Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

Γλώσσα Προγραμματισμού C

5η Διάλεξη

Δημοσθένης Κυριαζής

Τρίτη 30 Οκτωβρίου 2018

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

Η εντολή πολλαπλής επιλογής switch

- 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 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
```

Πρωτότυπα συναρτήσεων

- 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

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

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

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`

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
```

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

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
- 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
```

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

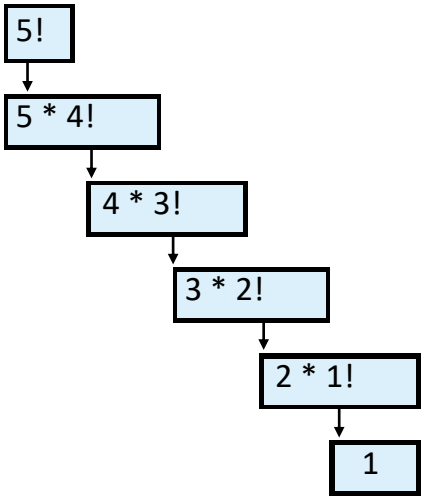
Αναδρομή (1/3)

- Αναδρομικές συναρτήσεις
  - Συναρτήσεις που καλούν τον εαυτό τους
    - Κάθε αναδρομική συνάρτηση μπορεί να υλοποιηθεί με επανάληψη
  - Υπάρχει δυνατότητα επίλυση βασικής περίπτωσης (base case)
  - Διαίρεση προβλήματος
    - Τι μπορεί να κάνει
    - Τι δεν μπορεί να κάνει
      - Αυτό που δεν μπορεί να κάνει μοιάζει με το αρχικό πρόβλημα
      - Η συνάρτηση ξεκινάει ένα νέο αντίγραφο του εαυτού της (βήμα αναδρομής) για να επιλύσει αυτό που δεν μπορεί να κάνει
  - Τελικά επιλύεται η βασική περίπτωση
    - Συνδέεται με τις υπόλοιπες περιπτώσεις, και σταδιακά συντίθεται η συνολική - τελική λύση

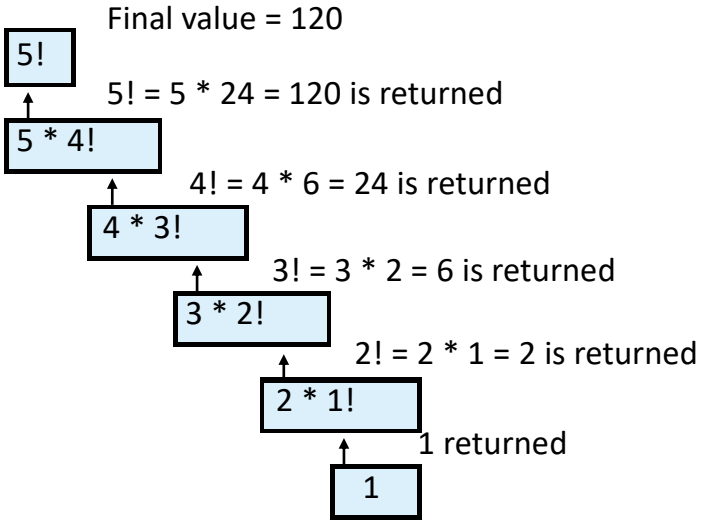
Αναδρομή (2/3)

- Παράδειγμα: παραγοντικά
  - 5! = 5 * 4 * 3 * 2 * 1
  - Σημείωση
    - 5! = 5 * 4!
    - 4! = 4 * 3! …
  - Υπολογισμός παραγοντικών αναδρομικά
  - Επίλυση βασικής περίπτωσης (1! = 0! = 1) και στη συνέχεια
    - 2! = 2 * 1! = 2 * 1 = 2;
    - 3! = 3 * 2! = 3 * 2 = 6;

Αναδρομή (3/3)



Ακολουθία αναδρομικών κλήσεων



Τιμές που επιστρέφονται από κάθε αναδρομική κλήση

Παράδειγμα 1 (1/2)

