Διαδικαστικός Προγραμματισμός

Ενότητα 4: Συναρτήσεις- Διαδικασίες

Καθηγήτρια Μαρία Σατρατζέμη

Τμήμα Εφαρμοσμένης Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

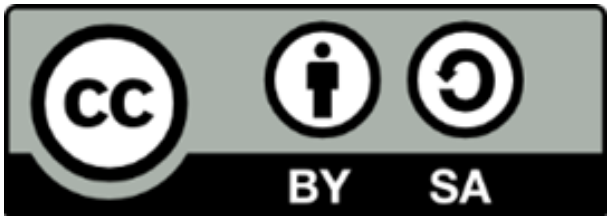
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοποί ενότητας

- Να εκτιμήσετε τις *συναρτήσεις* ως εργαλείο απλοποίησης της δομής των προγραμμάτων.
- Να κατανοήσετε την έννοια της *κλήσης* μιας συνάρτησης και το λόγο για τον οποίο μεταβιβάζονται *ορίσματα* κατά την κλήση.
- Να κατανοήσετε τα *πρωτότυπα των συναρτήσεων* και πώς να τα γράφετε.
- Να είστε σε θέση να υλοποιείτε απλές συναρτήσεις.
- Να μπορείτε να χρησιμοποιείτε την εντολή **return** για τον καθορισμό του αποτελέσματος μιας συνάρτησης.
- Να κατανοήσετε την έννοια των *κατηγορηματικών συναρτήσεων* και πώς να τις χρησιμοποιείτε αποτελεσματικά.
- Να κατανοήσετε τη σχέση μεταξύ των *τυπικών παραμέτρων* μιας συνάρτησης και των *ορισμάτων* που μεταβιβάζονται κατά την κλήση της.
- Να καταλάβετε τη σημασία του όρου *διαδικασία*.

Χρήση συναρτήσεων βιβλιοθήκης (1)

Μια **συνάρτηση** (function) αποτελείται από ένα σύνολο εντολών που έχουν ομαδοποιηθεί και τους έχει αποδοθεί ένα όνομα.

- Μια συνάρτηση, όπως για παράδειγμα η **printf**, αντιπροσωπεύει ένα σύνολο από βήματα που χρησιμεύουν για την εκτέλεση μιας λειτουργίας.
- Στη C το κυρίως πρόγραμμα είναι και αυτό μια συνάρτηση με το όνομα **main**.
- Η εννοιολογική διαφορά μεταξύ μιας συνάρτησης και ενός προγράμματος έγκειται κυρίως στο ποιος ή τι την/το χρησιμοποιεί:
  - τα προγράμματα καλούνται και εξυπηρετούν τις ανάγκες ενός *εξωτερικού χρήστη*
  - οι συναρτήσεις παρέχουν ένα μηχανισμό με τη βοήθεια του οποίου ένα *πρόγραμμα* μπορεί να καλέσει για λογαριασμό του ένα σύνολο λειτουργιών που έχουν οριστεί από πριν.

Χρήση συναρτήσεων βιβλιοθήκης (2)

- Η πράξη της εκτέλεσης του συνόλου των εντολών που σχετίζονται με μια συνάρτηση είναι γνωστή ως **κλήση** (calling) αυτής της συνάρτησης.
- Κατά την κλήση ορισμένων συναρτήσεων πρέπει να δώσουμε επιπλέον πληροφορίες, γνωστές ως **ορίσματα** (arguments).
- Όταν η συνάρτηση ολοκληρώσει το έργο της επιστρέφει στο βήμα του προγράμματος από το οποίο έγινε η κλήση της, **επιστρέφοντας** -ορισμένες φορές- **μια τιμή** (returning a value).

Ορίσματα

- Η κλήση μιας συνάρτησης επιτυγχάνετε γράφοντας το όνομα της συνάρτησης ακολουθούμενο από μια λίστα παραστάσεων μέσα σε παρενθέσεις.
- Οι παραπάνω παραστάσεις ονομάζονται **ορίσματα** (arguments) και επιτρέπουν στο καλούν πρόγραμμα να μεταβιβάζει πληροφορίες στη συνάρτηση.
- Αν μια συνάρτηση δεν απαιτεί πληροφορίες από αυτόν που την καλεί, δεν χρειάζεται να έχει κανένα όρισμα, αλλά το κενό ζεύγος παρενθέσεων είναι απαραίτητο.
- Για παράδειγμα, η κλήση της συνάρτησης **sqrt** (της βιβλιοθήκης **math**) που επιστρέφει την τετραγωνική ρίζα του ορίσματος της θα έχει τη μορφή:

  root3 = sqrt(3.0)

  distance = sqrt(x*x + y*y)

- Αντίθετα, η κλήση της **GetInteger** που δεν έχει ορίσματα θα έχει τη μορφή: **GetInteger()**.

Επιστρεφόμενες τιμές

- Η συνάρτηση παίρνει τα δεδομένα που της έχουν παρασχεθεί ως ορίσματα, κάνει τη δουλειά της, και μετά επιστρέφει στο βήμα του προγράμματος από το οποίο έγινε η κλήση της.
- Η παραπάνω διεργασία είναι γνωστή ως **επιστροφή** (returning) από τη συνάρτηση.
- Κατά την επιστροφή τους οι συναρτήσεις μπορούν να στέλνουν αποτελέσματα πίσω στο καλούν πρόγραμμα, λειτουργία γνωστή ως **επιστροφή τιμής** (returning a value).
- Η συνάρτηση **GetInteger()**, για παράδειγμα, αφού επιτελέσει το έργο της ανάγνωσης ενός ακεραίου από τον χρήστη, μεταβιβάζει αυτόν τον ακέραιο πίσω στο καλούν πρόγραμμα ως τιμή της κλήσης της **GetInteger()**.

