



Αφαιρετικότητα στις διεργασίες Συναρτήσεις

Εισαγωγή στον Προγραμματισμό

(CEID_NY131)

Procedural Abstraction - Functions



Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
<https://sites.google.com/site/thramboulidiskleanthis/>

Η συνάρτηση main()

HelloWorld.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char *argv[]) {
5     int i;
6
7     printf("Hello world!\n");
8     printf("argc=%d\n", argc);
9     for(i=0;i<argc;i++)
10        printf("argv[%d]=%s\n", i,argv[i]);
11     return 0;
12 }

```

Η συνάρτηση από την οποία ξεκινά η εκτέλεση του προγράμματος.

argc: Ο αριθμός των παραμέτρων που περνάμε στην main όταν εκτελούμε το πρόγραμμα

argv: Πίνακας δεικτών στα αλφαριθμητικά της γραμμής διαταγών (command line) με την οποία εκτελούμε το πρόγραμμα.

```

C:\Code\courses\I2P22-23\HelloWorld>helloworld "by kleanthis" Thramboulidis
Hello world!
argc=3
argv[0]=helloworld
argv[1]=by kleanthis
argv[2]=Thramboulidis

```

main(): Η συνάρτηση η οποία περιγράφει τη διεργασία που θα εκτελέσει ο υπολογιστής όταν τρέξουμε το πρόγραμμα.

Συνάρτηση - Ορισμός

Που είναι το interface και που το implementation της συνάρτησης sqrt;

- είναι μία αυτόνομη μονάδα κώδικα σχεδιασμένη να επιτελεί συγκεκριμένο έργο.
- π.χ. Έργο: Υπολογισμός τετραγωνικής ρίζας
 - Interface: **double sqrt(double x)**
 - Implementation: ?

Τύπος επιστρεφόμενης τιμής

Όνομα ορίσματος

Τύπος ορίσματος

Όνομα συνάρτησης

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 3

Βασική Βιβλιοθήκη της C

- Ορίζει (περιέχει) ένα σύνολο από συναρτήσεις που υλοποιούν **βασικές διεργασίες**
- Η γνώση των συναρτήσεων αυτών αποτελεί **βασική προϋπόθεση** για την ανάπτυξη σύνθετων προγραμμάτων.

Standard C library

<https://www.csse.uwa.edu.au/programming/ansic-library.html>
 Standard C Library Functions Table, By Name - IBM
<https://www.ibm.com/docs/en/i/7.3?topic=extensions-standard-c-library-functions-table-by-name>

- 1. ANS X3.159-1989** πρότυπο
τον Δεκέμβριο του 1989 και δημοσιεύθηκε άνοιξη του 1990
- 2. ISO 9899-1990** πρότυπο
είναι σχεδόν αντίγραφο του ANSI. (ISO:International Organization for Standardization)
- 3. ISO/IEC 899:1999**, το τρέχον πρότυπο
αντικατέστησε το 1990 ISO πρότυπο

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 4

Βασική Βιβλιοθήκη - κατηγορίες συναρτήσεων

- εισόδου εξόδου
- μαθηματικές
- διαχείρισης αλφαριθμητικών
- ταξινόμηση χαρακτήρων και μετατροπές χαρακτήρων
- μετατροπής δεδομένων
- αναζήτησης και ταξινόμησης
- διαχείρισης αρχείων
- διαχείρισης μνήμης
-

Δες παράρτημα Γ
του βιβλίου

standard library – συναρτήσεις I/O

■ Εισόδου

```
int getchar(void);
```

```
int scanf(const char* format, ...);
```

```
char *gets(char *str)
```

■ Εξόδου

```
int putchar(int c);
```

```
int puts(const char* s);
```

```
int printf(const char *format,  
όρισμα2, όρισμα3, ...);
```

Δες άσκηση 1 Κεφάλαιο 11
Χρήση συναρτήσεων BB

Δες άσκηση 4 Κεφάλαιο 11
Για προχωρημένα θέματα συναρτήσεων

Η συνάρτηση printf

Η συνάρτηση printf