```
1  /* Recursive factorial function */
2  #include <stdio.h>
3
4  long factorial( long number ); /* function prototype */
5
6  /* function main begins program execution */
7  int main()
8  {
9     int i; /* counter */
10
11    /* loop 10 times.  During each iteration, calculate
12       factorial( i ) and display result */
13    for ( i = 1; i <= 10; i++ ) {
14       printf( "%2d! = %ld\n", i, factorial( i ) );
15    } /* end for */
16
17    return 0; /* indicates successful termination */
18
19 } /* end main */
20
```

Παράδειγμα 1 (2/2)

```
21 /* recursive definition of function factorial */
22 long factorial( long number )
23 {
24    /* base case */
25    if ( number <= 1 ) {
26       return 1;
27    } /* end if */
28    else { /* recursive step */
29       return ( number * factorial( number - 1 ) );
30    } /* end else */
31
32 } /* end function factorial */
```

```
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
```

Παράδειγμα 2 : Ακολουθία Fibonacci (1/5)

- Ακολουθία Fibonacci : 0, 1, 1, 2, 3, 5, 8...
  - Κάθε αριθμός είναι το άθροισμα των δύο προηγούμενων
  - Μπορεί να λυθεί αναδρομικά:

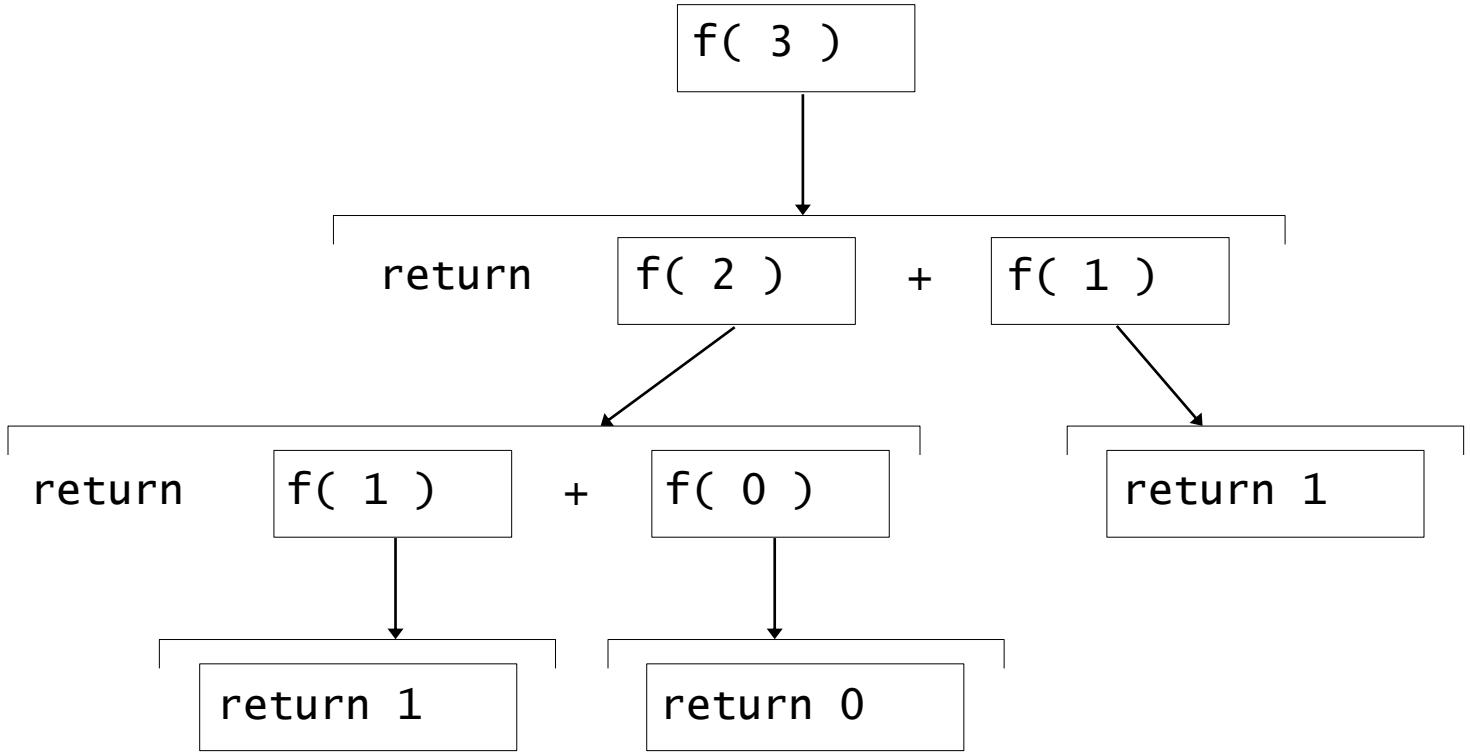
```
fib( n ) = fib( n - 1 ) + fib( n – 2 )
```

  - Κώδικας για τη συνάρτηση fibonacci