Οι έννοιες της εισόδου/εξόδου

- Επειδή τα ορίσματα παρέχουν – κατά μία έννοια – είσοδο στις συναρτήσεις και οι τιμές επιστροφής παρέχουν έξοδο πίσω σε αυτές που τις καλούν, συχνά (τα ορίσματα και οι επιστρεφόμενες τιμές) συγχέονται με τις έννοιες της εισόδου και της εξόδου στα πλαίσια ενός προγράμματος.
- Παρά την εννοιολογική ομοιότητα, είναι εξαιρετικά σημαντικό να διακρίνετε τη λειτουργία της εισόδου (π.χ. την **GetInteger**) και της χρήσης ορισμάτων στα πλαίσια μιας συνάρτησης.
- Είναι σημαντικό να κατανοήσετε ότι:

  Η είσοδος και η έξοδος επιτρέπουν την επικοινωνία ανάμεσα σε ένα *πρόγραμμα* και τον *χρήστη* του,

  ενώ

  τα ορίσματα και τα αποτελέσματα επιτρέπουν την επικοινωνία ανάμεσα σε μια *συνάρτηση* και το *καλούν πρόγραμμα*.

Δήλωση συναρτήσεων

- Στην ANSI C, όλες οι συναρτήσεις –όπως και οι μεταβλητές- πρέπει να έχουν δηλωθεί για να μπορούν να χρησιμοποιηθούν.
- Μια δήλωση συνάρτησης στη C ονομάζεται **πρωτότυπο συνάρτησης** (function protoype) και έχει την ακόλουθη μορφή:

επιστρεφόμενος_τύπος όνομα(προσδιοριστικά_ορισμάτων);

όπου*:*
επιστρεφόμενος_τύπος είναι ο τύπος της τιμής που επιστρέφει η συνάρτηση
όνομα είναι το όνομα της συνάρτησης
προσδιοριστικά_ορισμάτων είναι μια λίστα, με κόμματα ως διαχωριστικά, των μεμονωμένων προδιαγραφών των τύπων των ορισμάτων. Κάθε προδιαγραφή ορίσματος αποτελείται από έναν τύπο ο οποίος ακολουθείται προαιρετικά από ένα περιγραφικό όνομα μεταβλητής

- Στο πρωτότυπο καθορίζονται μόνο οι τύποι των τιμών οι οποίες μεταβιβάζονται από και προς τον καλούντα και την ίδια τη συνάρτηση, και δεν παρέχεται καμία πληροφορία για τις εντολές που απαρτίζουν τη συνάρτηση και το τι κάνει αυτή.

Συγγραφή συναρτήσεων

Η προσθήκη μιας νέας συνάρτησης σε ένα πρόγραμμα C αποτελείται από δύο χωριστά βήματα:

- Χρειάζεται να ορίσετε το πρωτότυπο της συνάρτησης, κάτι που συνήθως γίνεται κοντά στην αρχή του προγράμματος, μετά τις γραμμές **#include**.
- Σε κάποιο μεταγενέστερο σημείο του προγράμματος θα πρέπει να συμπεριλάβετε την υλοποίηση αυτής της συνάρτησης, στην οποία θα ορίζετε τα πραγματικά βήματα που πρέπει να εκτελεστούν.

Η **υλοποίηση** μιας συνάρτησης:

- γίνεται με βάση το πρωτότυπο της – αφαιρώντας το ελληνικό ερωτηματικό μετά τη λίστα ορισμάτων και προσθέτοντας το σώμα της συνάρτησης
- το **σώμα της συνάρτησης** (function body), όπως και κάθε άλλο μπλοκ κώδικα, αποτελείται από εντολές που περικλείονται σε άγκιστρα.

*Στο σώμα της συνάρτησης μπορούμε να έχουμε δηλώσεις μεταβλητών, απλές εντολές, εντολές υπό συνθήκη/επανάληψης, κλήσεις άλλων συναρτήσεων και επιπλέον μια εντολή **return***

Η εντολή return

- Αν μια συνάρτηση επιστρέφει κάποιο αποτέλεσμα τότε οι εντολές που περιλαμβάνονται στο σώμα της πρέπει να περιέχουν τουλάχιστον μία εντολή **return**, η οποία να καθορίζει την τιμή που πρόκειται να επιστραφεί:

```
return (παράσταση);

όπου:
        παράσταση είναι η τιμή που πρόκειται να επιστραφεί

Αν η συνάρτηση δεν έχει αποτέλεσμα η σύνταξη είναι:
        return;
```

Η εντολή **return** έχει διπλό ρόλο:

- δηλώνει ότι η εκτέλεση της συνάρτησης έχει ολοκληρωθεί, οπότε ο έλεγχος του προγράμματος επιστρέφει σε εκείνο το σημείο όπου έγινε η κλήση της
- επιστρέφει την τιμή που προκύπτει από την αποτίμηση της παράστασης.

Συνδυασμός συναρτήσεων με κύρια προγράμματα

- ***Πρόβλημα:*** να γράψετε ένα πρόγραμμα που να δημιουργεί και εμφανίζει έναν πίνακα μετατροπής θερμοκρασιών από την κλίμακα Κελσίου (C) [0, 100] στην κλίμακα Φαρενάιτ (F) βάσει του τύπου:

$$F = \frac{9}{5}C + 32$$

```
#include <stdio.h>
#include "genlib.h"

/*
 * ΣΤΑΘΕΡΕΣ
 * LowerLimit -- αρχική τιμή του πίνακα θερμοκρασιών
 * UpperLimit -- τελική τιμή του πίνακα θερμοκρασιών
 * StepSize   -- μέγεθος βήματος μεταξύ των καταχωρήσεων του πίνακα
 */
#define LowerLimit   0
#define UpperLimit 100
#define StepSize     5

/* Πρωτότυπα συναρτήσεων */
double CelsiusToFahrenheit(double c);
```

