Δομημένος Προγραμματισμός

Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων

www.bpis.teicrete.gr

Δομές Δεδομένων

- Μέχρι τώρα έχουμε δει μεταβλητές που ο τύπος τους είναι ένας από τους βασικούς τύπους που υποστηρίζει η C (π.χ. int, char, float κλπ.)

- Μια σχετική εξαίρεση ήταν οι Πίνακες που μπορούμε να αποθηκεύσουμε δεδομένα (ενός συγκεκριμένου τύπου) σε μια ακολουθιακή μορφή

- Δομή – Structure
  νέα μορφή μεταβλητής
  - Επιτρέπει να αποθηκευτούν δεδομένα διαφορετικών τύπων που έχουν κάποια «λογική» σχέση μεταξύ τους
  - Είναι πιο δομημένη από την απλή ακολουθιακή μορφή Πινάκων

```
struct όνομα_προτύπου
{
        τύπος_πεδίου_1    όνομα_πεδίου_1;
        τύπος_πεδίου_2    όνομα_πεδίου_2;
        ...
        τύπος_πεδίου_ν    όνομα_πεδίου_ν;
};
```

```
struct date
{
        int day;
        int month;
        int year;
};
```

```
struct person
{
        char name[50];
        int age;
        float height;
};
```

```
struct company
{
        char name[50];
        int start_year;
        int field;
        int tax_num;
        int num_empl;
        char addr[50];
        float balance;
};
```

Παράδειγμα

Δήλωση Δομής:

```
struct όνομα_προτύπου όνομα_δομής_1,..., όνομα_δομής_ν;
```

Παραδείγματα:

```
struct person person_1;
```

```
struct company comp_1, comp_2;
```

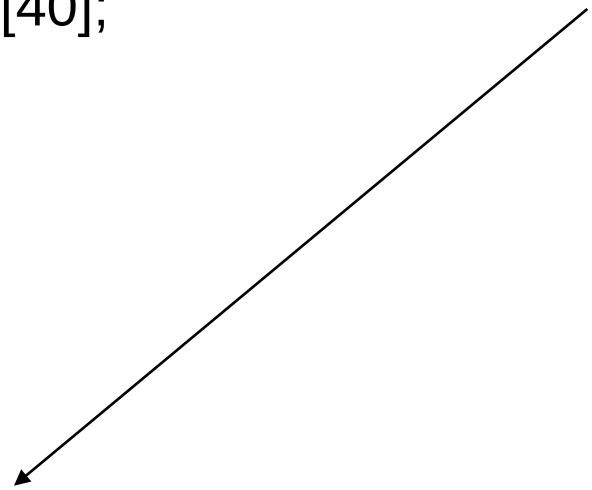
```
struct book
{
        char title[100];
        int year;
        float price;
} book_1, book_2;
```

Παράδειγμα

```
struct {
    int number;
    int sex;
    char name[10];
    char address[40];
    int age;
} person1;
```

Διαφορετικές εμβέλειες

```
struct {
    int number;
    int name[10];
    int yearsOfEmployment;
} employee1;
```



Αρχικές τιμές σε Δομές

Όπως και στους πίνακες, έτσι και μία δομή μπορεί να πάρει αρχικές τιμές κατά τη δήλωση της π.χ.

```
struct {
  int number;
  int sex;
  char name[11];
} person1 = {154, 1, “John Smith”},
  person2 = {180, 2, “Mary Jones”};
```