```
long fibonacci( long n )
{
    if (n == 0 || n == 1)  // base case
       return n;
    else
       return fibonacci( n - 1) + fibonacci( n – 2 );
}
```

Παράδειγμα 2 : Ακολουθία Fibonacci (2/5)

- Σύνολο αναδρομικών κλήσεων της συνάρτησης fibonacci



Παράδειγμα 2 : Ακολουθία Fibonacci (3/5)

```c
/* Recursive fibonacci function */
#include <stdio.h>

long fibonacci( long n ); /* function prototype */

/* function main begins program execution */
int main()
{
   long result; /* fibonacci value */
    long number; /* number input by user */

    /* obtain integer from user */
    printf( "Enter an integer: " );
    scanf( "%ld", &number );

    /* calculate fibonacci value for number input by user */
    result = fibonacci( number );

    /* display result */
    printf( "Fibonacci( %ld ) = %ld\n", number, result );

    return 0; /* indicates successful termination */

} /* end main */
```

Παράδειγμα 2 : Ακολουθία Fibonacci (4/5)

```
26 /* Recursive definition of function fibonacci */
27 long fibonacci( long n )
28 {
29    /* base case */
30    if ( n == 0 || n == 1 ) {
31       return n;
32    } /* end if */
33    else { /* recursive step */
34       return fibonacci( n - 1 ) + fibonacci( n - 2 );
35    } /* end else */
36
37 } /* end function fibonacci */
```

Παράδειγμα 2 : Ακολουθία Fibonacci (5/5)

