which grade was input */
21       switch ( grade ) { /* switch nested in while */
22
23          case 'A':    /* grade was uppercase A */
24          case 'a':    /* or lowercase a */
25             ++aCount; /* increment aCount */
26             break;    /* necessary to exit switch */
27
```

Παράδειγμα (2/4)

```
28          case 'B':    /* grade was uppercase B */
29          case 'b':    /* or lowercase b */
30             ++bCount; /* increment bCount */
31             break;    /* exit switch */
32
33          case 'C':    /* grade was uppercase C */
34          case 'c':    /* or lowercase c */
35             ++cCount; /* increment cCount */
36             break;    /* exit switch */
37
38          case 'D':    /* grade was uppercase D */
39          case 'd':    /* or lowercase d */
40             ++dCount; /* increment dCount */
41             break;    /* exit switch */
42
43          case 'F':    /* grade was uppercase F */
44          case 'f':    /* or lowercase f */
45             ++fCount; /* increment fCount */
46             break;    /* exit switch */
47
48          case '\n':   /* ignore newlines, */
49          case '\t':   /* tabs, */
50          case ' ':    /* and spaces in input */
51             break;    /* exit switch */
52
```

Παράδειγμα (3/4)

```
53          default:     /* catch all other characters */
54             printf( "Incorrect letter grade entered." );
55             printf( " Enter a new grade.\n" );
56             break;    /* optional; will exit switch anyway */
57       } /* end switch */
58
59    } /* end while */
60
61    /* output summary of results */
62    printf( "\nTotals for each letter grade are:\n" );
63    printf( "A: %d\n", aCount ); /* display number of A grades */
64    printf( "B: %d\n", bCount ); /* display number of B grades */
65    printf( "C: %d\n", cCount ); /* display number of C grades */
66    printf( "D: %d\n", dCount ); /* display number of D grades */
67    printf( "F: %d\n", fCount ); /* display number of F grades */
68
69    return 0; /* indicate program ended successfully */
70
71 } /* end function main */
```

Παράδειγμα (4/4)

```
Enter the letter grades.
Enter the EOF character to end input.
a
b
c
C
A
d
f
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
b
^Z

Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1
```

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Η εντολή break

- Προκαλεί άμεση έξοδο από μια εντολή `while`, `for`, `do…while` ή `switch`
- Η εκτέλεση του προγράμματος συνεχίζει με την αμέσως επόμενη εντολή που ακολουθεί τον κορμό μιας από τις παραπάνω εντολές
- Η εντολή `break` χρησιμοποιείται συνήθως
  - Για την πρόωρη έξοδο από ένα βρόχο
  - Για την παράλειψη του υπόλοιπου τμήματος μιας εντολής `switch`

Παράδειγμα

```c
/* Using the break statement in a for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x; /* counter */

   /* loop 10 times */
   for ( x = 1; x <= 10; x++ ) {

      /* if x is 5, terminate loop */
      if ( x == 5 ) {
         break; /* break loop only if x is 5 */
      } /* end if */

      printf( "%d ", x ); /* display value of x */
   } /* end for */

   printf( "\nBroke out of loop at x == %d\n", x );

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
1 2 3 4
Broke out of loop at x == 5
```

Η εντολή continue

- Παραλείπει τις υπόλοιπες εντολές του κορμού μιας εντολής while, for ή do…while
  - Συνεχίζει με την επόμενη επανάληψη του βρόχου
- while και do…while
  - Ο έλεγχος της συνθήκης επανάληψης γίνεται αμέσως μετά την εκτέλεση της εντολής continue
- for
  - Εκτελείται η αύξηση / μείωση και στη συνέχεια γίνεται ο έλεγχος της συνθήκης επανάληψης

Παράδειγμα

```c
/* Using the continue statement in a for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x; /* counter */

   /* loop 10 times */
   for ( x = 1; x <= 10; x++ ) {

      /* if x is 5, continue with next iteration of loop */
      if ( x == 5 ) {
         continue; /* skip remaining code in loop body */
      } /* end if */

      printf( "%d ", x ); /* display value of x */
   } /* end for */

   printf( "\nUsed continue to skip printing the value 5\n" );

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Η εντολή goto (1/2)

- Μπορεί να εκτελέσει μια "μετάβαση" από ένα σημείο του προγράμματος σε ένα άλλο
  - Η λειτουργία της βασίζεται στην ύπαρξη μιας ετικέτας, που είναι ένα όνομα που ακολουθείται από άνω κάτω τελεία
- Γενική μορφή

```
goto <ετικέτα>;
```

  - Η μετάβαση γίνεται στη γραμμή που έχει την εξής μορφή

```
<ετικέτα>: πρόταση
```

Η εντολή goto (2/2)

- Παράδειγμα

```
goto mylabel;
printf("This will not print.");
mylabel: printf("This will print.");
```

- Καλύτερα να μη χρησιμοποιείται
  - Καταστρέφει τη δόμηση του κώδικα - μπορεί να καταστήσει αδύνατη τη μελλοντική ανάγνωση του προγράμματος
  - Δεν είναι απαραίτητη - οποιοδήποτε πρόγραμμα μπορεί να γραφτεί χωρίς τη χρήση της