154

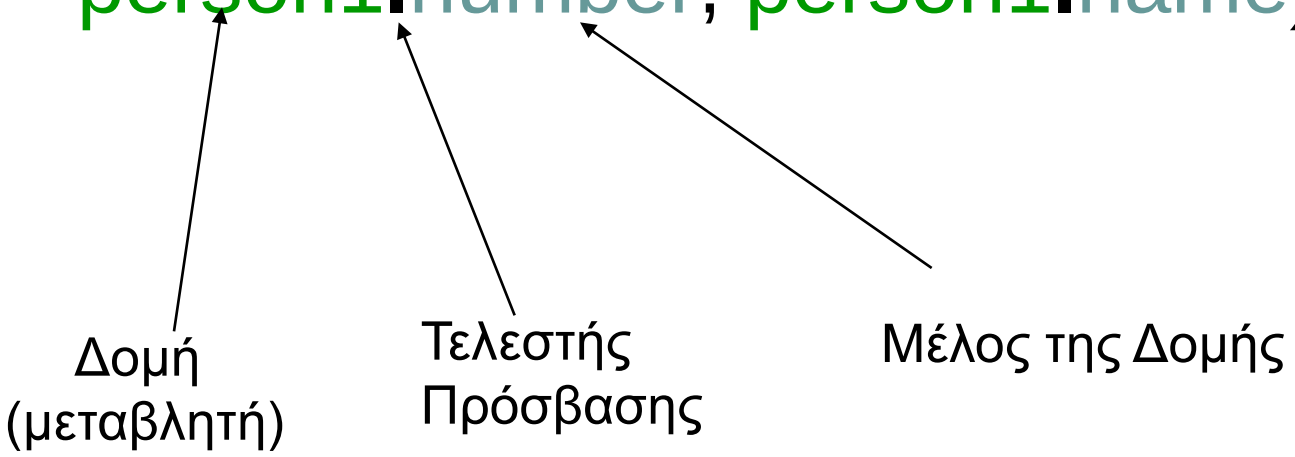
1

John Smith

Λειτουργίες σε Δομές

Όπως και στους πίνακες, έτσι και στις δομές, η πιο κοινή λειτουργία είναι να βρούμε κάποιο στοιχείο της π.χ.

```
printf("The name of person %d is %s",
        person1.number, person1.name);
```



Παράδειγμα

scanf(“%d”, &pesron1.number);

Διαβάζουμε το μέλος number

person1 = person2;

Αποθέτουμε τις τιμές των μελών της person2, στην person1

Σημείωση: Αυτό δεν ήταν εφικτό στους πίνακες

Δήλωση Τύπου Δομής

Θα θέλαμε να έχουμε την ευχέρεια να δηλώσουμε τον δικό μας τύπο και απλά να χρησιμοποιούμε το όνομά του όπως θα χρησιμοποιούσαμε int, float, κλπ

Αυτό επιτυγχάνεται με την εντολή **typedef**

Ένα πρότυπο δομής μπορεί να δηλωθεί εναλλακτικά με χρήση του προσδιοριστικού **typedef**

Σε αυτή την περίπτωση το όνομα του προτύπου εισάγεται μετά το δεξί άγκιστρο, π.χ.

```
typedef struct
{
        char title[100];
        int year;
        float price;
} book;
```

Αν το πρότυπο μίας δομής έχει δηλωθεί με χρήση της εντολής **typedef**, τότε δεν προσθέτουμε τη λέξη **struct** πριν από τη δήλωση μίας δομής

Δηλ. αν είχαμε δηλώσει την παραπάνω δομή (book), θα μπορούσαμε να δηλώσουμε τις μεταβλητές book_1 και book_2 ως εξής:

```
book book_1, book_2;
```

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

struct1

Άσκηση

Στον κώδικα που σας δίνεται, ορίστε μία μεταβλητή **person,** τύπου δομής με τρία πεδία:

- **id : κωδικός**
- **name : όνομα**
- **age : ηλικία**

Αρχικοποιήστε την κατά την δήλωση της (με χρήση των { }) με τα στοιχεία σας και τον κωδικό 10 και τυπώστε την στη οθόνη

Άσκηση

Στο προηγούμενο παράδειγμα ορίστε ένα νέο τύπο δεδομένων (με **typedef**) για την δομή σας με όνομα customer.

Χρησιμοποιήστε τον τύπο customer, για να αρχικοποιήσετε 2 μεταβλητές **person1** και **person2** και τυπώστε τις μεταβλητές.