```
Enter an integer: 0
Fibonacci( 0 ) = 0

Enter an integer: 1
Fibonacci( 1 ) = 1

Enter an integer: 2
Fibonacci( 2 ) = 1

Enter an integer: 3
Fibonacci( 3 ) = 2

Enter an integer: 4
Fibonacci( 4 ) = 3

Enter an integer: 5
Fibonacci( 5 ) = 5

Enter an integer: 6
Fibonacci( 6 ) = 8

Enter an integer: 10
Fibonacci( 10 ) = 55

Enter an integer: 20
Fibonacci( 20 ) = 6765

Enter an integer: 30
Fibonacci( 30 ) = 832040

Enter an integer: 35
Fibonacci( 35 ) = 9227465
```

Σύγκριση αναδρομής - επανάληψης

- Επανάληψη
  - Επανάληψη: σαφής βρόχο (loop)
  - Αναδρομή: επαναλαμβανόμενες κλήσεις συνάρτησης
- Τερματισμός
  - Επανάληψη: η συνθήκη του βρόχου αποτυγχάνει (γίνεται ψευδής)
  - Αναδρομή: αναγνωρίζεται η βασική περίπτωση (base case)
- Και στις δύο περιπτώσεις μπορεί να υπάρξουν ατέρμονοι βρόχοι (infinite loops)
- Ισορροπία
  - Επιλογή μεταξύ επίδοσης (επανάληψη) και καλής προγραμματιστικής τεχνικής (αναδρομή)

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

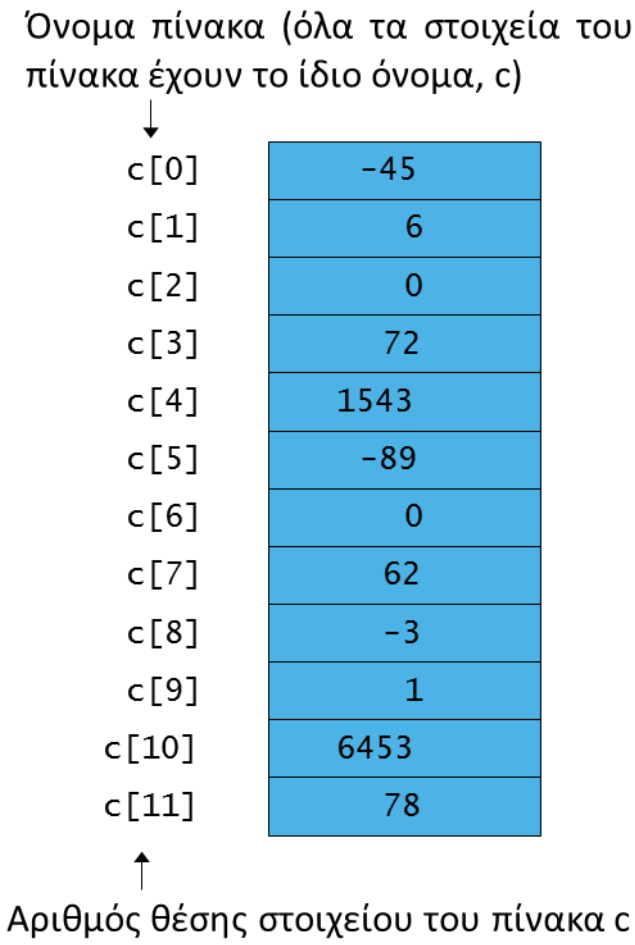
Εισαγωγή

- Θέματα προς μελέτη
  - Δομή δεδομένων πινάκων
  - Αποθήκευση, ταξινόμηση και αναζήτηση πίνακα τιμών
  - Ορισμός, αρχικοποίηση και αναφορά σε μεμονωμένα στοιχεία πίνακα
  - “Πέρασμα” πινάκων σε συναρτήσεις
  - Βασικές τεχνικές ταξινόμησης
- Πίνακες
  - Δομές σχετιζόμενων στοιχείων δεδομένων
  - Στατική οντότητα - ίδιο μέγεθος σε όλη τη διάρκεια του προγράμματος

Πίνακες

- Πίνακας
  - Ομάδα διαδοχικών θέσεων μνήμης
    - Ίδιο όνομα και τύπος
- Η αναφορά στο στοιχείο ενός πίνακα γίνεται με προσδιορισμό του
  - Ονόματος του πίνακα
  - Αριθμού (δείκτη) της θέσης του στοιχείου
- Γενική μορφή

  όνομα πίνακα [αριθμός θέσης]
  - Το πρώτο στοιχείο βρίσκεται στη θέση 0
    - c[0]
  - Το n-οστό στοιχείο ενός πίνακα
    - c[n – 1]



Ορισμός πινάκων

- Όταν ορίζουμε πίνακες προσδιορίζουμε
  - Όνομα του πίνακα
  - Τύπο δεδομένων του πίνακα
  - Αριθμό στοιχείων

    ```
    τύπος_πίνακα όνομα_πίνακα[ αριθμός_στοιχείων ];
    ```
  - Παραδείγματα
    - `int c[ 10 ];`
    - `float myArray[ 3284 ];`
- Ορισμός πολλών πινάκων του ίδιου τύπου
  - Όμοια με τον ορισμό πολλών μεταβλητών του ίδιου τύπου
  - Παράδειγμα:
    - `int b[ 100 ], x[ 27 ];`
- Τα στοιχεία πινάκων είναι κανονικές μεταβλητές
    - `c[ 0 ] =  3;`
    - `printf( "%d", c[ 0 ] );`

Παραδείγματα χρήσης πινάκων

- Αρχικοποίηση
  - `int n[ 5 ] = { 1, 2, 3, 4, 5 };`
  - Αν δεν υπάρχουν αρκετές τιμές αρχικοποίησης τότε τα δεξιότερα στοιχεία παίρνουν την τιμή 0
    - Με την εντολή `int n[ 5 ] = { 0 };`
      - Όλα τα στοιχεία του πίνακα n παίρνουν την τιμή 0 (αρχικά)
  - Αν υπάρχουν **περισσότερες τιμές αρχικοποίησης** (από ότι στοιχεία του πίνακα) τότε παράγεται συντακτικό σφάλμα!
- Αν παραληφθεί το μέγεθος του πίνακα (ο αριθμός των στοιχείων) τότε καθορίζεται από τις τιμές αρχικοποίησης
  - Με την εντολή `int n[ ] = { 1, 2, 3, 4, 5 };`
    - Δίνονται 5 τιμές αρχικοποίησης οπότε ο πίνακας ορίζεται να έχει 5 στοιχεία

Παράδειγμα 1 - Αρχικοποίηση πίνακα (1/2)