πρωτότυπο συνάρτησης

Συνδυασμός συναρτήσεων με κύρια προγράμματα

```
/* Κυρίως Πρόγραμμα */
main()
{
    int c;

    printf("Celsius to Fahrenheit table.\n");
    printf("  C    F\n");
    for (c = LowerLimit; c <= UpperLimit; c += StepSize) {
        printf("%3d  %3g\n", c, CelsiusToFahrenheit(c));
    }
}

/*
 * Συνάρτηση: CelsiusToFahrenheit
 * Χρήση: f = CelsiusToFahrenheit(c);
 * ----------------------------------
 * Επιστρέφει την ισοδύναμη θερμοκρασία σε βαθμούς Φαρενάιτ
 * της εκφρασμένης σε βαθμούς Κελσίου θερμοκρασίας c.
 */
double CelsiusToFahrenheit(double c)
{
    return (9.0 / 5.0 * c + 32);
}
```

Κλήση συνάρτησης

Υλοποίηση (συγγραφή) συνάρτησης

Συναρτήσεις με εσωτερικές δομές ελέγχου (1)

- Έστω ότι έπρεπε να υλοποιήσουμε τη συνάρτηση **abs** που ορίζεται στην πρότυπη βιβλιοθήκη ANSI stdlib και επιστρέφει την απόλυτη τιμή του ορίσματος της. Λαμβάνοντας υπόψη ότι η συνάρτηση έχει το εξής πρωτότυπο

  int abs(int n);

- θα μπορούσαμε να την υλοποιήσουμε ως εξής:

```
int abs(int n)
{
    if (n < 0){
        return (-n);
    } else {
        return (n);
    }
}
```

- Παρομοίως, θα μπορούσαμε να ορίσουμε μια συνάρτηση MinF που να επιστρέφει το μικρότερο από 2 ορίσματα κινητής υποδιαστολής που της μεταβιβάζονται:

```
double MinF(double x, double y)
{
    if (x < y){
        return (x);
    } else {
        return (y);
    }
}
```

Συναρτήσεις με εσωτερικές δομές ελέγχου (2)

Τέλος, θα μπορούσαμε να γράψουμε μια συνάρτηση με όνομα **Factorial** που να δέχεται έναν ακέραιο **n** και να επιστρέφει το παραγοντικό του (το γινόμενο μεταξύ του 1 και του **n**):

Factoriali(0)	=	1	(*εξ ορισμού*)	
Factoriali(1)	=	1	=	1
Factoriali(2)	=	2	=	1 x 2
…				
Factoriali(6)	=	720	=	1 x 2 x 3 x 4 x 5 x 6

```
int Factorial(int n)
{
   int product, i;

   product = 1;
   for (i = 1; i <= n; i++){
        product *= i;
   }
   return (product);
}
```

Συναρτήσεις που επιστρέφουν αλφαριθμητικά (1)

Αν και έχουν παρουσιαστεί μόνο αριθμητικές συναρτήσεις, οι συναρτήσεις της C μπορούν να επιστρέφουν τιμές οποιουδήποτε τύπου δεδομένων, όπως για παράδειγμα

αλφαριθμητικά:

- ***Πρόβλημα:*** να γράψετε μια συνάρτηση η οποία να μετατρέπει τον αριθμό του μήνα (από 1 έως 12) στο αλφαριθμητικό του αντίστοιχου ονόματος του μήνα (από τον Ιανουάριο έως το Δεκέμβριο), προκειμένου να χρησιμοποιηθεί σε ένα πρόγραμμα που επεξεργάζεται ημερομηνίες.

```
string όνομα(προσδιοριστικά_ορισμάτων);
```

```
string MonthName(int month)
{
   switch (month) {
        case  1: return ("January");
        case  2: return ("February");
        case  3: return ("March");
        case  4: return ("April");
        case  5: return ("May");
        case  6: return ("June");
        case  7: return ("July");
        case  8: return ("August");
        case  9: return ("September");
        case 10: return ("October");
        case 11: return ("November");
        case 12: return ("December");
        default: return ("Illegal month");
   }
}
```

Συναρτήσεις που επιστρέφουν αλφαριθμητικά (2)

Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **MonthName**, καλώντας την από κάποιο άλλο σημείο του προγράμματος και χρησιμοποιώντας –για παράδειγμα- την **printf** για να εμφανίσουμε το αποτέλεσμα.

Για παράδειγμα, οι παρακάτω εντολές

```
int month, day, year;
month = 7;
day = 20;
year = 1969;
printf("%s %d, %d\n", MonthName(month), day, year);
```

θα έχουν ως αποτέλεσμα **July 20, 1969**

Προσοχή!! Στην εντολή switch της συνάρτησης **MonthName**, οι εντολές **return** σε κάθε όρο **case** προκαλούν την αυτόματη έξοδο από ολόκληρη τη συνάρτηση και κάνουν περιττή τη χρήση μιας ρητής εντολής **break**.
Σε κάθε περίπτωση, κάθε όρος **case** πρέπει να τελειώνει είτε με μια εντολή **break** είτε με μια **return**.

Κατηγορηματικές συναρτήσεις (1)

Κατηγορηματικές συναρτήσεις (predicate functions) ονομάζονται οι συναρτήσεις που επιστρέφουν τιμές τύπου **bool**.

- Θυμηθείτε ότι ο τύπος **bool**, όπως και ο τύπος **string**, δεν έχει οριστεί στις πρότυπες βιβλιοθήκες της ANSI C και συνεπώς για να χρησιμοποιηθεί θα πρέπει να έχει συμπεριληφθεί στο πρόγραμμα η βιβλιοθήκη **genlib** στην οποία ορίζονται οι παραπάνω τύποι.