Καταχωρήστε τις τιμές της person2 στην person1 και τυπώστε την person1

Πίνακες Δομών

Προφανώς μπορούμε να έχουμε διανύσματα που τα στοιχεία τους είναι δομές !! Π.χ.

```
typedef struct {
        char firstName[10];
        char lastName[10];
} personType;

personType aDataBase[100];
```

Διάνυσμα 100 Δομών Τύπου PersonType

Οπότε μπορούμε να γράψουμε

```
aDataBase[30].fisrtName[0] = '\0'    /* Άδεια συμβολοσειρά */
```

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

struct2

Άσκηση

Ορίστε τον τύπο δεδομένων customer καθολικά (στην αρχή του κώδικα σας)

Επεκτείνετε το πρόγραμμα σας ώστε να έχετε ένα πίνακα 3 θέσεων τύπου customer

Γράψτε μία συνάρτηση που θα τυπώνει τον πίνακα

Συνδυασμός Δομών

Οι δομές μπορούν να συνδυασθούν χωρίς περιορισμούς π.χ.

```
typedef struct {
        char firstName[10];
        char lastName[10];
} personType;

struct student {
   personType aPerson;
   int id;
   int age;
   int sex;
} student1, student2;
```

Οπότε μπορούμε να γράψουμε

strcpy(student1.aPerson.firstName, “John”);

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

struct3

Άσκηση

Ορίστε τον τύπο δεδομένων δομής **personType** με δύο μέλη

πίνακα 10 χαρακτήρων **firstName**

πίνακα 10 χαρακτήρων **lastName**

Ορίστε τον τύπο δεδομένων **student** με τρία μέλη

μεταβλητή **aPerson** τύπου personType

μεταβλητή τύπου ακεραίου με όνομα **AM**

Ορίστε δύο **καθολικές** μεταβλητές **student1** και **student2**

Αρχικοποιήστε τις μία μεταβλητή (**student1**) με τα στοιχεία σας και το ΑΜ σας και την άλλη μεταβλητή (**student2**) με τα αντίστοιχα στοιχεία ενός συμφοιτητή σας

Άσκηση

```
typedef struct {
        char firstName[10];
        char lastName[10];
} personType;

typedef  struct {
   personType aPerson;
   int AM;
} student;


...
student student1, student2;
```

Άσκηση

Αλλάξτε το πρόγραμμα σας ώστε τα δεδομένα για τις δύο μεταβλητές να δίνονται από το πληκτρολόγιο

Δημιουργήστε μία συνάρτηση **void insert_student();** με την οποία εισάγονται τα δεδομένα στη δομή σας

Δημιουργήστε μία συνάρτηση **void print_student()** που τυπώνει τις δύο μεταβλητές στην εξής μορφή

onoma : Yannis

epwnymo: Aikaterinidis

A.M. : 100

Άσκηση

Επεκτείνετε το πρόγραμμα σας ώστε πλέον να μην έχετε δύο μεταβλητές student1 και student2, αλλά ένα πίνακα 10 θέσεων.

Αλλάξτε τη συνάρτηση εισόδου **insert_student()** ώστε

να εισάγει ο χρήστης τον αριθμό Ν των φοιτητών

και να συμπληρώνονται κατάλληλα από το πληκτρολόγιο οι Ν πρώτες θέσεις με τα στοιχεία των φοιτητών

Αλλάξτε τη συνάρτηση εξόδου **print_student()** ώστε

να τυπώνονται τα στοιχεία των Ν φοιτητών

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

student

Άσκηση

Γράψτε ένα πρόγραμμα που θα δέχεται από το πληκτρολόγιο το πλήθος των φοιτητών που θα βαθμολογηθούν.

Στη συνέχεια δέχεται από το πληκτρολόγιο τους βαθμούς 5 μαθημάτων για κάθε φοιτητή. Οι βαθμοί είναι πραγματικοί αριθμοί με εύρος τιμών από 0 έως 10.

Αφού αποθηκευτούν οι βαθμοί σε ένα πίνακα δύο διαστάσεων, υπολογίστε και τυπώστε στην οθόνη το μέσο όρο κάθε φοιτητή.