```c
/* Example of
   initializing an array */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int n[ 10 ]; /* n is an array of 10 integers */
   int i;       /* counter */

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ ) {
      n[ i ] = 0; /* set element at location i to 0 */
   } /* end for */

   printf( "%s%13s\n", "Element", "Value" );

   /* output contents of array n in tabular format */
   for ( i = 0; i < 10; i++ ) {
      printf( "%7d%13d\n", i, n[ i ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

Παράδειγμα 1 - Αρχικοποίηση πίνακα (2/2)

```
Element         Value
      0             0
      1             0
      2             0
      3             0
      4             0
      5             0
      6             0
      7             0
      8             0
      9             0
```

Παράδειγμα 2 - Αρχικοποίηση πίνακα (1/2)

```
1  /* Example of
2     Initializing an array with an initializer list */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     /* use initializer list to initialize array n */
9     int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10    int i; /* counter */
11
12    printf( "%s%13s\n", "Element", "Value" );
13
14    /* output contents of array in tabular format */
15    for ( i = 0; i < 10; i++ ) {
16       printf( "%7d%13d\n", i, n[ i ] );
17    } /* end for */
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
```

Παράδειγμα 2 - Αρχικοποίηση πίνακα (2/2)

```
Element         Value
      0            32
      1            27
      2            64
      3            18
      4            95
      5            14
      6            90
      7            70
      8            60
      9            37
```

Παράδειγμα 3 - Αρχικοποίηση πίνακα (1/2)

```c
/* Example
   Initialize the elements of array s to the even integers from 2 to 20 */
#include <stdio.h>
#define SIZE 10

/* function main begins program execution */
int main()
{
   /* symbolic constant SIZE can be used to specify array size */
   int s[ SIZE ]; /* array s has 10 elements */
   int j;         /* counter */

   for ( j = 0; j < SIZE; j++ ) { /* set the values */
      s[ j ] = 2 + 2 * j;
   } /* end for */

   printf( "%s%13s\n", "Element", "Value" );

   /* output contents of array s in tabular format */
   for ( j = 0; j < SIZE; j++ ) {
      printf( "%7d%13d\n", j, s[ j ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

Παράδειγμα 3 - Αρχικοποίηση πίνακα (1/2)