```
#include <stdio.h> /* αρχείο επικεφαλίδας */
```

Το **πρωτότυπο** της (function prototype) είναι:

```
int printf(const char *format, όρισμα2,...);
```

- Το πρώτο όρισμα ορίζει σύμφωνα με δεδομένους κανόνες (δες εγχειρίδιο βασικής βιβλιοθήκης) τον αριθμό και τον τύπο των ορισμάτων που θα εξάγει η συνάρτηση στην κύρια έξοδο. Αυτά θα πρέπει να είναι σε συμφωνία με τα υπόλοιπα ορίσματα που ακολουθούν. Τις τιμές αυτών των ορισμάτων βγάζει η συνάρτηση στην έξοδο.

C library function - printf()

© Κλεάνθης Θραμπουλιδής Procedural Abstraction – Functions 7

Χρήση της printf()

Προσδιορίστης	Για εκτύπωση
%d, %i	ακεραίου
%c	χαρακτήρα
%s	αλφαριθμητικού
%f	Κινητής υποδιαστολής
%x	Ακέραιο σε δεκαεξαδική μορφή
%o	Ακέραιο σε οκταδική μορφή
%p	Δείκτη σε void

```
#include <stdio.h>
main()
{
int num1 = 2;
int num2 = 4;
int sum;
sum = 2 + 4;
printf("2 + 4 = %d\n", sum);
printf("%d + %d = %d\n", num1, num2, sum);
}
```

© Κλεάνθης Θραμπουλιδής Procedural Abstraction – Functions 8

Χρήση της sqrt() της BB

για την χρήση της συνάρτησης sqrt() είναι απαραίτητη η συμπερίληψη του αρχείου math.h

```
#include <math.h>
#include <stdio.h>
main()
{
double num1 = 16;
double result;
result = sqrt(num1);
printf("sqrt of %f is %f\n", num1, result);
printf("sqrt of %f is %f\n", num1, sqrt(num1) );
}
```

Αξιοποίηση επιστρεφόμενης τιμής

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 9

Μορφές εμφάνισης συνάρτησης

- **δήλωση** συνάρτησης

```
int max(int a, int b);
```

Πρέπει να προηγείται της κλήσης
- **ορισμός** συνάρτησης

```
int max(int a, int b) {
return (a>b?a:b);
}
```

Πρέπει να υπάρχει στο πρόγραμμα μας εκτός αν υπάρχει σε βιβλιοθήκη
- **κλήση** συνάρτησης

```
printf("%d\n", max(num1, num2) );
max_num = max(num1, num3);
```

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 10

Function prototype in C

- **Ορίζει για την συνάρτηση**
 - το όνομα με το οποίο καλείται
 - τον **τύπο της επιστρεφόμενης τιμής** της
 - η χρήση του **void** σημαίνει ότι η συνάρτηση δεν επιστρέφει τιμή
 - τον **αριθμό και τον τύπο των ορισμάτων** που αυτή δέχεται
 - η χρήση του **void** σημαίνει ότι η συνάρτηση δεν δέχεται ορίσματα

double max(double a, double b);

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 11

Arguments - Ορίσματα

- **τυπικά** (formal)

διαδραματίζουν ρόλο μεταβλητών για την συνάρτηση

int max(int a, int b);

a και b είναι τα τυπικά ορίσματα της max
- **πραγματικά** (actual)

αποτελούν τις πραγματικές τιμές που αποδίδονται (by value ή by reference) στα τυπικά ορίσματα

max_num = max(num1, num2);

num1 και num2 είναι τα Πραγματικά ορίσματα

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 12

Επιστροφή τιμής

- το keyword **return**
 - τερματίζει** τη συνάρτηση επιστρέφοντας τον έλεγχο στην επόμενη πρόταση από αυτή της κλήσης της συνάρτησης και
 - επιστρέφει** την τιμή της έκφρασης που το ακολουθεί.
return n<m?n:m;
- για τον τερματισμό συνάρτησης χωρίς επιστρεφόμενη τιμή χρησιμοποιούμε την πρόταση
return;

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 13

Επεξήγηση μηχανισμού κλήσης συνάρτησης 1/2

result = max(max(15, 13), 17);


- Αποδίδονται τιμές στα τυπικά ορίσματα a, b.
- Μεταφέρεται ο έλεγχος στο σώμα της συνάρτησης max.
- Επιστρέφει ο έλεγχος και η επιστρεφόμενη τιμή της συνάρτησης είναι η τιμή της έκφρασης max(15,13)

