# Η ΓΛΩΣΣΑ C

## Μάθημα 23:

Δείκτες σε Συναρτήσεις

Δημήτρης Ψούνης

# Περιεχόμενα Μαθήματος

## Α. Πίνακες

1. 1. Δείκτες σε Συναρτήσεις
    1. Υπογραφή Συνάρτησης
    2. Ορισμός Δείκτη σε Συνάρτηση
    3. Διαχείριση Συνάρτησης μέσω Δείκτη
    4. Παράδειγμα
    5. Συνάρτηση που παίρνει ως όρισμα δείκτη σε συνάρτηση
    6. Πίνακες Συναρτήσεων
    7. Δείκτες Συναρτήσεων και typedef
    8. … και μία εισαγωγή στη C++!
    Πολυδιάστατοι Πίνακες

## Β. Ασκήσεις

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 1. Υπογραφή Συνάρτησης

- Μία συνάρτηση καταλαμβάνει χώρο στη μνήμη.
  - Και συγκεκριμένα αποθηκεύει τις εντολές (σε γλώσσα μηχανής) από τις οποίες αποτελείται.
- Έτσι μπορούμε να βάλουμε έναν δείκτη να δείξει στην διεύθυνση της συνάρτησης.

- Η συνάρτηση έχει μια υπογραφή (όπως λέμε ότι μια μεταβλητή έχει κάποιο τύπο δεδομένων)
  - Η υπογραφή καθορίζεται από:
    - Τα ορίσματα της συνάρτησης
    - και τον τύπο επιστροφής της
  - Η υπογραφή είναι σημαντική γιατί:
    - Ένας δείκτης σε συνάρτηση θα καθορίζεται από την υπογραφή της συνάρτησης
  - Δηλαδή
    - όπως ένας δείκτης σε μεταβλητή, καθορίζεται από τον τύπο δεδομένων της μεταβλητής.
    - έτσι ένας δείκτης σε συνάρτηση, καθορίζεται από την υπογραφή της συνάρτησης.

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 2. Ορισμός Δείκτη σε Συνάρτηση

- Ο ορισμός δείκτη σε συνάρτηση καθορίζεται από την υπογραφή της
- Έτσι π.χ. η δήλωση ενός δείκτη σε μια συνάρτηση που έχει ορίσματα δύο ακέραιες μεταβλητές και επιστρέφει ακέραια μεταβλητή γίνεται ως εξής:

```
int (*ptr_name) (int, int)
```

  - Προσοχή ότι οι παρενθέσεις που περιβάλλουν το όνομα του δείκτη είναι υποχρεωτικές.

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 3. Διαχείριση Συνάρτησης μέσω Δείκτη

- Από τη στιγμή που έχουμε μια συνάρτηση, π.χ.:

```
int sum(int a, int b)
```

- Μπορούμε να δηλώσουμε έναν δείκτη που θέλουμε να δείξει σε αυτήν τη συνάρτηση:

```
int (*ptr)(int, int)
```

- Έπειτα βάζουμε το δείκτη να δείχνει στη συνάρτηση:

```
ptr = sum;
```

- Και στη συνέχεια μπορούμε να καλέσουμε τη συνάρτηση και μέσω του δείκτη:

```
ptr(3,4);
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 4. Παράδειγμα

- Βλέπουμε και ολοκληρωμένο το παράδειγμα:

```
/* pointer_to_function.c Deiktis se sinartisi */

#include <stdio.h>

int sum(int a, int b);

main()
{
   int (*ptr)(int, int);

   ptr=sum;

   printf("%d", ptr(1,2));
}

int sum(int a, int b)
{
   return a+b;
}
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 4. Παράδειγμα

- Και ένα ακόμη παράδειγμα στο οποίο βάζουμε το δείκτη να δείχνει σε διαφορετικές συναρτήσεις ανάλογα με την είσοδο του χρήστη:

```
/* pointer_to_diff_functions.c  */

#include <stdio.h>

int add(int a, int b);
int sub(int a, int b);
int mult(int a, int b);

main()
{
   int (*ptr)(int, int);
   int x,y, choice;

 printf("Dwse x: ");
   scanf("%d", &x);
   printf("Dwse y: ");
   scanf("%d", &y);
   printf("Dwse sinartisi (1-add, 2-sub, 3-mult): ");
   scanf("%d", &choice);
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 4. Παράδειγμα

- Και ένα ακόμη παράδειγμα στο οποίο βάζουμε το δείκτη να δείχνει σε διαφορετικές συναρτήσεις ανάλογα με την είσοδο του χρήστη:

```
switch(choice)
  {
     case 1:
        ptr=add;
        break;
     case 2:
        ptr=sub;
        break;
     case 3:
        ptr=mult;
  }

  printf("result=%d", ptr(x,y));
}
```

```
int add(int a, int b)
{
   return a+b;
}

int sub(int a, int b)
{
   return a-b;
}

int mult(int a, int b)
{
   return a*b;
}
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 5. Συνάρτηση που παίρνει ως όρισμα δείκτη σε συνάρτηση