```
Element         Value
      0             2
      1             4
      2             6
      3             8
      4            10
      5            12
      6            14
      7            16
      8            18
      9            20
```

Παράδειγμα 4 - Υπολογισμός αθροίσματος στοιχείων πίνακα

```
1  /* Example
2     Compute the sum of the elements of the array */
3  #include <stdio.h>
4  #define SIZE 12
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* use initializer list to initialize array */
10    int a[ SIZE ] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
11    int i;          /* counter */
12    int total = 0; /* sum of array */
13
14    /* sum contents of array a */
15    for ( i = 0; i < SIZE; i++ ) {
16       total += a[ i ];
17    } /* end for */
18
19    printf( "Total of array element values is %d\n", total );
20
21    return 0; /* indicates successful termination */
22
23 } /* end main */
```

```
Total of array element values is 383
```

Παράδειγμα 5 – Γκάλοπ (1/3)

```
1  /* Example
2     Student poll program */
3  #include <stdio.h>
4  #define RESPONSE_SIZE 40 /* define array sizes */
5  #define FREQUENCY_SIZE 11
6
7  /* function main begins program execution */
8  int main()
9  {
10    int answer; /* counter */
11    int rating; /* counter */
12
13    /* initialize frequency counters to 0 */
14    int frequency[ FREQUENCY_SIZE ] = { 0 };
15
16    /* place survey responses in array responses */
17    int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
18         1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
19         5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
20
```

Παράδειγμα 5 – Γκάλοπ (2/3)

```
21    /* for each answer, select value of an element of array responses
22         and use that value as subscript in array frequency to
23         determine element to increment */
24    for ( answer = 0; answer < RESPONSE_SIZE; answer++ ) {
25       ++frequency[ responses [ answer ] ];
26    } /* end for */
27
28    /* display results */
29    printf( "%s%17s\n", "Rating", "Frequency" );
30
31    /* output frequencies in tabular format */
32    for ( rating = 1; rating < FREQUENCY_SIZE; rating++ ) {
33       printf( "%6d%17d\n", rating, frequency[ rating ] );
34    } /* end for */
35
36    return 0; /* indicates successful termination */
37
38 } /* end main */
```

Παράδειγμα 5 – Γκάλοπ (3/3)

Rating	Frequency
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

Παράδειγμα 6 – Ιστόγραμμα (1/2)

```
1  /* Example
2     Histogram printing program */
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* use initializer list to initialize array n */
10    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
11    int i; /* outer counter */
12    int j; /* inner counter */
13
14    printf( "%s%13s%17s\n", "Element", "Value", "Histogram" );
15
16    /* for each element of array n, output a bar in histogram */
17    for ( i = 0; i < SIZE; i++ ) {
18       printf( "%7d%13d        ", i, n[ i ]) ;
19
20       for ( j = 1; j <= n[ i ]; j++ ) { /* print one bar */
21          printf( "%c", '*' );
22        } /* end inner for */
```

Παράδειγμα 6 – Ιστόγραμμα (2/2)

```
24          printf( "\n" ); /* start next line of output */
25       } /* end outer for */
26
27       return 0; /* indicates successful termination */
28
29    } /* end main */
```

```
Element         Value           Histogram
      0            19           ********************
      1             3           ***
      2            15           ***************
      3             7           *******
      4            11           ***********
      5             9           *********
      6            13           *************
      7             5           *****
      8            17           *****************
      9             1           *
```

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

Πέρασμα πινάκων σε συναρτήσεις (1/2)

- Πέρασμα πινάκων
  - Για το πέρασμα ενός πίνακα σε μια συνάρτηση, δίνουμε το όνομα του πίνακα χωρίς αγκύλη
    - `int myArray[ 24 ];`
    - `myFunction( myArray, 24 );`
    - Συνήθως το μέγεθος του πίνακα ως παράμετρο στη συνάρτηση
  - Το πέρασμα πινάκων γίνεται με **κλήση με αναφορά**
  - Το όνομα του πίνακα είναι ουσιαστικά η διεύθυνση του πρώτου στοιχείου του πίνακα
  - Η συνάρτηση γνωρίζει που είναι αποθηκευμένος ο πίνακας
    - Τροποποιεί τα αυθεντικά περιεχόμενα των θέσεων της μνήμης

Πέρασμα πινάκων σε συναρτήσεις (2/2)

- Πέρασμα στοιχείων πινάκων
  - Το πέρασμα γίνεται με **κλήση με τιμή**
  - Περνάμε το "όνομα" του στοιχείου του πίνακα, δηλαδή `myArray[ 3 ]` στη συνάρτηση
- Πρωτότυπο συνάρτησης

