Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κεντρικής Μακεδονίας - Σέρρες
Τμήμα Μηχανικών Πληροφορικής

Προγραμματισμός ΙΙ (Θ)

Δρ. Δημήτρης Βαρσάμης
Επίκουρος Καθηγητής

Μάρτιος 2017

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΙΙ (Θ)

Αναδρομικές συναρτήσεις I

Αναδρομικές, λέγονται οι συναρτήσεις που καλούν τον εαυτό τους.

Οι αναδρομικές συναρτήσεις περιέχουν πιο συμπαγή κώδικα, είναι ιδιαίτερα χρήσιμες για λίστες, δέντρα κ.α.

Προσοχή,

- οι αναδρομικές συναρτήσεις έχουν το πλεονέκτημα της εύκολης συγγραφής του κώδικα
- αλλά, η κακή-άσκοπη χρήση τους μπορεί να οδηγήσει σε υπερχείλιση
- και στη υπερβολική καθυστέρηση στον χρόνο εκτέλεσης του προγράμματος

Για την δήλωση και την κατασκευή των αναδρομικών συναρτήσεων ισχύει ότι είπαμε παραπάνω για τις συναρτήσεις με επιστρεφόμενη τιμή.

Αναδρομικές συναρτήσεις II

Η μόνη και βασική διαφορά είναι ότι οι αναδρομικές συναρτήσεις περιέχουν μια συνθήκη τερματισμού.

Αναδρομικές συναρτήσεις I

Ας δούμε όμως ένα παράδειγμα για να καταλάβουμε καλύτερα τι είναι μία αναδρομική συνάρτηση:

Ο υπολογισμός του αθροίσματος

$$S_n = 1 + 2 + 3 + \ldots + n$$

υλοποιείται σειριακά με τον ακόλουθο τρόπο (με χρήση συνάρτησης)

```
int sumS(int n){
    int i, sum=0;
    for(i=0;i<n;i=i+1){
        sum=sum+i;
    }
    return(sum);
}
```


Αναδρομικές συναρτήσεις II

Θα μπορούσαμε να υπολογίσουμε το άθροισμα με τον εξής τρόπο

$$S_n = \begin{cases} 1 & n = 1 \\ n + S_{n-1} & n > 1 \end{cases}$$

ο οποίος υλοποιείται με τον ακόλουθο τρόπο (με χρήση συνάρτησης)

```
int sumR(int n){
    if (n==1){
        return(1);
    }
    else{
        return(sumR(n-1)+n);
    }
}
```

Αναδρομικές συναρτήσεις I

Παρατηρούμε λοιπόν ότι, και στις δυο περιπτώσεις έχουμε μια συνάρτηση που υπολογίζει το άθροισμα.

Μια διαφορά που έχουν μεταξύ τους είναι η **συνθήκη τερματισμού**. Δίνουμε μεγάλη βάση ώστε να υπάρχει πάντα συνθήκη τερματισμού στις αναδρομικές συναρτήσεις γιατί αλλιώς η συνάρτηση μας δεν θα τερματιστεί.

Μια άλλη διαφορά που έχουν μεταξύ τους είναι ο ξεκάθαρος κώδικας που έχει η αναδρομική συνάρτηση. Ο κώδικας ακολουθεί ακριβώς την μαθηματική μοντελοποίηση του προβλήματος.

Καθολικές μεταβλητές (global variables) I

Δηλώνονται πριν τη `main()`.

Εφαρμόζονται σε όλα τα τμήματα ενός προγράμματος.

Όταν μεταβάλλεται η τιμή μίας καθολικής μεταβλητής σε οποιοδήποτε σημείο του προγράμματος, η νέα τιμή μεταφέρεται σε όλο το υπόλοιπο πρόγραμμα.

Οι καθολικές μεταβλητές είναι μία κακή ιδέα, καθώς αποτρέπουν τον ξεκάθαρο μερισμό του προβλήματος σε ανεξάρτητα τμήματα.

Για μία τοπική μεταβλητή ο χώρος στη μνήμη δεσμεύεται μόλις ο έλεγχος περάσει στη συνάρτηση, αποδεσμεύεται δε με το τέλος αυτής, οπότε και η μεταβλητή δεν έχει πλέον νόημα.

Εμβέλεια μεταβλητών (scope) I

Εμβέλεια προγράμματος: μεταβλητές αυτής της εμβέλειας είναι οι καθολικές. Είναι ορατές από όλες τις συναρτήσεις του προγράμματος, έστω κι αν βρίσκονται σε διαφορετικά αρχεία πηγαίου κώδικα.