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Εισαγωγή

- “Διαίρει και βασίλευε”
  - Κατασκευή προγράμματος από μικρότερες μονάδες
    - Ευκολότερη διαχείριση κάθε μονάδας ξεχωριστά
- Μονάδες Προγράμματος (Program Modules) στη C
  - Τα προγράμματα συνδυάζουν συναρτήσεις που ορίζονται από το χρήστη με συναρτήσεις βιβλιοθήκης
    - Η standard βιβλιοθήκη της C έχει πολλές συναρτήσεις
- Κλήση συναρτήσεων
  - Όνομα συνάρτησης και παράμετροι (δεδομένα)
  - Η συνάρτηση πραγματοποιεί κάποια λειτουργία
  - Επιστρέφει αποτελέσματα

Μαθηματικές συναρτήσεις (1/2)

- Πραγματοποιούν κοινούς (βασικούς) μαθηματικούς υπολογισμούς

```
#include <math.h>
```

- Γενική μορφή κλήσης συναρτήσεων

```
Όνομα_συνάρτησης ( παράμετρος );
```

  - Αν υπάρχουν πολλές παράμετροι διαχωρίζονται με κόμμα (,)
  - Παράδειγμα: `printf( "%.2f", sqrt( 900.0 ) );`
    - Καλεί τη συνάρτηση sqrt, που επιστρέφει την τετραγωνική ρίζα της παραμέτρου
    - Όλες οι μαθηματικές συναρτήσεις επιστρέφουν τιμές τύπου double
  - Οι παράμετροι μπορεί να είναι σταθερές, μεταβλητές ή εκφράσεις

Μαθηματικές συναρτήσεις (2/2)

Συνάρτηση	Περιγραφή	Παράδειγμα
`sqrt( x )`	Τετραγωνική ρίζα του x	`sqrt( 900.0 ) "επιστρέφει" 30.0` `sqrt( 9.0 ) "επιστρέφει" 3.0`
`exp( x )`	Εκθετική συνάρτηση e^x	`exp( 1.0 ) "επιστρέφει" 2.718282` `exp( 2.0 ) "επιστρέφει" 7.389056`
`log( x )`	Λογάριθμος του x με βάση το e ($ln x$)	`log( 2.718282 ) "επιστρέφει" 1.0` `log( 7.389056 ) "επιστρέφει" 2.0`

`log10( x )`	Λογάριθμος του x (με βάση το 10)	`log10( 1.0 ) "επιστρέφει" 0.0`
		`log10( 10.0 ) "επιστρέφει" 1.0`
		`log10( 100.0 ) "επιστρέφει" 2.0`

`fabs( x )`	Απόλυτη τιμή του x	`fabs( 5.0 ) "επιστρέφει" 5.0`
		`fabs( 0.0 ) "επιστρέφει" 0.0`
		`fabs( -5.0 ) "επιστρέφει" 5.0`

`ceil( x )`	Στρογγυλοποιεί το x στο μικρότερο ακέραιο (μεγαλύτερο ή ίσο του x)	`ceil( 9.2 ) "επιστρέφει" 10.0` `ceil( -9.8 ) "επιστρέφει" -9.0`
`floor( x )`	Στρογγυλοποιεί το x στο μεγαλύτερο ακέραιο (μικρότερο ή ίσο του x)	`floor( 9.2 ) "επιστρέφει" 9.0` `floor( -9.8 ) "επιστρέφει" -10.0`
`pow( x, y )`	x εις την y (x^y)	`pow( 2, 7 ) "επιστρέφει" 128.0` `pow( 9, .5 ) "επιστρέφει" 3.0`
`fmod( x, y )`	υπόλοιπο της διαίρεσης x/y σαν αριθμό κινητής υποδιαστολής	`fmod( 13.657, 2.333 ) "επιστρέφει" 1.992`
`sin( x )`	ημίτονο του x (x σε ακτίνια)	`sin( 0.0 ) "επιστρέφει" 0.0`
`cos( x )`	συνημίτονο του x (x σε ακτίνια)	`cos( 0.0 ) "επιστρέφει" 1.0`
`tan( x )`	εφαπτομένη του x (x σε ακτίνια)	`tan( 0.0 ) "επιστρέφει" 0.0`

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Πρωτότυπα συναρτήσεων (1/3)

- Πρωτότυπο συνάρτησης
  - Όνομα συνάρτησης
  - Παράμετροι - τι εισάγεται στη συνάρτηση
  - Επιστρεφόμενος τύπος – τύπος δεδομένων που επιστρέφει η συνάρτηση (default `int`)
  - Χρησιμοποιείται για την επικύρωση συναρτήσεων
  - Το πρωτότυπο χρειάζεται μόνο αν ο ορισμός (υλοποίηση) της συνάρτησης γίνεται μετά τη χρησιμοποίηση της συνάρτησης στο πρόγραμμα

Πρωτότυπα συναρτήσεων (2/3)