```
void modifyArray( int b[], int arraySize );
```

  - Τα ονόματα των παραμέτρων είναι προαιρετικά στη δήλωση του πρωτοτύπου
    - `int b[]` θα μπορούσε να γραφεί `int []`
    - `int arraySize` θα μπορούσε να γραφεί `int`
  - Το παραπάνω παράδειγμα μπορεί να γραφεί ως

```
void modifyArray( int [], int);
```

Παράδειγμα (1/4)

```
1  /* Example
2     Passing arrays and individual array elements to functions */
3  #include <stdio.h>
4  #define SIZE 5
5
6  /* function prototypes */
7  void modifyArray( int b[], int size );
8  void modifyElement( int e );
9
10 /* function main begins program execution */
11 int main()
12 {
13    int a[ SIZE ] = { 0, 1, 2, 3, 4 }; /* initialize a */
14    int i; /* counter */
15
16    printf( "Effects of passing entire array by reference:\n\nThe "
17            "values of the original array are:\n" );
18
19    /* output original array */
20    for ( i = 0; i < SIZE; i++ ) {
21       printf( "%3d", a[ i ] );
22    } /* end for */
23
24    printf( "\n" );
```

Παράδειγμα (2/4)

```
26    /* pass array a to modifyArray by reference */
27    modifyArray( a, SIZE );
28
29    printf( "The values of the modified array are:\n" );
30
31    /* output modified array */
32    for ( i = 0; i < SIZE; i++ ) {
33       printf( "%3d", a[ i ] );
34    } /* end for */
35
36    /* output value of a[ 3 ] */
37    printf( "\n\n\nEffects of passing array element "
38            "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
39
40    modifyElement( a[ 3 ] ); /* pass array element a[ 3 ] by value */
41
42    /* output value of a[ 3 ] */
43    printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
44
45    return 0; /* indicates successful termination */
46
47 } /* end main */
```

Παράδειγμα (3/4)

```
49 /* in function modifyArray, "b" points to the original array "a"
50    in memory */
51 void modifyArray( int b[], int size )
52 {
53    int j; /* counter */
54
55    /* multiply each array element by 2 */
56    for ( j = 0; j < size; j++ ) {
57       b[ j ] *= 2;
58    } /* end for */
59
60 } /* end function modifyArray */
61
62 /* in function modifyElement, "e" is a local copy of array element
63    a[ 3 ] passed from main */
64 void modifyElement( int e )
65 {
66    /* multiply parameter by 2 */
67    printf( "Value in modifyElement is %d\n", e *= 2 );
68 } /* end function modifyElement */
```

Παράδειγμα (4/4)

```
Effects of passing entire array by reference:

The values of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8


Effects of passing array element by value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[ 3 ] is 6
```

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

Ταξινόμηση πινάκων

- Ταξινόμηση δεδομένων
  - Σημαντική υπολογιστική εφαρμογή
  - Ουσιαστικά κάθε οργανισμός πρέπει να ταξινομήσει κάποια δεδομένα
- Ταξινόμηση φυσαλίδας (bubble sort)
  - Αρκετά “περάσματα” του πίνακα
  - Σύγκριση διαδοχικών ζευγαριών στοιχείων ενός πίνακα
    - Αν είναι στην επιθυμητή σειρά, καμία αλλαγή
    - Αν όχι αντιστρέφεται η σειρά τους - εναλλάσσονται τα στοιχεία του πίνακα
  - Επανάληψη
- Παράδειγμα:
  - αρχικά: 3 4 2 6 7
  - πέρασμα 1: 3 2 4 6 7
  - πέρασμα 2: 2 3 4 6 7
  - Μικρότερα στοιχεία «μαζεύονται» (bubble) στην κορυφή

Παράδειγμα (1/3)