- Εφόσον υπάρχουν μόνο δύο τιμές τύπου **bool** – οι **TRUE** και **FALSE** – μια κατηγορηματική συνάρτηση – ανεξάρτητα από το πόσα ορίσματα δέχεται ή πόσο περίπλοκη είναι η εσωτερική της διεργασία – πρέπει τελικά να επιστρέψει μία από τις δύο προαναφερθείσες τιμές.

bool όνομα(προσδιοριστικά_ορισμάτων);

Κατηγορηματικές συναρτήσεις (2)

- Έστω, για παράδειγμα, ότι σας ζητήθηκε να γράψετε μια συνάρτηση, η οποία, όταν της μεταβιβαστεί ένας ακέραιος **n**, απαντά στην ερώτηση «αν ο **n** είναι άρτιος»:

```
bool IsEven(int n)
{
   return (n % 2 == 0);
}
```

- Επειδή η IsEven επιστρέφει ένα λογικό αποτέλεσμα, μπορεί να χρησιμοποιηθεί απευθείας σε οποιαδήποτε συνθήκη.

- Στο διπλανό παράδειγμα η IsEven χρησιμοποιείται σε ένα πρόγραμμα για την εμφάνιση όλων των άρτιων ακεραίων μεταξύ του 1 και του 10:

```
main()
{
   int i;

   for (i = 1; i <= 10; i++){
      if (IsEven(i)) printf(%2d\n", i);
}
```

- Θυμηθείτε ότι είναι περιττό στην υλοποίηση της IsEven να προσθέσουμε μια if, ή να κάνουμε περιττές συγκρίσεις με την τιμή TRUE: if (IsEven(i) == TRUE) --- το == TRUE είναι περιττό

Κατηγορηματικές συναρτήσεις (3)

- Στη συνέχεια, παρουσιάζεται ένα ακόμα παράδειγμα κατηγορηματικής συνάρτησης που ελέγχει αν ένα έτος είναι δίσεκτο ή όχι και επιστρέφει την τιμή **TRUE** ή **FALSE** αντίστοιχα:

```
bool IsleapYear(int year)
{
   return ( ((year % 4 == 0) && (year % 100 !=0))
            || (year % 400 == 0) );
}
```

- Έχοντας υλοποιήσει τη συνάρτηση μπορούμε να τη χρησιμοποιήσουμε σε οποιαδήποτε συνθήκη:

```
if (IsLeapYear(year)) . . .
```

Η συνάρτηση StringEqual (1)

Η συνάρτηση **StringEqual:**

- ορίζεται στη βιβλιοθήκη **strlib**
- όπως φαίνεται και από το πρωτότυπο της

  bool StringEqual(string s1, string s2);

  πρόκειται για μια κατηγορηματική συνάρτηση που δέχεται ως ορίσματα δύο αλφαριθμητικά
- ελέγχει χαρακτήρα προς χαρακτήρα τα δύο ορίσματα και

  - επιστρέφει την τιμή **TRUE** αν τα 2 αλφαριθμητικά είναι ακριβώς ίδια
  - επιστρέφει την τιμή **FALSE** αν υπάρχει οποιαδήποτε διαφορά:

StringEqual(“abc”, “abc”)	*επιστρέφει* **TRUE**
StringEqual(“abc”, “def”)	*επιστρέφει* **FALSE**
StringEqual(“abc”, “abcd”)	*επιστρέφει* **FALSE**
StringEqual(“abc”, “aBc”)	*επιστρέφει* **FALSE**

Η συνάρτηση StringEqual (2)

Η συνάρτηση **StringEqual** μπορεί να χρησιμοποιηθεί, για παράδειγμα, κάθε φορά που θέλετε να θέσετε στον χρήστη μια ερώτηση και έπειτα να ενεργήσετε ανάλογα με την απάντηση του.

Υποθέστε ότι έχετε γράψει ένα πρόγραμμα-παιχνίδι και θέλετε να δώσετε στο χρήστη τη δυνατότητα να ξαναπαίξει το παιχνίδι όταν τελειώσει μια παρτίδα. Μπορείτε πολύ απλά

1. να κάνετε μια ερώτηση εμφανίζοντας ένα προτρεπτικό μήνυμα
2. να πάρετε την απάντηση καλώντας τη συνάρτηση **GetLine**
3. να ελέγξετε την απάντηση χρησιμοποιώντας την **StringEqual**

```
main()
{
   string answer;

   while(TRUE){
        PlayOneGame();
        printf("Would you like to play again?");   1
    2   answer = GetLine();
        if (StringEqual(answer, "no")) break;      3
}
```

Ένα πρόγραμμα υπολογισμού πίνακα παραγοντικών

```
#include <stdio.h>
#include "genlib.h"

#define LowerLimit 0          /* σταθερά – αρχική τιμή πίνακα παραγοντικών */
#define UpperLimit 7          /* σταθερά – αρχική τιμή πίνακα παραγοντικών */

int Factorial(int n);         /* πρωτότυπο συνάρτησης */

main()                        /* κυρίως πρόγραμμα */
{
    int i;

    for (i = LowerLimit; i <= UpperLimit; i++) {
        printf("%d! = %5d\n", i, Factorial(i));
    }
}

int Factorial(int n)          /* ορισμός συνάρτησης */
{
    int product, i;

    product = 1;
    for (i = 1; i <= n; i++) {
        product *= i;
    }
    return (product);
}
```

Ο μηχανισμός κλήσης συναρτήσεων

- Αν δείτε το προηγούμενο πρόγραμμα ως δύο μέρη – το κυρίως πρόγραμμα και τη συνάρτηση **Factorial** – είναι μάλλον εύκολο να το κατανοήσετε.

- Αν όμως δείτε το πρόγραμμα συνολικά – και δεν έχετε κατανοήσει τον τρόπο με τον οποίο πρέπει να βλέπετε τις συναρτήσεις – θα υπάρχει μια σύγχυση, αφού:
  - υπάρχουν δύο μεταβλητές με όνομα **i**, μία στο κυρίως πρόγραμμα και μία στη συνάρτηση **Factorial**, οι οποίες χρησιμοποιούνται ως μεταβλητές βρόχου αλλά έχουν διαφορετικές τιμές
  - χρησιμοποιούνται δύο διαφορετικά ονόματα (**i**, **n**) που αναφέρονται στην ίδια τιμή: στο κυρίως πρόγραμμα το **i** έχει ως τιμή κάθε φορά τον αριθμό του οποίου το παραγοντικό επιχειρούμε να υπολογίσουμε, ενώ στη συνάρτηση **Factorial** αυτή η ίδια τιμή ονομάζεται **n**

Για να κατανοήσετε τι αντιπροσωπεύει η κάθε μεταβλητή θα πρέπει να καταλάβετε πως δουλεύουν οι συναρτήσεις εσωτερικά και πως συνεργάζονται μεταξύ τους και με το κυρίως πρόγραμμα.