Άσκηση (Παραλλαγή με δομές)

Αλλάξτε τον κώδικα σας ώστε για κάθε φοιτητή να διατηρείτε ένα πίνακα 5 θέσεων για τους βαθμούς των μαθημάτων και μία μεταβλητή **mo** τύπου float με το μέσο όρο των μαθημάτων.

Δημιουργήστε μία δομή με δύο μέλη (πίνακας βαθμών και μέσος όρος) και ορίστε ένα πίνακα 100 θέσεων με αυτή τη δομή. Θα χρησιμοποιήσετε τις Ν πρώτες θέσεις (όπου το Ν το εισάγει ο χρήστης από το πληκτρολόγιο)

Δημιουργήστε τρεις συναρτήσεις
εισαγωγή βαθμών (**insertGrades**)
υπολογισμός και εκτύπωση μέσου όρου (**printMO**)
εκτύπωση βαθμών (**printGrades**)

Εισάγετε βαθμούς για 3 φοιτητές, και καλέστε τη συνάρτηση εκτύπωσης των μέσων όρων, και τη συνάρτηση εκτύπωσης των βαθμών για τον 1ο και τον 3ο φοιτητή

Αναδρομικές Συναρτήσεις

Μία συνάρτηση ονομάζεται αναδρομική όταν μία εντολή του σώματος της συνάρτησης καλεί τον ίδιο της τον εαυτό.

Έστω ότι μία αναδρομική συνάρτηση καλείται να λύσει ένα πρόβλημα. Μία τέτοια συνάρτηση δύναται να λύσει μόνο την απλούστερη περίπτωση, τη λεγόμενη **βάση της αναδρομής** (base case). Εάν η περίπτωση είναι πολύπλοκη, το πρόβλημα μερίζεται σε ένα ή περισσότερα υποπροβλήματα, τα οποία μοιάζουν με το αρχικό πρόβλημα αλλά αποτελούν μικρότερες εκδοχές του. Η αναδρομική συνάρτηση καλεί τον εαυτό της για την επίλυση των υποπροβλημάτων. Αυτή είναι μία **αναδρομική κλήση** ή **αναδρομικό βήμα** (recursion step). Η διαδικασία συνεχίζεται έως ότου ο μερισμός σε υποπροβλήματα οδηγήσει στη βάση της αναδρομής, η οποία επιλύεται άμεσα. Κατόπιν ακολουθείται η αντίστροφη διαδικασία, επιλύοντας αρχικά τα μικρότερα υποπροβλήματα και προχωρώντας προς τα μεγαλύτερα.

```c
#include <stdio.h>

void show(int num);

int main()
{
	int num;

	printf("Enter number: ");
	scanf("%d",&num);

	show(num);
	return 0;
}

void show(int num)
{
	if(num > 1)
		show(num-1);

	printf("val = %d\n",num);
}
```

Αν ο χρήστης πληκτρολογήσει 3:

Έξοδος:
```
val = 1
val = 2
val = 3
```

Διότι, όταν μία συνάρτηση καλεί τον εαυτό της, οι επόμενες εντολές του σώματός της καθώς και οι τιμές των εμπλεκόμενων μεταβλητών (οι οποίες αποθηκεύονται) παραμένουν στη μνήμη.

Όταν η συνάρτηση σταματήσει να καλεί τον εαυτό της, οι αποθηκευμένες εντολές εκτελούνται με αντίστροφη σειρά (δηλ. από την τελευταία προς την πρώτη)

Παράδειγμα Άθροισμα αριθμών από 1 έως N

Εάν το **n** είναι ίσο με **1**, τότε το άθροισμα ταυτίζεται με το **n (βάση της αναδρομής).**

Στη γενική περίπτωση, ο υπολογισμός του αθροίσματος **n** μπορεί να θεωρηθεί ως υπολογισμός του αθροίσματος των αριθμών από το **1** έως το **n-1** συν το **n**.

Αντίστοιχα, ο υπολογισμός του αθροίσματος **n-1** μπορεί να θεωρηθεί ως υπολογισμός του αθροίσματος των αριθμών από το **1** έως το **n-2** συν το **n-1**.