```

int max(int a, int b)
{
    return(a>b?a:b);
}

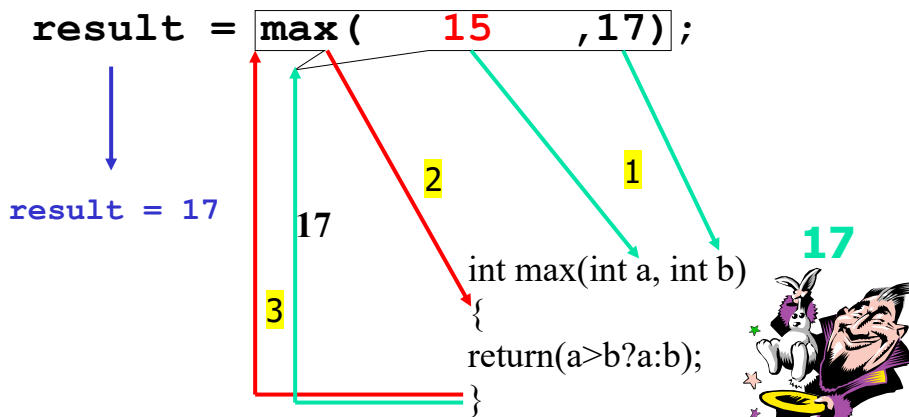
```

15



© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 14

Επεξήγηση μηχανισμού κλήσης συνάρτησης 2/2



Πλεονεκτήματα χρήσης συνάρτησης

- αποφυγή επανάληψης
- επαναχρησιμοποίηση
 - Μείωση χρόνου ανάπτυξης
 - Αύξηση αξιοπιστίας
- βελτίωση αναγνωσιμότητας
- βελτίωση της διαδικασίας συντήρησης

Η συνάρτηση readExpression()

Δες άσκηση Fractions

Function prototype (περιέχεται στο αρχείο i2p.h)

```
void readExpression(char *operatorPtr, int *op1nPtr, int *op1dPtr, int
*op2nPtr, int *op2dPtr);
```

- διαβάζει τα στοιχεία που αποτελούν μια έκφραση προθεματικού τελεστή (prefix notation) (παράγραφος 4.2.1) με κλάσματα.
π.χ. + 2/13 5/13
- Δηλαδή τον τελεστή και τους αριθμητές και παρανομαστές των κλασμάτων.
- Δέχεται ως ορίσματα δείκτες σε μεταβλητές όπου θα βάλει τις τιμές που θα διαβάσει από την κύρια είσοδο.
- Για κατανόηση της δείτε την readInt() στη συνέχεια, η οποία διαβάζει από την βασική είσοδο ένα ακέραιο.

readInt() με επιστρεφόμενη τιμή

int readInt(void); //Function prototype (Δήλωση συνάρτησης)

```
23 int readInt(void){
24     int num;
25     printf("Dose akeraio:");
26     scanf("%d",&num);
27     return num;
28 }
```

Ορισμός της
συνάρτησης

Δηλώνει τοπική μεταβλητή και την χρησιμοποιεί για να αποθηκεύσει τον ακέραιο που διαβάζει με την scanf

Παράδειγμα χρήσης
της συνάρτησης