Μεταβίβαση παραμέτρων

Στο πρωτότυπο της συνάρτησης **Factorial**

int Factorial(int n)

δηλώνεται μια μεταβλητή τύπου **int** με όνομα **n**, η οποία χρησιμεύει ως δεσμευτικό θέσης για το πραγματικό όρισμα, με απλά λόγια για μια συγκεκριμένη τιμή του ίδιου τύπου που θα μεταβιβαστεί στη συνάρτηση κατά την κλήση της είτε από μια άλλη συνάρτηση είτε από το κυρίως πρόγραμμα.

Μια τέτοια μεταβλητή, η οποία ορίζεται στην κεφαλίδα μιας συνάρτησης και χρησιμεύει ως δεσμευτικό θέσης ονομάζεται **τυπική παράμετρος** (formal parameter).

Για παράδειγμα, αν στη **main** υπάρχει η κλήση **Factorial(4)** τότε στην τυπική παράμετρο **n** της συνάρτησης αντιγράφεται η τιμή 4, ή αλλιώς λέμε ότι στη συνάρτηση μεταβιβάζεται η τιμή 4 ή αλλιώς ότι η συνάρτηση δέχεται την τιμή 4.

Κλήση μιας συνάρτησης

Όταν καλείται μια συνάρτηση, εκτελούνται τα ακόλουθα βήματα:

1. Οι τιμές κάθε ορίσματος υπολογίζονται ως μέρος της λειτουργίας του καλούντος προγράμματος. *Τα ορίσματα είναι παραστάσεις και μπορεί να περιλαμβάνουν τελεστές ή ακόμα και άλλες συναρτήσεις που πρέπει να αποτιμηθούν πριν να κληθεί η νέα συνάρτηση*.

2. Η τιμή κάθε ορίσματος αντιγράφεται στην αντίστοιχη τυπική παράμετρο. *Αν είναι αναγκαίο, εκτελείται αυτόματη μετατροπή τύπων μεταξύ των τιμών των ορισμάτων και των τυπικών παραμέτρων, όπως ακριβώς και σε μια εντολή ανάθεσης.*

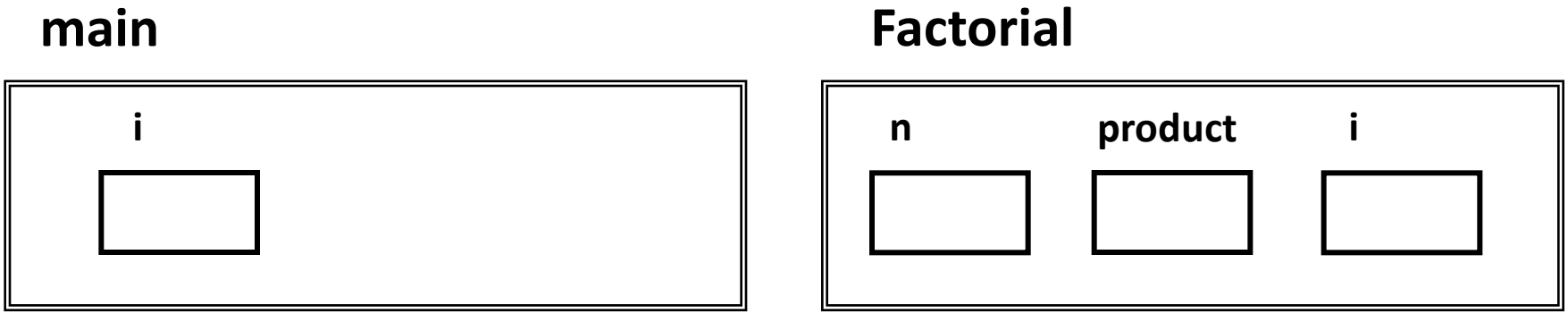
3. Οι εντολές που περιλαμβάνονται στο σώμα της συνάρτησης αποτιμώνται μέχρι να εμφανιστεί μια εντολή **return**.

4. Η τιμή της παράστασης αποτιμάται και μετατρέπεται, αν είναι αναγκαίο, στον τύπο αποτελέσματος που καθορίζεται για τη συνάρτηση.

5. Η εκτέλεση του καλούντος προγράμματος συνεχίζεται, με την κλήση της συνάρτησης να αντικαθιστάται από την επιστρεφόμενη τιμή της.

Τοπικές μεταβλητές

- Κάθε κλήση μιας συνάρτησης έχει ως αποτέλεσμα τη δημιουργία ενός χωριστού συνόλου μεταβλητών, συμπεριλαμβανομένων και των τυπικών παραμέτρων.
- Αυτές οι μεταβλητές έχουν νόημα μόνο μέσα στη συνάρτηση (δηλαδή στο κυρίως πρόγραμμα ή οποιαδήποτε άλλη συνάρτηση) όπου δηλώνονται και γι' αυτόν το λόγο ονομάζονται **τοπικές μεταβλητές** (local variables).
- Η συλλογή μεταβλητών ονομάζεται **πλαίσιο** (frame) – ή για λόγους που θα γίνουν προφανείς στη συνέχεια, **στοίβα πλαισίων** (stack frame) – της συγκεκριμένης συνάρτησης.