Παράδειγμα
Άθροισμα αριθμών από 1 έως N

Χωρίς αναδρομή

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

Με αναδρομή

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

Παράδειγμα Άθροισμα αριθμών από 1 έως Ν

Οι διαδοχικές κλήσεις για **n=4** :

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

sum(4) καλεί τη sum(3)

sum(3) καλεί τη sum(2)

sum(2) καλεί τη sum(1)

η sum(1) δίνει αποτέλεσμα 1 και το επιστρέφει στη sum(2)

η sum(2) δίνει αποτέλεσμα 1+2=3 και το επιστρέφει στη sum(3)

η sum(3) δίνει αποτέλεσμα 3+3=6 και το επιστρέφει στη sum(4)

η sum(4) δίνει αποτέλεσμα 6+4=10, το οποίο είναι και το τελικό

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

recursive

Άσκηση

Δημιουργήστε μία αναδρομική συνάρτηση που υπολογίζει το x^y

Παρατήρηση:

$x^y = x^{y-1} * x$

Πλεονεκτήματα των αναδρομικών συναρτήσεων

Το βασικότερο πλεονέκτημα των αναδρομικών συναρτήσεων είναι ότι μπορούν να χρησιμοποιηθούν για να δημιουργηθούν καθαρότερες και απλούστερες εκδοχές πολλών αλγορίθμων.

Δημιουργείται συμπαγέστερος κώδικας και είναι ιδιαίτερα χρήσιμες σε αναδρομικώς οριζόμενα δεδομένα όπως οι λίστες και τα δένδρα.

Μειονεκτήματα των αναδρομικών συναρτήσεων

Οι περισσότερες αναδρομικές συναρτήσεις δεν εξοικονομούν σημαντικό μέγεθος κώδικα ή μνήμης για τις μεταβλητές.

Οι αναδρομικές εκδοχές των περισσότερων συναρτήσεων μπορεί να εκτελούνται κάπως πιο αργά από τα επαναληπτικά τους ισοδύναμα εξαιτίας των πρόσθετων κλήσεων σε συναρτήσεις. Η πιθανή μείωση όμως δεν είναι αξιοσημείωτη.

Υπάρχει μικρή πιθανότητα οι πολλές αναδρομικές κλήσεις μίας συνάρτησης να προκαλέσουν υπερχείλιση της στοίβας (stack overflow), επειδή ο χώρος αποθήκευσης των παραμέτρων και των τοπικών μεταβλητών της συνάρτησης είναι στη στοίβα και κάθε νέα κλήση παράγει ένα νέο αντίγραφο αυτών των μεταβλητών.

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

recursive2

Άσκηση

Δημιουργήστε μία αναδρομική συνάρτηση που υπολογίζει το παραγοντικό ενός αριθμού

x! = 1*2*3*4…..*(x-1)*x

Δείκτης στη C

Μεταβλητή δείκτη (**pointer variable**) ή δείκτης (**pointer**) είναι μία μεταβλητή που διαχειρίζεται τιμές που αντιστοιχούν σε μία διεύθυνση της κύριας μνήμης.

Το όνομα μιας μεταβλητής αναφέρεται σε μία τιμή
→ **άμεση** αναφορά στη τιμή

Το όνομα μιας μεταβλητής δείκτη αναφέρεται στη διεύθυνση μνήμης μίας τιμής
→ **έμμεση** αναφορά στη τιμή

Η μνήμη RAM (Random Access Memory) ενός υπολογιστή αποτελείται από πολλές χιλιάδες θέσεις αποθήκευσης δεδομένων που έχουν διαδοχική αρίθμηση

Κάθε θέση ή κελί μνήμης προσδιορίζεται από μία μοναδική διεύθυνση

Η διεύθυνση της κάθε θέσης μνήμης είναι ένας αύξοντας αριθμός με τιμή που κυμαίνεται από το 0 έως μία μέγιστη τιμή (η οποία εξαρτάται από το μέγεθος της διαθέσιμης μνήμης στον συγκεκριμένο υπολογιστή)