- Ένας δείκτης σε συνάρτηση μπορεί όπως κάθε άλλη μεταβλητή να μπει ως όρισμα συνάρτηση.
- Για παράδειγμα αν έχουμε ένα δείκτη σε συνάρτηση:

```
int (*ptr_name) (int, int)
```

  - μπορούμε να τη διοχετεύσουμε ως όρισμα σε μια συνάρτηση:

```
void func(int (*ptr_name) (int, int));
```

- Στο παράδειγμα της ακόλουθης διαφάνειας, ορίζουμε
  - Μία συνάρτηση που επιστρέφει το αποτέλεσμα της συνάρτησης που δέχεται ως όρισμα.

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 5. Συνάρτηση που παίρνει ως όρισμα δείκτη σε συνάρτηση

```
/* function_with_arg_pointer.c  */

#include <stdio.h>

int inc(int x);
int dec(int x);
int half(int x);

int func(int (*ptr1)(int), int arg);

main()
{
   printf("result=%d", func(inc, 2));
}
```

```
int inc(int x)
{
   return ++x;
}

int dec(int x)
{
   return --x;
}

int half(int x)
{
   return x/2;
}

int func(int (*ptr1)(int), int arg)
{
   return ptr1(arg);
}
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 6. Πίνακες Συναρτήσεων

- Μπορούμε να ορίσουμε και πίνακες συναρτήσεων (εφόσον αυτές έχουν την ίδια υπογραφή)
- Η δήλωση του πίνακα γίνεται με την εντολή:

```
int (*ptr_name[N]) (int, int)
```

- Μπορούμε έπειτα να έχουμε πρόσβαση σε κάθε δείκτη ως συνήθως (για πίνακα):

```
ptr_name[i]=…
```

- Καθώς και να καλέσουμε μια συνάρτηση του πίνακα με ορίσματα, π.χ.

```
ptr_name[i] (4, 5)
```

- Βλέπουμε και ένα ολοκληρωμένο παράδειγμα:

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 6. Πίνακες Συναρτήσεων

```
/* array_of_functions.c  */

#include <stdio.h>

#define N 3

int inc(int x);
int dec(int x);
int half(int x);

main()
{
   int (*ptr [N])(int);
   int i;

   ptr[0]=inc;
   ptr[1]=dec;
   ptr[2]=half;
```

```
  for (i=0; i<N; i++)
      printf("%d ", ptr[i](3));

}

int inc(int x)
{
   return ++x;
}

int dec(int x)
{
   return --x;
}

int half(int x)
{
   return x/2;
}
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 7. Δείκτες Συναρτήσεων και typedef

- Ολοκληρώνουμε με τη χρήση της typedef για να απλοποιήσουμε το συντακτικό των δεικτών σε συναρτήσεις.
- Η εντολή είναι :

```
typedef int (*pf) (int, int);
```

  - Το όνομα (pf) είναι δική μας επιλογή.
  - όπου έχει δηλωθεί ότι το pf είναι συνώνυμο δείκτη σε συνάρτηση με δύο ορίσματα ακεραίων και επιστροφής ακέραιου

- Έπειτα μπορούμε να δηλώσουμε έναν δείκτη με την εντολή:

```
pf ptr;
```

- που είναι ισοδύναμος με τη δήλωση:

```
int (*ptr) (int, int);
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 7. Δείκτες Συναρτήσεων και typedef

```
/* typedef_pointer_function.c  */

#include <stdio.h>

typedef int (*pf) (int, int);

int add(int a, int b);
int sub(int a, int b);
int mult(int a, int b);

main()
{
   int x,y, choice;
   pf ptr;

   printf("Dwse x: ");
   scanf("%d", &x);
   printf("Dwse y: ");
   scanf("%d", &y);
   printf("Dwse sinartisi (1-add, 2-sub, 3-mult): ");
   scanf("%d", &choice);

   switch(choice)
   {
      case 1:
         ptr=add;
         break;
      case 2:
         ptr=sub;
         break;
      case 3:
         ptr=mult;
   }

   printf("result=%d", ptr(x,y));
}

int add(int a, int b)
{
   return a+b;
}

...
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 8… και μία εισαγωγή στη C++!