Εμβέλεια αρχείου: μεταβλητές αυτής της εμβέλειας είναι ορατές μόνο στο αρχείο που δηλώνονται και μάλιστα από το σημείο της δήλωσής τους και κάτω. Μεταβλητή που δηλώνεται με τη λέξη κλειδί `static` πριν από τον τύπο, έχει εμβέλεια αρχείου, π.χ. `static int velocity`.

Εμβέλεια συνάρτησης: Προσδιορίζει την ορατότητα του ονόματος από την αρχή της συνάρτησης έως το τέλος της. Εμβέλεια συνάρτησης έχουν μόνο οι `goto` ετικέτες.

Εμβέλεια μεταβλητών (scope) II

Εμβέλεια μπλοκ: Προσδιορίζει την ορατότητα από το σημείο δήλωσης έως το τέλος του μπλοκ στο οποίο δηλώνεται. Μπλοκ είναι ένα σύνολο από προτάσεις, οι οποίες περικλείονται σε άγκιστρα. Μπλοκ είναι η σύνθετη πρόταση αλλά και το σώμα συνάρτησης. Εμβέλεια μπλοκ έχουν και τα τυπικά ορίσματα των συναρτήσεων.

Η C επιτρέπει τη χρήση ενός ονόματος για την αναφορά σε διαφορετικά αντικείμενα, με την προϋπόθεση ότι αυτά έχουν διαφορετική εμβέλεια ώστε να αποφεύγεται η σύγκρουση ονομάτων (name conflict). Εάν οι περιοχές εμβέλειας έχουν επικάλυψη, τότε το όνομα με τη μικρότερη εμβέλεια αποκρύπτει (hides) το όνομα με τη μεγαλύτερη.

Διάρκεια μεταβλητών (duration) I

Η διάρκεια ορίζει το χρόνο κατά τον οποίο το όνομα της μεταβλητής είναι συνδεδεμένο με τη θέση μνήμης που περιέχει την τιμή της μεταβλητής.

Ορίζονται ως χρόνοι δέσμευσης και αποδέσμευσης οι χρόνοι που το όνομα συνδέεται με και αποσυνδέεται από τη μνήμη, αντίστοιχα.

Για τις καθολικές μεταβλητές δεσμεύεται χώρος με την έναρξη εκτέλεσης του προγράμματος και η μεταβλητή συσχετίζεται με την ίδια θέση μνήμης έως το τέλος του προγράμματος. Είναι πλήρους διάρκειας.

Διάρκεια μεταβλητών (duration) II

Οι τοπικές μεταβλητές είναι περιορισμένης διάρκειας. Η ανάθεση της μνήμης σε τοπική μεταβλητή γίνεται με τη είσοδο στο χώρο εμβέλειάς της και η αποδέσμευσή της με την έξοδο από αυτόν. Δηλαδή η τοπική μεταβλητή δε διατηρεί την τιμή της από τη μία κλήση της συνάρτησης στην επόμενη. Εάν όμως προστεθεί στη δήλωση μίας τοπικής μεταβλητής η λέξη static, διατηρεί την τιμή της και καθίσταται πλήρους διάρκειας.

Καθολικές μεταβλητές - Παράδειγμα I

```
#include <stdio.h>
#include <stdlib.h>

int global;
void function(void);
main(){
    global=10;
    printf("Global variable=%d", global);
    function();
    printf("Global variable=%d", global);
    return 0;
}

void function(void){
    global = global+1;
    printf("Global variable = %d", global);
}
```


Καθολικές μεταβλητές - Παράδειγμα II

Παράδειγμα 1: Συνάρτηση με επιστρεφόμενη τιμή I

Να δημιουργηθεί πρόγραμμα το οποίο θα:

- δημιουργεί μια συνάρτηση substraction
- δέχεται στο κυρίως πρόγραμμα δύο τιμές από τον χρήστη και θα κάνει έλεγχο για το αν οι τιμές είναι διάφορες του μηδενός (0)
- καλεί την συνάρτηση που δημιουργήσαμε για να πάρει τις τιμές και να βρει την διαφορά τους.

Παράδειγμα 1: Συνάρτηση με επιστρεφόμενη τιμή I

```
#include <stdio.h>

int substraction(int a, int b);

int main(){
    int x, y, difference;
    printf("Give me a number: ");
    scanf("%d", &x);
    printf("Give me a second number: ");
    scanf("%d", &y);
    if((x!=0) && (y!=0)){
        difference = substraction(x, y);
    }
    printf("Their difference is: ", difference);
    return 0;
```

Παράδειγμα 1: Συνάρτηση με επιστρεφόμενη τιμή II

```
}

int substraction(int a, int b){
    int dif;
    dif = a-b;
    return(dif);
}
```


Παράδειγμα 2: Συνάρτηση χωρίς επιστρεφόμενη τιμή I

Να δημιουργηθεί πρόγραμμα το οποίο θα:

- κάνει χρήση καθολικής μεταβλητής
- δημιουργεί μια συνάρτηση `global` η οποία δεν θα δέχεται καμία τιμή, αλλά θα αυξάνει την τιμή της καθολικής μεταβλητής κατά ένα (1) και θα την εμφανίζει στον χρήστη
- το κυρίως πρόγραμμα θα εκχωρεί μια τιμή στην καθολική μεταβλητή θα την εμφανίζει και μετά θα καλεί την συνάρτηση.