- Πρωτότυπο συνάρτησης
  - Γενική μορφή πρωτότυπου

```
επιστρεφόμενος-τύπος όνομα-συνάρτησης(τύπος όνομα-παραμέτρου1,
                                       τύπος όνομα-παραμέτρου2,
                                                    .
                                                    .
                                                    .
                                       τύπος όνομα-παραμέτρουN,
                                                    );
```

  - Παράδειγμα

```
int maximum( int x, int y, int z );
```

    - Παίρνει 3 ακέραιους (int)
    - Επιστρέφει αποτέλεσμα τύπου int

Παράδειγμα (1/2)

```
1  /* Creating and using a programmer-defined function */
2  #include <stdio.h>
3
4  int square( int y );   /* function prototype */
5
6  /* function main begins program execution */
7  int main()
8  {
9     int x; /* counter */
10
11    /* loop 10 times and calculate and output square of x each time */
12    for ( x = 1; x <= 10; x++ ) {
13       printf( "%d ", square( x ) ); /* function call */
14    } /* end for */
15
16    printf( "\n" );
17
18    return 0; /* indicates successful termination */
19
20 } /* end main */
21
```

Παράδειγμα (2/2)

```
22 /* square function definition returns square of an integer */
23 int square( int y ) /* y is a copy of argument to function */
24 {
25    return y * y; /* returns square of y as an int */
26
27 } /* end function square */
```

```
1  4  9  16  25  36  49  64  81  100
```

Πρωτότυπα συναρτήσεων (3/3)

- Σημαντικό χαρακτηριστικό
  - Μετατροπή των παραμέτρων στον κατάλληλο τύπο
    - Τιμές των παραμέτρων που δεν αντιστοιχούν στους τύπους των παραμέτρων στα πρωτότυπα των συναρτήσεων μετατρέπονται στο σωστό τύπο προτού κληθεί η συνάρτηση
    - Αυτές οι μετατροπές μπορούν να οδηγήσουν σε λανθασμένα αποτελέσματα αν δεν ακολουθούνται ορισμένοι κανόνες
- Κανόνες προαγωγής (promotion rules)
  - Καθορίζουν τον τρόπο με τον οποίο μπορούν οι τύποι δεδομένων να μετατραπούν σε άλλους τύπους χωρίς την απώλεια δεδομένων
    - Π.χ. Η μετατροπή ενός int σε double δεν παρουσιάζει κανένα πρόβλημα. Αντίστροφα, η μετατροπή double σε int προκαλεί αποκοπή του δεκαδικού μέρους (του μέρους του αριθμού που ακολουθεί την υποδιαστολή)

Τροποποιητές τύπου

- `long`
  - Ορισμός ακεραίων τιμών με πεδίο τιμών συχνά μεγαλύτερο από το πεδίο τιμών του τύπου int
- `short`
  - Ορισμός ακεραίων τιμών με πεδίο τιμών συχνά μικρότερο από το πεδίο τιμών του τύπου int
- `signed`
  - Ορίζει ότι ο αντίστοιχος ακέραιος τύπος αφορά θετικές και αρνητικές τιμές
- `unsigned`
  - Εφαρμόζεται πριν από τον ορισμό ακεραίων τύπων (char, short, int, long) για να ορίσει ότι η τύπος αφορά μόνο θετικές τιμές (διπλασιασμός πεδίου τιμών). Π.χ. unsigned int i;
- Παρατηρήσεις
  - Ο τύπος `int` είναι αυτός που προσφέρει κατά βάση η αρχιτεκτονική του επεξεργαστή για υπολογισμούς ακεραίων
  - Τύπους `short` και `long` χρησιμοποιούμε μόνο σε ειδικές περιπτώσεις (π.χ. χρήση έτοιμων συναρτήσεων που απαιτούν τέτοιους τύπους ή προγράμματα συστήματος)
  - Οι τύποι `short`, `int` και `long` είναι πάντα `signed`. Ο τύπος `char` μπορεί να είναι `signed` ή `unsigned` ανάλογα με την αρχιτεκτονική του υπολογιστή και τον μεταγλωττιστή.

Κανόνες προαγωγής

- Promotion Rules
- Εφαρμόζονται αυτόματα σε εκφράσεις που περιέχουν τιμές δύο ή περισσότερων τύπων (μικτού τύπου εκφράσεις)
  - Ο τύπος κάθε μεταβλητής σε μια μικτού τύπου έκφραση "προάγεται" στον "υψηλότερο" τύπο στην έκφραση
    - Ο πίνακας απεικονίζει τους τύπους από τον "υψηλότερο" προς το "χαμηλότερο"

Τύπος δεδομένων	Χρήση στην printf	Χρήση στην scanf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
short	%hd	%hd
char	%c	%c

Ορισμός συναρτήσεων (1/2)

- Γενική μορφή ορισμού συναρτήσεων

```
επιστρεφόμενος-τύπος όνομα-συνάρτησης (λίστα-παραμέτρων)
{
   δηλώσεις και εντολές
}
```