- (Μιας και φτάνουμε στο τέλος της C)
- Το επόμενο βήμα μετά τη C είναι η C++!
  - Το πιο σημαντικό νέο χαρακτηριστικό της C++ είναι οι κλάσεις
  - οι οποίες ομαδοποιούν δεδομένα και συναρτήσεις που επενεργούν πάνω σε αυτές.
- Με χρήση των δεικτών σε συναρτήσεις μπορούμε να προσομοιώσουμε αυτήν τη λειτουργία (αντικειμενοστρέφεια)
- Θα δούμε ένα πολύ απλό παράδειγμα.
  - Την δομή μιγαδικός αριθμός
  - Η οποία έχει ως μέλη τις ακέραιες τιμές του πραγματικού και του φανταστικού μέρους.
  - Και επίσης ως μέλος μία συνάρτηση εκτύπωσης.
- Βλέπουμε ένα στοιχειώδες παράδειγμα:

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 8… και μία εισαγωγή στη C++!

```
/* class.c  */

#include <stdio.h>

typedef void (*pf) (int, int);

void print(int x, int y);

struct complex {
  int x;
  int y;
  pf print;
};

main()
{
   struct complex a,b;
   a.x=1;
   a.y=1;
   a.print=print;
   b.x=2;
   b.y=2;
   b.print=print;

   a.print(a.x,a.y);
   a.print(b.x,b.y);
}

void print(int x, int y)
{
            printf("(%d,%d) ",x,y);
}
```

# Α. Πίνακες

## 1. Δείκτες σε Συναρτήσεις

### 8… και μία εισαγωγή στη C++!

- Το πρόγραμμα μας δίνει μία αίσθηση της C++
  - Για τη σημασία της ομαδοποίησης δεδομένων και συναρτήσεων!

- Ωστόσο η C++ έχει πολύ πιο εύκολο συντακτικό σε σχέση με αυτό που είδαμε και έτσι εκεί θα μπορούμε να γράφουμε πολύ πιο κομψά π.χ.:

```
a.x = 1;
a.y = 1;
a.print();
```

- Περισσότερα όταν τελειώσουμε (με το καλό) τη C!

# Β. Ασκήσεις
## 1. Συνάρτηση εντοπισμού ρίζας

Γράψτε ένα πρόγραμμα το οποίο:

1. Δηλώνει μια συνάρτηση f της αρεσκείας σας (π.χ. f(x)=2x) με ΤΔ ορίσματος και επιστροφής double
2. Δηλώνει μία συνάρτηση root η οποία:
   - Παίρνει ως ορίσματα:
     - Έναν δείκτη σε συνάρτηση
     - Αρχή και πέρας διαστήματος [α,β]
   - Αν a είναι ρίζα, να το τυπώνει
   - Αν b είναι ρίζα, να το τυπώνει
   - Αν f(a) και f(b) είναι ομόσημοι, τότε να τυπώνει ότι πιθανόν δεν υπάρχει ρίζα.
   - Αν f(a) και f(b) είναι ετερόσημοι, τότε να εξετάζει την τιμή f((a+b)/2) και να επαναλαμβάνει την διαδικασία στο διάστημα που περιέχει τη ρίζα
     - εωσότου το διάστημα είναι αρκούντως μικρό (π.χ. <0.05)
     - οπότε να επιστρέφει το μέσο του διαστήματος
   - [Σημείωση: Όλες οι χρήσεις συνάρτησης μέσα στη root να γίνουν μέσω του δείκτη που θέσαμε ως όρισμα]

# Β. Ασκήσεις

## 2.1. Συνάρτηση compare

Υλοποιήστε μία συνάρτηση compare_asc (αύξουσα σειρά) η οποία με ορίσματα δύο ακεραίους:

- -1, αν ο 1ος είναι μικρότερος του 2ου
- 0, αν είναι ίσοι
- 1. αν ο 1ος είναι μεγαλύτερος του 2ου

Υλοποιήστε μία συνάρτηση compare_desc (φθίνουσα σειρά) η οποία με ορίσματα δύο ακεραίους:

- 1, αν ο 1ος είναι μικρότερος του 2ου
- 0, αν είναι ίσοι
- -1. αν ο 1ος είναι μεγαλύτερος του 2ου

# Β. Ασκήσεις
## 2.2. Συνάρτηση insertion_sort

Στο μάθημα «Αλγόριθμοι σε C – Μάθημα 3: Ταξινόμηση Πίνακα» είδαμε τον αλγόριθμο ταξινόμησης με εισαγωγή (insertion sort)

```
/* Taksinomisi me Eisagogi */
for (i=1; i<N; i++)
{
   for (j=i; j>=1; j--)
   {
      if (pinakas[j]<pinakas[j-1])
         swap(&pinakas[j], &pinakas[j-1]);
      else
         break;
   }
}
```

Κατασκευάστε την συνάρτηση insertion_sort η οποία:

- Θα παίρνει ως όρισμα τον πίνακα, τη διάστασή του, αλλά και τη μέθοδο σύγκρισης δύο αριθμών
- Θα ταξινομεί τον πίνακα με τον παραπάνω αλγόριθμο.

Κατασκευάστε συνάρτηση main η οποία:

- Θα ταξινομεί τον πίνακα (πειραματιστείτε με τις μεθόδους σύγκρισης ως όρισμα)