Παράδειγμα 2: Συνάρτηση χωρίς επιστρεφόμενη τιμή I

```
#include <stdio.h>
#include <stdlib.h>

int globalVariable;
int global(void);

int main(){
    globalVariable = 6;
    printf("Global variable is: %d", globalVariable);
    global();
    return 0;
}

int global(){
    globalVariable = globalVariable +1;
```


Παράδειγμα 2: Συνάρτηση χωρίς επιστρεφόμενη τιμή II

```
    printf("Global variable is: %d", globalVariable);
}
```

Παράδειγμα 3: Συνάρτηση με είσοδο πίνακα

Να δημιουργηθεί πρόγραμμα το οποίο θα:

αρχικοποιεί τις τιμές ενός πίνακα και θα τις εμφανίζει

χρησιμοποιεί μια συνάρτηση χωρίς έξοδο η οποία θα διαβάζει τις τιμές του πίνακα και θα τις εμφανίζει

και θα εμφανίζει (στην main) τις τιμές του πίνακα.

Παράδειγμα 3: Συνάρτηση με είσοδο πίνακα I

```
#include <stdio.h>
#include <stdlib.h>

void fun3(int A[3]);

int main(){
    int A[3]={1,1,1};
    printf("A[0]=%d, A[1]=%d, A[2]=%d\n\n",A[0],A[1],A
        [2]);
    fun3(A);
    printf("A[0]=%d, A[1]=%d, A[2]=%d\n\n",A[0],A[1],A
        [2]);
    return 0;
}

void fun3(int A[3]){
    int i;
```

Παράδειγμα 3: Συνάρτηση με είσοδο πίνακα II

```
    for(i=0; i<3; i=i+1) {
        printf("Give A[%d] = ", i);
        scanf("%d",&A[i]);
    }
    printf("FUN: A[0]=%d, A[1]=%d, A[2]=%d\n\n",A[0],A
        [1],A[2]);
}
```


Παράδειγμα 3: Συνάρτηση με είσοδο πίνακα

Στο παραπάνω πρόγραμμα παρατηρούμε ότι, ο πίνακας `A` έχει αρχικές τιμές οι οποίες αλλάζουν μέσα στην συνάρτηση `fun3` και μεταφέρονται `main`.

Η χρήση πίνακα είναι ένας εναλλακτικός τρόπος να μας επιστρέψει μια συνάρτηση περισσότερες από μια τιμές.

Για να επιστρέψουμε τιμές μέσω πίνακα από μια συνάρτηση πρέπει ο πίνακας να είναι όρισμα (μεταβλητή εισόδου) της συνάρτησης.

Παράδειγμα 4: Δημιουργία αναδρομικής συνάρτησης I

Να δημιουργηθεί πρόγραμμα το οποίο θα:

υπολογίζει την ακολουθία **Fibonacci** κάνοντας χρήση αναδρομικής συνάρτησης

$$F_n = \begin{cases} 0 & n = 1 \\ 1 & n = 2 \\ F_{n-1} + F_{n-2} & n > 2 \end{cases}$$

καλεί και θα εμφανίζει τα αποτελέσματα μέσα από την `main`

Παράδειγμα 4: Δημιουργία αναδρομικής συνάρτησης II

Παράδειγμα 4: Δημιουργία αναδρομικής συνάρτησης III

της αρχικής συνάρτησης ώστε να φτάσουμε στο ζητούμενο αποτέλεσμα χρησιμοποιούμε αναδρομικές συναρτήσεις.

Παράδειγμα 4: Δημιουργία αναδρομικής συνάρτησης I

```
#include <stdio.h>
#include <stdlib.h>

int fibonacci(int n);

int main(){
    int n, i;
    scanf("%d",&n);
    printf("Fibonacci series\n");
    for(i=1; i<=n; i=i+1){
        printf("F(%d)=%d\n", i, fibonacci(i));
    }
    return 0;
}
```

Παράδειγμα 4: Δημιουργία αναδρομικής συνάρτησης II

```
int fibonacci(int n){
    if(n==1){
        return 0;
    }
    else if(n==2){
        return 1;
    }
    else{
        return(fibonacci(n-1) + fibonacci(n-2));
    }
}
```