```
7 int readInt(void);
8
9 int main(int argc, char *argv[]) {
10     int num1;
11
12     num1 = readInt();
13     printf("num1=%d\n", num1);
```

readInt() χωρίς επιστρεφόμενη τιμή

void readInt(int *numPtr); //Function prototype (Δήλωση συνάρτησης)

Ορισμός της συνάρτησης

```

23 void readInt(int *numPtr){
24     // int num;
25     printf("Dose akeraio:");
26     scanf("%d", numPtr);
27     return;
28 }

```

Δεν χρειάζεται τοπική μεταβλητή. Καλεί την scanf και της ζητάει να αποθηκεύσει τον ακέραιο που διαβάζει στην διεύθυνση που δείχνει ο numPtr.

Παράδειγμα χρήσης της συνάρτησης

```

7 void readInt(int *numPtr);
8
9 int main(int argc, char *argv[]) {
10     int num1;
11
12     readInt(&num1);
13     printf("num1=%d\n", num1);

```

Καλεί την συνάρτηση και της περνάει ως όρισμα τον δείκτη στην num1 (πέρασμα by reference) ώστε εκεί να βάλει η scanf τον ακέραιο που διαβάζει από την stdin.

© Κλεάνθης Θραμπουλιδής Procedural Abstraction – Functions 19

readInt (Υπέρ – Κατὰ)

- Η έκδοση με την επιστρεφόμενη τιμή είναι καλύτερη γιατί μας επιτρέπει να την χρησιμοποιούμε σε εκφράσεις, όπως π.χ.
 - if(readNum()!=0)
 - printf("%d\n",readInt());
- Η έκδοση χωρίς επιστρεφόμενη τιμή μας δίνει τη δυνατότητα να ορίσουμε τη συνάρτηση να διαβάζει περισσότερους ακέραιους, όπως π.χ.
 - void readInt(int *num1Ptr, int *num2Ptr);

Για να χρησιμοποιηθεί την παραπάνω readInt θα πρέπει να έχουν δηλωθεί δύο ακέραιες μεταβλητές και να κληθεί η readInt περνώντας της τους δείκτες των μεταβλητών αυτών (πέρασμα ορίσματος by reference- κατα αναφορά).

© Κλεάνθης Θραμπουλιδής Procedural Abstraction – Functions 20

Πέρασμα Ορισμάτων

- **Κλήση κατά αξία** (by value)

τα πραγματικά ορίσματα αποδίδουν τιμές στα τυπικά ορίσματα. Έτσι η **συνάρτηση δουλεύει πάνω σε αντίγραφα** των πραγματικών ορισμάτων.

```
int max(int a, int b);
```
- **Κλήση κατά αναφορά** (by reference)

στην ουσία είναι κλήση κατά αξία, αλλά καθώς περνάνε δείκτες, επιτυγχάνεται το αποτέλεσμα της κλήσης κατά αναφορά. Η **συνάρτηση** χρησιμοποιώντας τους δείκτες έχει πρόσβαση στα πραγματικά ορίσματα.

```
void swap(int *, int *):
swap(&num1, &num2);
```

Οι πίνακες περνάνε πάντα κατά αναφορά

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 21

Συνάρτηση – Ορισμός

- είναι ο **μηχανισμός της γλώσσας** που επιτρέπει στον προγραμματιστή
 - να υλοποιήσει (implement) τις διεργασίες που έχει εντοπίσει, καταγράψει και περιγράψει (πιθανόν με δομημένα Ελληνικά) στη φάση σχεδιασμού του προγράμματος εφαρμόζοντας την βασική αρχή της αφαιρετικότητας. (*top-down approach*)
 - να ομαδοποιήσει ένα σύνολο από προτάσεις για να κάνει τον κώδικα του αρθρωτό (modular) και να επιτρέψει την εύκολη επαναχρησιμοποίηση τους σε πολλά σημεία του κώδικα. (*bottom-up approach*)

© Κλεάνθης Θραμπουλίδης Procedural Abstraction – Functions 22