  - Όνομα-συνάρτησης: οποιοδήποτε έγκυρο αναγνωριστικό
  - Επιστρεφόμενος-τύπος: τύπος δεδομένων του αποτελέσματος (default int)
    - void – υποδεικνύει ότι η συνάρτηση δεν επιστρέφει τίποτα
  - Λίστα-παραμέτρων: λίστα παραμέτρων που χωρίζονται με κόμμα (,)
    - Για κάθε παράμετρο θα πρέπει να δηλώνεται ο τύπος της διαφορετικά θεωρείται ότι είναι τύπου int

Ορισμός συναρτήσεων (2/2)

- Γενική μορφή ορισμού συναρτήσεων

```
επιστρεφόμενος-τύπος όνομα-συνάρτησης (λίστα-παραμέτρων)
{
   δηλώσεις και εντολές
}
```

  - Δηλώσεις και εντολές: σώμα συνάρτησης (block)
    - Επιτρέπεται η δήλωση μεταβλητών μέσα στο σώμα συναρτήσεων
    - Δεν επιτρέπεται η δήλωση συναρτήσεων σε άλλες συναρτήσεις
  - Επιστροφή ελέγχου
    - Αν επιστρέφεται κάποιο αποτέλεσμα
      - `return έκφραση;`
      - Αποτίμηση της έκφρασης και αντιγραφή της τιμής στο σημείο κλήσης
    - Αν δεν επιστρέφεται τίποτα
      - Συνέχιση της εκτέλεσης του προγράμματος με την εντολή που ακολουθεί την κλήση της συνάρτησης

Παράδειγμα (1/2)

```
1  /* Finding the maximum of three integers */
2  #include <stdio.h>
3
4  int maximum( int x, int y, int z ); /* function prototype */
5
6  /* function main begins program execution */
7  int main()
8  {
9     int number1; /* first integer */
10    int number2; /* second integer */
11    int number3; /* third integer */
12
13    printf( "Enter three integers: " );
14    scanf( "%d%d%d", &number1, &number2, &number3 );
15
16    /* number1, number2 and number3 are arguments
17       to the maximum function call */
18    printf( "Maximum is: %d\n", maximum( number1, number2, number3 ) );
19
20    return 0; /* indicates successful termination */
21
22 } /* end main */
23
```

Παράδειγμα (2/2)

```
24 /* Function maximum definition */
25 /* x, y and z are parameters */
26 int maximum( int x, int y, int z )
27 {
28    int max = x;     /* assume x is largest */
29
30    if ( y > max ) { /* if y is larger than max, assign y to max */
31       max = y;
32    } /* end if */
33
34    if ( z > max ) { /* if z is larger than max, assign z to max */
35       max = z;
36    } /* end if */
37
38    return max;      /* max is largest value */
39
40 } /* end function maximum */
```

```
Enter three integers: 22 85 17
Maximum is: 85
Enter three integers: 85 22 17
Maximum is: 85
Enter three integers: 22 17 85
Maximum is: 85
```

Κλήση συναρτήσεων: Με τιμή και με αναφορά

- Κλήση με τιμή (Call by value)
  - Το “αντίγραφο” μιας παραμέτρου περνάει στη συνάρτηση
  - Οι αλλαγές που γίνονται στη συνάρτηση δεν επηρεάζουν την αρχική παράμετρο
  - Χρησιμοποιείται όταν η συνάρτηση δεν χρειάζεται να τροποποιήσει την παράμετρο
    - Αποφυγή συμπτωματικών αλλαγών
- Κλήση με αναφορά (Call by reference)
  - Περνάει την αρχική παράμετρο
  - Οι αλλαγές που γίνονται στη συνάρτηση επηρεάζουν την αρχική παράμετρο
- Προς το παρόν εστιάζουμε στην κλήση συναρτήσεων με τιμή

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Αρχεία επικεφαλίδας (1/2)

- Αρχεία επικεφαλίδας (Header Files)
  - Περιέχουν πρωτότυπα συναρτήσεων για συναρτήσεις βιβλιοθήκης
    - Π.χ. `<stdlib.h>` , `<math.h>`, κλπ
  - "Φόρτωση" με: `#include <filename>`
    - Π.χ. `#include <math.h>`
- Κατασκευή δικών μας αρχείων επικεφαλίδας
  - Δημιουργία αρχείου με συναρτήσεις
  - Αποθήκευση του αρχείου με όνομα `filename.h`
  - Χρήση σε άλλα αρχεία με την εντολή `#include "filename.h"`
  - Επιτρέπει την επαναχρησιμοποίηση συναρτήσεων

Αρχεία επικεφαλίδας (2/2)