```
1  /* Example
2     This program sorts an array's values into ascending order */
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* initialize a */
10    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
11    int i;    /* inner counter */
12    int pass; /* outer counter */
13    int hold; /* temporary location used to swap array elements */
14
15    printf( "Data items in original order\n" );
16
17    /* output original array */
18    for ( i = 0; i < SIZE; i++ ) {
19       printf( "%4d", a[ i ] );
20    } /* end for */
21
```

Παράδειγμα (2/3)

```
22    /* bubble sort */
23    /* loop to control number of passes */
24    for ( pass = 1; pass < SIZE; pass++ ) {
25
26       /* loop to control number of comparisons per pass */
27       for ( i = 0; i < SIZE - 1; i++ ) {
28
29          /* compare adjacent elements and swap them if first
30          element is greater than second element */
31          if ( a[ i ] > a[ i + 1 ] ) {
32             hold = a[ i ];
33             a[ i ] = a[ i + 1 ];
34             a[ i + 1 ] = hold;
35          } /* end if */
36
37       } /* end inner for */
38
39    } /* end outer for */
40
41    printf( "\nData items in ascending order\n" );
42
```

Παράδειγμα (3/3)

```
43    /* output sorted array */
44    for ( i = 0; i < SIZE; i++ ) {
45       printf( "%4d", a[ i ] );
46    } /* end for */
47
48    printf( "\n" );
49
50    return 0; /* indicates successful termination */
51
```

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```

Σημερινή διάλεξη

- Σύνοψη 4ης διάλεξης
- Συναρτήσεις: Κατηγορίες αποθήκευσης
- Κανόνες εμβέλειας
- Αναδρομή
- Πίνακες
  - Εισαγωγή
  - Δήλωση Πινάκων
  - Παραδείγματα χρήσης πινάκων
  - «Πέρασμα» Πινάκων σε Συναρτήσεις
  - Ταξινόμηση Πινάκων
  - Αναζήτηση σε Πίνακες

Αναζήτηση σε πίνακες: Γραμμική και δυαδική αναζήτηση (1/2)

- Αναζήτηση μιας συγκεκριμένης τιμής σε ένα πίνακα
- Γραμμική αναζήτηση (Linear search)
  - Απλός αλγόριθμος
  - Κάθε στοιχείο του πίνακα συγκρίνεται με τη συγκεκριμένη τιμή
  - Χρήσιμη για μικρούς και μη ταξινομημένους πίνακες

Αναζήτηση σε πίνακες: Γραμμική και δυαδική αναζήτηση (2/2)

- Δυαδική αναζήτηση (Binary search)
  - Για ταξινομημένους πίνακες
  - Συγκρίνει το μεσαίο στοιχείο ενός πίνακα (έστω `middle`) με το στοιχείο που ψάχνουμε (έστω `key`)
    - Αν είναι ίσα τότε βρέθηκε το στοιχείο που ψάχνουμε
    - Αν `key < middle`, ψάχνει στο πρώτο μισό του πίνακα
    - Αν `key > middle`, ψάχνει στο δεύτερο μισό του πίνακα
    - Επανάληψη
  - Πολύ γρήγορος αλγόριθμος
    - Το πολύ N βήματα όπου 2^N > (αριθμός στοιχείων πίνακα)
      - Παράδειγμα: για πίνακα 30 στοιχείων χρειάζονται το πολύ 5 βήματα
        - $2^5 > 30$

Ερωτήσεις…