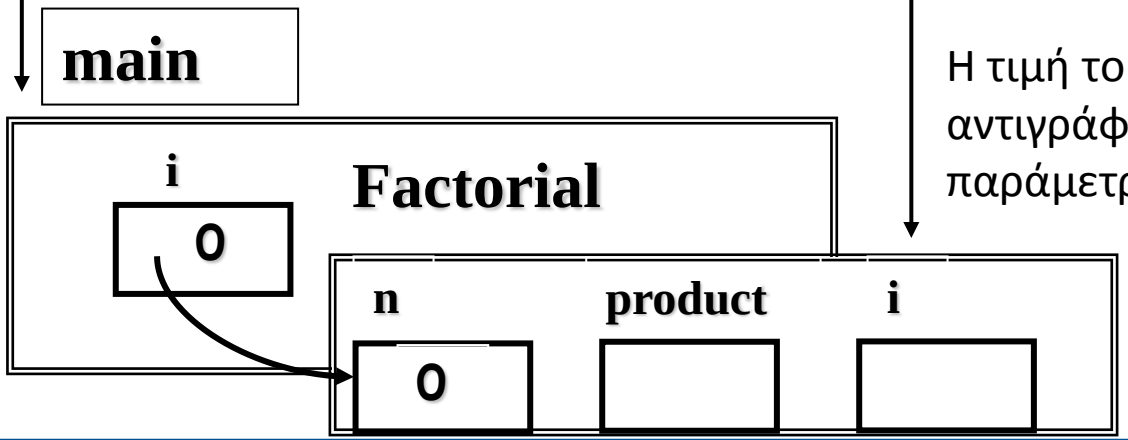


Εκτέλεση προγράμματος (1)

```
…

#define LowerLimit 0
…
main()
{
     int i;

     for (i = LowerLimit; i <= UpperLimit; i++) {

          printf("%d! = %5d\n", i, Factorial(i));
     }
}
```

1ος κύκλος εκτέλεσης του for

Κλήση της Factorial



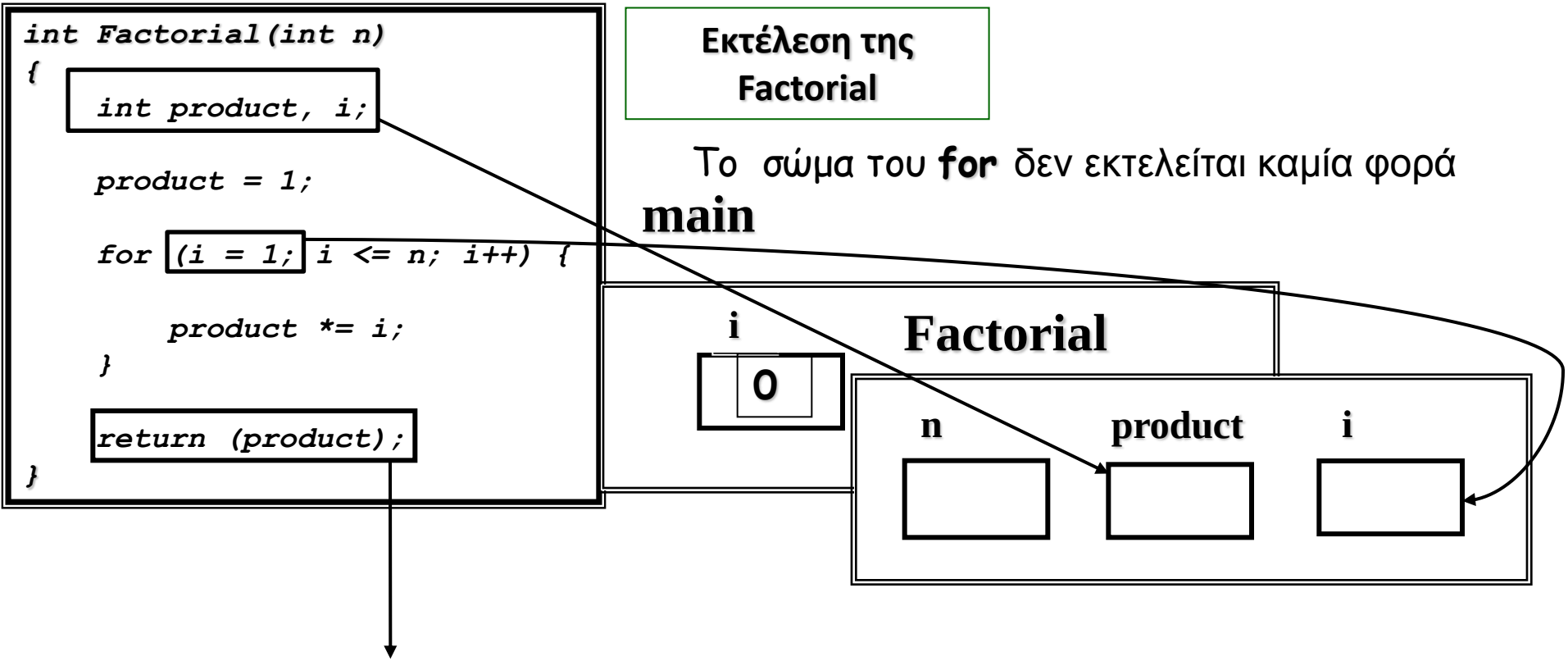
Η τιμή του ορίσματος αντιγράφεται στην τυπική παράμετρο της **Factorial**

Εκτέλεση προγράμματος (2)

```
int Factorial(int n)
{
    int product, i;

    product = 1;

    for (i = 1; i <= n; i++) {

        product *= i;
    }

    return (product);
}
```