Επικεφαλίδα standard βιβλιοθήκης (Standard library header)	Επεξήγηση
<ctype.h>	Περιέχει πρωτότυπα συναρτήσεων που ελέγχουν ορισμένες ιδιότητες χαρακτήρων, καθώς και πρωτότυπα συναρτήσεων για συναρτήσεις που μπορούν να χρησιμοποιηθούν για την μετατροπή πεζών (μικρών) γραμμάτων σε κεφαλαία γράμματα και αντίστροφα.
<math.h>	Περιέχει πρωτότυπα συναρτήσεων για μαθηματικές συναρτήσεις βιβλιοθήκης.
<setjmp.h>	Περιέχει πρωτότυπα συναρτήσεων για συναρτήσεις που επιτρέπουν την παράκαμψη της συνηθισμένης ακολουθίας κλήσης συναρτήσεων κι επιστροφής.
<stdio.h>	Περιέχει πρωτότυπα συναρτήσεων για standard συναρτήσεις εισόδου / εξόδου καθώς και πληροφορίες που χρησιμοποιούνται από αυτές.
<stdlib.h>	Περιέχει πρωτότυπα συναρτήσεων για μετατροπή αριθμών από σε κείμενο και κειμένου σε αριθμούς, κατανομή μνήμης (memory allocation), τυχαίους αριθμούς και άλλες γενικής χρήσης συναρτήσεις.
<string.h>	Περιέχει πρωτότυπα συναρτήσεων για την επεξεργασία αλφαριθμητικών.
<time.h>	Περιέχει πρωτότυπα συναρτήσεων και τύπους για το χειρισμό ώρας και ημερομηνίας.

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Δημιουργία τυχαίων αριθμών (1/2)

- Συνάρτηση `rand`
  - Φόρτωση της `<stdlib.h>`
  - Επιστρέφει έναν ”τυχαίο" αριθμό μεταξύ 0 και RAND_MAX (τουλάχιστον 32767)

```
i = rand();
```

  - Ψευδοτυχαίοι αριθμοί
    - Προκαθορισμένη ακολουθία “τυχαίων” αριθμών
    - Ίδια ακολουθία για κάθε κλήση συνάρτησης
  - Για την παραγωγή ενός τυχαίου αριθμού μεταξύ 1 και n (κλιμάκωση)

```
1 + ( rand() % n )
```

    - `rand() % n` επιστρέφει έναν αριθμό μεταξύ 0 και n - 1
    - Πρόσθεση του 1 προκειμένου να είναι αριθμός μεταξύ 1 και n
      - Π.χ. αριθμός μεταξύ 1 και 6: `1 + ( rand() % 6)`

Παράδειγμα 1

```
1  /* Shifted, scaled integers produced by 1 + rand() % 6 */
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int i; /* counter */
9
10    /* loop 20 times */
11    for ( i = 1; i <= 20; i++ ) {
12
13       /* pick random number from 1 to 6 and output it */
14       printf( "%10d", 1 + ( rand() % 6 ) );
15
16       /* if counter is divisible by 5, begin new line of output */
17       if ( i % 5 == 0 ) {
18          printf( "\n" );
19       } /* end if */
20
21    } /* end for */
22
22    return 0; /* indicates successful termination */
24
25 } /* end main */
```

```
6         6         5         5         6
5         1         1         5         3
6         6         2         4         2
6         2         3         4         1
```

Δημιουργία τυχαίων αριθμών (2/2)

- Συνάρτηση `srand`
  - Φόρτωση της `<stdlib.h>`
  - Η συνάρτηση `srand(seed)` χρησιμοποιεί το seed ως φύτρο για νέα ακολουθία ψευδοτυχαίων αριθμών
    - Εάν παραλειφθεί η συνάρτηση το αρχικό φύτρο είναι πάντοτε το 1
    - Η srand() εκτελείται πάντοτε μία φορά στην αρχή του προγράμματος και αρχικοποιεί τη γεννήτρια των τυχαίων αριθμών
    - Εάν θέλουμε να ξανατρέξουμε τον πρόγραμμά μας και να δημιουργήσουμε νέους τυχαίους αριθμούς πρέπει να αλλάξουμε τον ακέραιο seed
  - `srand( time( NULL ) ); / *load <time.h> */`
    - `time( NULL )`
      - Επιστρέφει το χρόνο σε δευτερόλεπτα στον οποίο μεταγλωττίστηκε (έγινε compile) το πρόγραμμα
      - “Τυχαιοποιεί" το φύτρο (seed)

Παράδειγμα 2 (1/2)

```
1  /* Randomizing die-rolling program */
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int i;         /* counter */
9     unsigned seed; /* number used to seed random number generator */
10
11    printf( "Enter seed: " );
12    scanf( "%u", &seed );
13
14    srand( seed ); /* seed random number generator */
15
16    /* loop 10 times */
17    for ( i = 1; i <= 10; i++ ) {
18
19       /* pick a random number from 1 to 6 and output it */
20       printf( "%10d", 1 + ( rand() % 6 ) );
21
```

Παράδειγμα 2 (2/2)

```
22          /* if counter is divisible by 5, begin a new line of output */
23          if ( i % 5 == 0 ) {
24             printf( "\n" );
25          } /* end if */
26
27       } /* end for */
28
29       return 0; /* indicates successful termination */
30
31    } /* end main */
```

```
Enter seed: 67
        6       1       4       6       2
        1       6       1       6       4
Enter seed: 867
        2       4       6       1       6
        1       1       3       6       2
Enter seed: 67
        6       1       4       6       2
        1       6       1       6       4
```

Παράδειγμα 3 (1/3)

- Ρίψη ζαριού 6000 φορές

```
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int frequency1 = 0; /* rolled 1 counter */
9     int frequency2 = 0; /* rolled 2 counter */
10    int frequency3 = 0; /* rolled 3 counter */
11    int frequency4 = 0; /* rolled 4 counter */
12    int frequency5 = 0; /* rolled 5 counter */
13    int frequency6 = 0; /* rolled 6 counter */
14
15    int roll; /* roll counter */
16    int face; /* represents one roll of the die, value 1 to 6 */
17
18    /* loop 6000 times and summarize results */
19    for ( roll = 1; roll <= 6000; roll++ ) {
20       face = 1 + rand() % 6; /* random number from 1 to 6 */
21
```

Παράδειγμα 3 (2/3)

```
22       /* determine face value and increment appropriate counter */
23       switch ( face ) {
24
25          case 1:          /* rolled 1 */
26             ++frequency1;
27             break;
28
29          case 2:          /* rolled 2 */
30             ++frequency2;
31             break;
32
33          case 3:          /* rolled 3 */
34             ++frequency3;
35             break;
36
37          case 4:          /* rolled 4 */
38             ++frequency4;
39             break;
40
41          case 5:          /* rolled 5 */
42             ++frequency5;
43             break;
44
```

Παράδειγμα 3 (3/3)

```
45          case 6:          /* rolled 6 */
46             ++frequency6;
47             break;
48       } /* end switch */
49
50    } /* end for */
51
52    /* display results in tabular format */
53    printf( "%s%13s\n", "Face", "Frequency" );
54    printf( "   1%13d\n", frequency1 );
55    printf( "   2%13d\n", frequency2 );
56    printf( "   3%13d\n", frequency3 );
57    printf( "   4%13d\n", frequency4 );
58    printf( "   5%13d\n", frequency5 );
59    printf( "   6%13d\n", frequency6 );
60
61    return 0; /* indicates successful termination */
62
63 } /* end main */
```

```
Face    Frequency
   1         1003
   2         1017
   3          983
   4          994
   5         1004
   6          999
```

Παράδειγμα: Ένα παιχνίδι τύχης

- Προσομοιωτής ζαριών
- Κανόνες
  - “Ρίχνουμε” δύο ζάρια
    - Ο παίκτης που φέρνει 7 ή 11 με την πρώτη ζαριά κερδίζει
    - Ο παίκτης που φέρνει 2, 3 ή 12 με την πρώτη ζαριά χάνει
    - 4, 5, 6, 8, 9, 10 - ο παίκτης παίρνει “πόντο” που αντιστοιχεί στην τιμή
  - Για να κερδίσει ένας παίκτης πρέπει να φέρει τον πόντο του πριν να φέρει 7

Παράδειγμα (1/5)

```
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <time.h> /* contains prototype for function time */
5
6  /* enumeration constants represent game status */
7  enum Status { CONTINUE, WON, LOST };
8
9  int rollDice( void ); /* function prototype */
10
11 /* function main begins program execution */
12 int main()
13 {
14    int sum;        /* sum of rolled dice */
15    int myPoint;    /* point earned */
16
17    enum Status gameStatus; /* can contain CONTINUE, WON, or LOST */
18
19    /* randomize random number generator using current time */
20    srand( time( NULL ) );
21
22    sum = rollDice(  ); /* first roll of the dice */
23
```

Παράδειγμα (2/5)

```
24    /* determine game status based on sum of dice */
25    switch( sum ) {
26
27       /* win on first roll */
28       case 7:
29       case 11:
30          gameStatus = WON;
31          break;
32
33       /* lose on first roll */
34       case 2:
35       case 3:
36       case 12:
37          gameStatus = LOST;
38          break;
39
40       /* remember point */
41       default:
42          gameStatus = CONTINUE;
43          myPoint = sum;
44          printf( "Point is %d\n", myPoint );
45          break; /* optional */
46    } /* end switch */
```

Παράδειγμα (3/5)

```
48    /* while game not complete */
49    while ( gameStatus == CONTINUE ) {
50       sum = rollDice(  ); /* roll dice again */
51
52       /* determine game status */
53       if ( sum == myPoint ) { /* win by making point */
54          gameStatus = WON;
55       } /* end if */
56       else {
57
58          if ( sum == 7 ) { /* lose by rolling 7 */
59             gameStatus = LOST;
60          } /* end if */
61
62       } /* end else */
63
64    } /* end while */
65
66    /* display won or lost message */
67    if ( gameStatus == WON ) {
68       printf( "Player wins\n" );
69    } /* end if */
70    else {
71       printf( "Player loses\n" );
72    } /* end else */
```

Παράδειγμα (4/5)

```
74    return 0; /* indicates successful termination */
75
76 } /* end main */
77
78 /* roll dice, calculate sum and display results */
79 int rollDice( void )
80 {
81    int die1;    /* first die */
82    int die2;    /* second die */
83    int workSum; /* sum of dice */
84
85    die1 = 1 + ( rand() % 6 ); /* pick random die1 value */
86    die2 = 1 + ( rand() % 6 ); /* pick random die2 value */
87    workSum = die1 + die2;     /* sum die1 and die2 */
88
89    /* display results of this roll */
90    printf( "Player rolled %d + %d = %d\n", die1, die2, workSum );
91
92    return workSum; /* return sum of dice */
93
94 } /* end function rollRice */
```