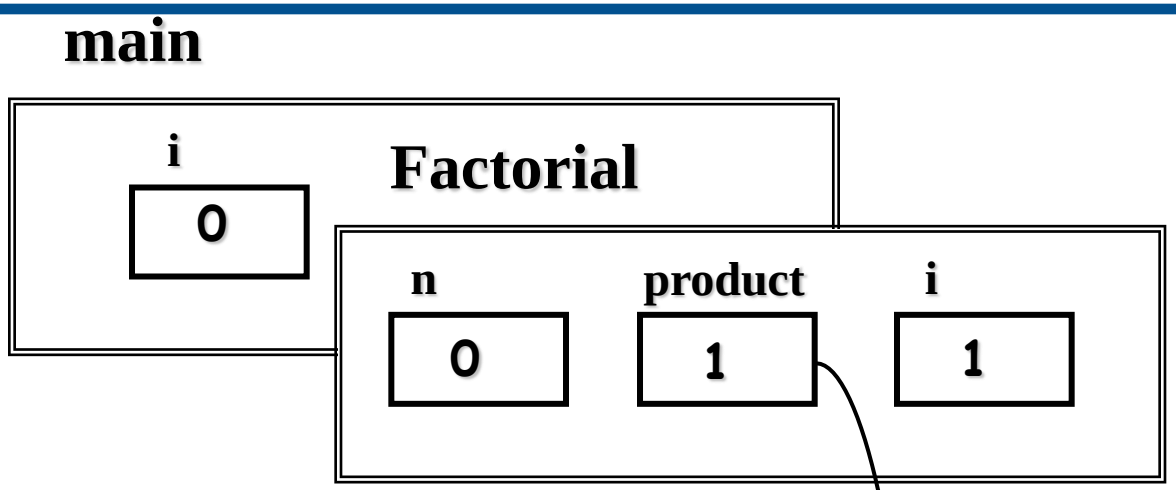
Εκτέλεση της Factorial

Το σώμα του **for** δεν εκτελείται καμία φορά



Με την εντολή **return** ολοκληρώνεται η εκτέλεση της συνάρτησης και επιστρέφεται η τιμή της **product** εκεί που έγινε η κλήση της

Εκτέλεση προγράμματος (3)

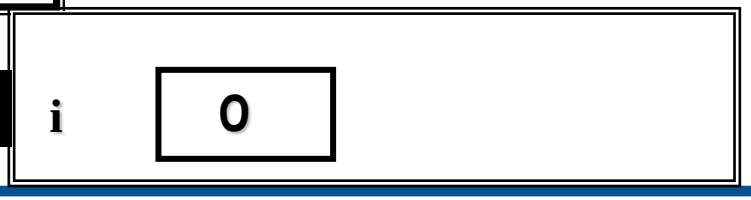



```
main()
{
    int i;

    for (i = LowerLimit; i <= UpperLimit; i++) {
        printf("%d! = %5d\n", i, Factorial(i));
    }
}
```

Επιστροφή μετά την εκτέλεση της `Factorial`

Ως αποτέλεσμα εκτέλεσης της συνάρτησης επιστρέφεται η τιμή 1

main



Ολοκληρώνεται η εκτέλεση της **printf** και εμφανίζονται στην οθόνη τα εξής: **0! = 1**

Κλήση συναρτήσεων μέσα σε άλλες (1)

- Από τη στιγμή που θα ορίσουμε μία συνάρτηση μπορούμε να τη χρησιμοποιήσουμε, όχι μόνο στα πλαίσια του κύριου προγράμματος, αλλά και στον ορισμό άλλων συναρτήσεων.

- Όταν ο υπολογιστής εκτελεί μια νέα κλήση συνάρτησης, καταγράφει το σημείο του καλούντος προγράμματος από το οποίο θα πρέπει να συνεχίσει η εκτέλεση του προγράμματος, μόλις ολοκληρωθεί η εκτέλεση της συνάρτησης που κλήθηκε. Αυτό το σημείο ονομάζεται **διεύθυνση επιστροφής μεταβλητές** (return address).

Στη συνέχεια, παρουσιάζεται ένα παράδειγμα όπου χρησιμοποιείται η συνάρτηση **Factorial** στον ορισμό της νέας συνάρτησης **Combinations** που υλοποιεί τη συνάρτηση συνδυασμών, η οποία δίνει το πλήθος των τρόπων με τους οποίους μπορεί να επιλεγεί ένα υποσύνολο **k** αντικειμένων από ένα σύνολο **n** διακριτών αντικειμένων:

$$C(n,k) = \frac{n!}{k! \times (n-k)!}$$

Κλήση συναρτήσεων μέσα σε άλλες (2)

```
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
/* πρωτότυπα συναρτήσεων */
int Combinations(int n, int k);
int Factorial(int n);
/* κυρίως πρόγραμμα */
main()
{
    int n, k;

    printf("Enter number of objects in the set (n)? ");
    n = GetInteger();
    printf("Enter number to be chosen (k)? ");
    k = GetInteger();
    printf("C(%d, %d) = %d\n", n, k, Combinations(n, k));
}
/* ο ορισμός της συνάρτησης Combinations */
int Combinations(int n, int k)
{
    return (Factorial(n) / (Factorial(k) * Factorial(n - k)));
}
/* ο ορισμός της συνάρτησης Factorial */
int Factorial(int n)
{
    //ο κώδικας παραλείπεται
}
```

Διαδικασίες (1)

Μια συνάρτηση η οποία δεν επιστρέφει κάποια τιμή, αλλά εκτελείται για την επίδραση που έχει ονομάζεται **διαδικασία** (procedure).