Παράδειγμα (5/5)

```
Player rolled 5 + 6 = 11
Player wins

Player rolled 4 + 1 = 5
Point is 5
Player rolled 6 + 2 = 8
Player rolled 2 + 1 = 3
Player rolled 3 + 2 = 5
Player wins

Player rolled 1 + 1 = 2
Player loses

Player rolled 1 + 4 = 5
Point is 5
Player rolled 3 + 4 = 7
Player loses
```

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Κατηγορίες αποθήκευσης (1/2)

- Προσδιοριστές κατηγοριών αποθήκευσης
  - Διάρκεια αποθήκευσης – για πόσο ένα αντικείμενο (μια μεταβλητή) παραμένει στη μνήμη
  - Εμβέλεια – από ποια σημεία του προγράμματος μπορεί να γίνει αναφορά σε ένα αντικείμενο
  - Σύνδεση – προσδιορίζει τα αρχεία στα οποία είναι γνωστό ένα αναγνωριστικό
- Αυτόματη αποθήκευση
  - Το αντικείμενο δημιουργείται και καταστρέφεται μέσα στο block του
  - `auto`: default για τοπικές μεταβλητές

```
auto double x, y;
```

  - `register`: προσπαθεί να βάλει τις μεταβλητές σε καταχωρητές υψηλής ταχύτητας
    - Μπορεί να χρησιμοποιηθεί μόνο για αυτόματες μεταβλητές

```
register int counter = 1;
```

Κατηγορίες αποθήκευσης (2/2)

- Στατική αποθήκευση
  - Οι μεταβλητές υπάρχουν για όλη τη διάρκεια της εκτέλεσης του προγράμματος
  - Default τιμή του 0
  - `static:` οι τοπικές μεταβλητές που δηλώνονται σε συναρτήσεις
    - Κρατάνε την τιμή τους μετά το τέλος των συναρτήσεων (περιέχουν την τελευταία τιμή από την προηγούμενη κλήση)
    - Γνωστές μόνο μέσα στη δική τους συνάρτηση
  - `extern:` default για ολικές μεταβλητές (global variables) και συναρτήσεις
    - Γνωστές σε κάθε συνάρτηση

Παράδειγμα

```
main ()
{
   increment();
   increment();
   increment();
}

void increment (void)
{
   int j=2;
   static int k=2;

   printf("j:%d\t k:%d\n", j++, k++);
}
```

```
j:2       k:2
j:2       k:3
j:2       k:4
```

Διάρκεια μεταβλητών (1/2)

- Ορίζει το χρόνο κατά τον οποίο το όνομα μιας μεταβλητής είναι συνδεδεμένο με τη θέση μνήμης που περιέχει την τιμή της μεταβλητής
- Χρόνος δέσμευσης και αποδέσμευσης
- Καθολικές μεταβλητές
  - Δεσμεύεται χώρος με την έναρξη της εκτέλεσης του προγράμματος
  - Η μεταβλητή συσχετίζεται με την ίδια θέση μνήμης μέχρι το τέλος του προγράμματος (πλήρους διάρκειας)
- Τοπικές μεταβλητές
  - Δεσμεύεται χώρος με την είσοδο στο μπλοκ όπου είναι δηλωμένη η μεταβλητή (αποδεσμεύεται κατά την έξοδο από το μπλοκ)
  - Μια τοπική μεταβλητή δεν διατηρεί την τιμή της από τη μία κλήση της συνάρτησης στην επόμενη (περιορισμένης διάρκειας)
  - Η διάρκεια μιας τοπικής μεταβλητής μπορεί να μετατραπεί από περιορισμένη σε πλήρη χρησιμοποιώντας τη λέξη κλειδί `static` στη δήλωση της μεταβλητής

Διάρκεια μεταβλητών (2/2)

- Παράδειγμα

```
static int num;
void func(int x){
    static int count=0;
    int num=100;
    ...
}
```

- Η `static` στη δήλωση της καθολικής μεταβλητής `num` περιορίζει την ορατότητά της μόνο στο αρχείο που δηλώνεται
- Αντίθετα, η `static` στη δήλωση της τοπικής μεταβλητής `count` ορίζει γι' αυτήν διάρκεια προγράμματος
- Η `count` ως τοπική μεταβλητή πλήρους διάρκειας αρχικοποιείται μια φορά με την είσοδο στο πρόγραμμα
- Αντίθετα, η `num` ως τοπική μεταβλητή περιορισμένης διάρκειας αρχικοποιείται σε κάθε ενεργοποίηση της `func`

Σημερινή διάλεξη

- Σύνοψη 3ης διάλεξης
- Η εντολή πολλαπλής επιλογής switch
- Η εντολή break
- Η εντολή goto
- Συναρτήσεις
  - Εισαγωγή
  - Μαθηματικές συναρτήσεις
  - Πρωτότυπα συναρτήσεων
  - Ορισμός συναρτήσεων
  - Αρχεία επικεφαλίδας
  - Κλήση συναρτήσεων
  - Δημιουργία τυχαίων αριθμών
  - Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας

Κανόνες εμβέλειας (1/2)

- Scope rules
- Προσδιορίζουν το τμήμα του πηγαίου κώδικα στο οποίο ένα όνομα είναι «ενεργό» ή «ορατό»
- Εμβέλεια προγράμματος
  - Μεταβλητές με τέτοια εμβέλεια λέγονται γενικές ή καθολικές (global)
  - Είναι ορατές από όλες τις συναρτήσεις που απαρτίζουν το πρόγραμμα (έστω κι αν βρίσκονται σε διαφορετικά αρχεία πηγαίου κώδικα)
  - Μεταβλητή που δηλώνεται έξω από μπλοκ και χωρίς τη λέξη κλειδί `static` έχει εμβέλεια προγράμματος
- Εμβέλεια αρχείου
  - Τέτοιες μεταβλητές είναι ορατές μόνο στο αρχείο που δηλώνονται (από το σημείο δήλωσής τους και κάτω)
  - Μεταβλητή που δηλώνεται έξω από μπλοκ, με τη λέξη κλειδί `static` (πριν από τον τύπο της) έχει εμβέλεια αρχείου

Κανόνες εμβέλειας (2/2)

- Εμβέλεια συνάρτησης
  - Προσδιορίζει την ορατότητα ενός ονόματος σε όλο το σώμα της συνάρτησης
  - Τέτοια εμβέλεια έχουν μόνο οι `goto` ετικέτες
- Εμβέλεια μπλοκ
  - Προσδιορίζει την ορατότητα ενός ονόματος από το σημείο δήλωσής του μέχρι το τέλος του μπλοκ στο οποίο δηλώνεται
  - Τέτοια εμβέλεια έχουν και τα τυπικά ορίσματα των συναρτήσεων (μπλοκ είναι η σύνθετη πρόταση αλλά και το σώμα μιας συνάρτησης)

Παράδειγμα (1/3)

```
1  /* A scoping example */
2  #include <stdio.h>
3
4  void useLocal( void );       /* function prototype */
5  void useStaticLocal( void ); /* function prototype */
6  void useGlobal( void );      /* function prototype */
7
8  int x = 1; /* global variable */
9
10 /* function main begins program execution */
11 int main()
12 {
13    int x = 5; /* local variable to main */
14
15    printf("local x in outer scope of main is %d\n", x );
16
17    { /* start new scope */
18       int x = 7; /* local variable to new scope */
19
20       printf( "local x in inner scope of main is %d\n", x );
21    } /* end new scope */
22
23    printf( "local x in outer scope of main is %d\n", x );
```

Παράδειγμα (2/3)

```
25    useLocal();       /* useLocal has automatic local x */
26    useStaticLocal(); /* useStaticLocal has static local x */
27    useGlobal();      /* useGlobal uses global x */
28    useLocal();       /* useLocal reinitializes automatic local x */
29    useStaticLocal(); /* static local x retains its prior value */
30    useGlobal();      /* global x also retains its value */
31
32    printf( "local x in main is %d\n", x );
33
34    return 0; /* indicates successful termination */
35
36 } /* end main */
37
38 /* useLocal reinitializes local variable x during each call */
39 void useLocal( void )
40 {
41    int x = 25;  /* initialized each time useLocal is called */
42
43    printf( "\nlocal x in a is %d after entering a\n", x );
44    x++;
45    printf( "local x in a is %d before exiting a\n", x );
46 } /* end function useLocal */
```

Παράδειγμα (3/3)

```
48 /* useStaticLocal initializes static local variable x only the first time
49    the function is called; value of x is saved between calls to this
50    function */
51 void useStaticLocal( void )
52 {
53    /* initialized only first time useStaticLocal is called */
54    static int x = 50;
55
56    printf( "\nlocal static x is %d on entering b\n", x );
57    x++;
58    printf( "local static x is %d on exiting b\n", x );
59 } /* end function useStaticLocal */
60
61 /* function useGlobal modifies global variable x during each call */
62 void useGlobal( void )
63 {
64    printf( "\nglobal x is %d on entering c\n", x );
65    x *= 10;
66    printf( "global x is %d on exiting c\n", x );
67 } /* end function useGlobal */
```

Αποτέλεσμα προγράμματος

```
local x in outer scope of main is 5
local x in inner scope of main is 7
local x in outer scope of main is 5

local x in a is 25 after entering a
local x in a is 26 before exiting a

local static x is 50 on entering b
local static x is 51 on exiting b

global x is 1 on entering c
global x is 10 on exiting c

local x in a is 25 after entering a
local x in a is 26 before exiting a

local static x is 51 on entering b
local static x is 52 on exiting b

global x is 10 on entering c
global x is 100 on exiting c
local x in main is 5
```

Ερωτήσεις…