- Σε αρκετές γλώσσες προγραμματισμού, όπως στην Pascal και την Fortran, οι συναρτήσεις και οι διαδικασίες είναι διαφορετικές εννοιολογικές οντότητες και ορίζονται με εντελώς διαφορετικούς μηχανισμούς.
- Στη C, οι έννοιες της συνάρτησης και της διαδικασίας είναι συγχωνευμένες, και τις περισσότερες φορές ο όρος *συνάρτηση* χρησιμοποιείται και για τις συναρτήσεις και για τις διαδικασίες.
- Στη C, μια διαδικασία αναγνωρίζεται από το γεγονός ότι στο πρωτότυπο της συνάρτησης χρησιμοποιείται ως τύπος του αποτελέσματος της η λέξη-κλειδί **void**:

void όνομα(προσδιοριστικά_ορισμάτων);

Διαδικασίες (2)

Μια διαδικασία μπορεί να χρησιμοποιηθεί, για παράδειγμα, προκειμένου να εμφανίσουμε με εντολές **printf** κάποιες πληροφορίες ή ένα μενού επιλογών και να έχουμε ένα πιο ευανάγνωστο κύριο πρόγραμμα.

```
…

/* πρωτότυπο συνάρτησης */
void GiveInstructions(void);
…
main()
{
    GiveInstructions();
    /* υπόλοιπο του κύριου προγράμματος */
}

/* ορισμός συνάρτησης */
void GiveInstructions(void)
{
    printf("……");
    …
}
```

Βηματική εκλέπτυνση (1)

- Η διαδικασία της διαίρεσης ενός προβλήματος σε μικρότερα μέρη, τα οποία μπορούμε να διαχειριστούμε ευκολότερα, ονομάζεται **αποσύνθεση** (decomposition) και αποτελεί μια θεμελιώδη στρατηγική του προγραμματισμού.

- Η εύρεση της ενδεδειγμένης μεθόδου αποσύνθεσης είναι αρκετά δύσκολη και απαιτεί εξάσκηση.

- Μια γνωστή στρατηγική επίλυσης προβλημάτων είναι η **βηματική εκλέπτυνση** (stepwise refinement) ή **αναλυτικός σχεδιασμός** (top-down design, **σχεδιασμός "από πάνω προς τα κάτω"**):
    - αρχίζουμε με το κύριο πρόγραμμα, και προσπαθούμε να διακρίνουμε τα κύρια μέρη/"κομμάτια" του προγράμματος
    - τα "κομμάτια" ή αλλιώς τα συστατικά στοιχεία του προγράμματος που είναι πολύπλοκα υποδιαιρούνται σε μικρότερα μέρη
    - η διεργασία αυτή συνεχίζεται μέχρι κάθε "κομμάτι" του προβλήματος να είναι αρκετά απλό ώστε να είναι δυνατό να λυθεί από μόνο του.

Βηματική εκλέπτυνση (2)

Πρόβλημα: να γράψετε ένα πρόγραμμα που θα δέχεται ένα έτος και θα εμφανίζει στην οθόνη ένα ημερολόγιο για όλο το έτος, όπως φαίνεται παρακάτω:

```
This program displays a calendar for a full
year.  The year must not be before 1900.
Which year? 2007
    January 2007
 Su Mo Tu We Th Fr Sa
     1  2  3  4  5  6
  7  8  9 10 11 12 13
 14 15 16 17 18 19 20
 21 22 23 24 25 26 27
 28 29 30 31

    February 2007
 Su Mo Tu We Th Fr Sa
              1  2  3
  4  5  6  7  8  9 10
 11 12 13 14 15 16 17
 18 19 20 21 22 23 24
 25 26 27 28

    March 2007
 Su Mo Tu We Th Fr Sa
              1  2  3
  4  5  6  7  8  9 10
 11 12 13 14 15 16 17
 18 19 20 21 22 23 24
 25 26 27 28 29 30 31

    April 2007
 Su Mo Tu We Th Fr Sa
  1  2  3  4  5  6  7
  8  9 10 11 12 13 14
 15 16 17 18 19 20 21
 22 23 24 25 26 27 28
 29 30

    May 2007
 Su Mo Tu We Th Fr Sa
```

Ξεκίνημα από "επάνω" – κυρίως πρόγραμμα

```
main()
{
   int year;

   GiveInstructions();
   year = GetYearFromUser();
   PrintCalendar(year);
}
```

Βηματική εκλέπτυνση (3)

```
void PrintCalendar(int year)
{
    int month;
    for (month = 1; month <= 12; month++) {
          PrintCalendarMonth(month, year);
          printf("\n");
    }
}
```

Υλοποίηση της PrintCalendar

```
void PrintCalendarMonth(int month, int year)
{
    int weekday, nDays, day;
    printf("    %s %d\n", MonthName(month), year);
    printf(" Su Mo Tu We Th Fr Sa\n");
    nDays = MonthDays(month, year);
    weekday = FirstDayOfMonth(month, year);
    IndentFirstLine(weekday);
    for (day = 1; day <= nDays; day++) {
          printf(" %2d", day);
          if (weekday == Saturday) printf("\n");
          weekday = (weekday + 1) % 7;     }
    if (weekday != Sunday) printf("\n");
}
```

Υλοποίηση της PrintCalendarMonth

Βηματική εκλέπτυνση (4)

```
void IndentFirstLine(int weekday)
{
    int i;

    for (i = 0; i < weekday; i++) {
         printf("   ");
    }
}
```

Υλοποίηση της IndentFirstLine

```
int MonthDays(int month, int year)
{
   switch (month) {
        case 2:
                if (IsLeapYear(year)) return (29);
        return (28);
        case 4: case 6: case 9: case 11:
                return (30);
        default:
                return (31);
   }
}
```

Υλοποίηση της MonthDays

Βηματική εκλέπτυνση (5)

```
int FirstDayOfMonth(int month, int year)
{
   int weekday, i;
   weekday = Monday;
   for (i = 1900; i < year; i++) {
         weekday = (weekday + 365) % 7;
         if (IsLeapYear(i)) weekday = (weekday + 1) % 7;    }
   for (i = 1; i < month; i++) {
   weekday = (weekday + MonthDays(i, year)) % 7;
   }
   return (weekday);
}
```

Υλοποίηση της FirstDayOfMonth

```
int GetYearFromUser(void)
{
   int year;
   while (TRUE) {
         printf("Which year? ");
         year = GetInteger();
         if (year >= 1900) return (year);
         printf("The year must be at least 1900.\n");    }
}
```

Συμπλήρωση των τελευταίων κενών

Τέλος Ενότητας