Το περιεχόμενο της κάθε θέσης μνήμης είναι ένας ακέραιος αριθμός με μέγεθος 1 byte

Όταν δηλώνεται μία μεταβλητή, ο μεταγλωττιστής δεσμεύει τις απαραίτητες συνεχόμενες θέσεις (bytes) στη μνήμη, για να αποθηκεύσει την τιμή της

Όπως ήδη ξέρουμε, κάθε τύπος μεταβλητής απαιτεί συγκεκριμένο χώρο στη μνήμη

Π.χ. ο τύπος **`char`** απαιτεί 1 byte μνήμης, οι τύποι **`float`** και **`int`** απαιτούν 4 bytes, ο τύπος **`double`** απαιτεί 8 bytes, κ.ο.κ.

Όταν μία μεταβλητή καταλαμβάνει πολλές θέσεις μνήμης (δηλ. περισσότερα από 1 byte), τότε ως διεύθυνση της μεταβλητής θεωρείται η διεύθυνση της πρώτης θέσης μνήμης (δηλ. του 1ου byte από τα bytes που καταλαμβάνει η μεταβλητή)

Έστω η δήλωση: `int a;`

Τότε: Ο μεταγλωττιστής ψάχνει και βρίσκει 4 συνεχόμενες θέσεις μνήμης στη RAM, οι οποίες δεν πρέπει να έχουν δεσμευτεί για άλλη μεταβλητή, και τις δεσμεύει

Σε αυτές τις θέσεις θα αποθηκεύεται η τιμή της μεταβλητής a

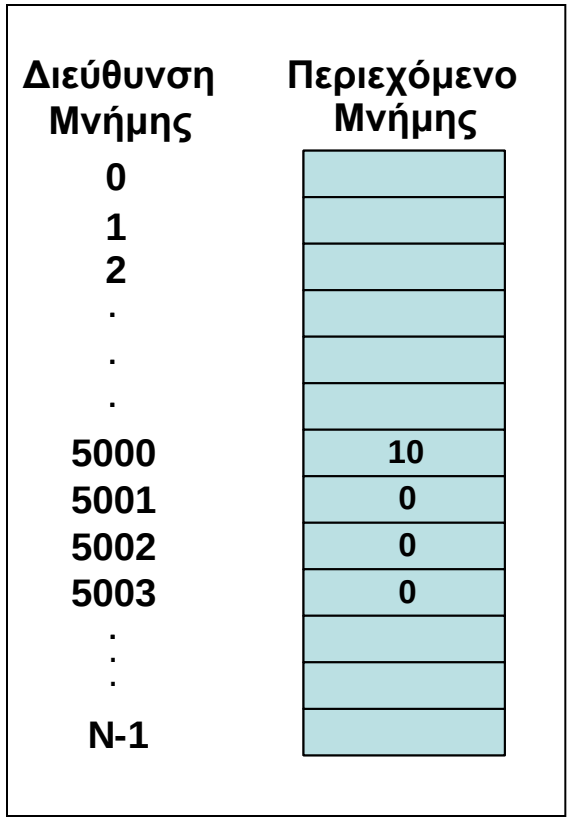
Στο διπλανό σχήμα θεωρούμε ότι η διεύθυνση της μεταβλητής a αρχίζει στη θέση `5000`

Η τιμή της a θα αποθηκευτεί στις θέσεις μνήμης από `5000` έως και `5003`

Ο μεταγλωττιστής συσχετίζει το όνομα της μεταβλητής a, με τη διεύθυνση της μεταβλητής

Όταν το πρόγραμμα χρησιμοποιεί το όνομα της μεταβλητής, ο μεταγλωττιστής προσπελαύνει αυτομάτως τη διεύθυνση της μεταβλητής

Π.χ. με την εντολή `a = 10;` ο μεταγλωττιστής γνωρίζει ότι η διεύθυνση της a είναι η `5000` και θέτει το περιεχόμενό της ίσο με `10`, ξεκινώντας από την οκτάδα με την χαμηλότερη διεύθυνση



Δείκτης στη C

Αρχικοποίηση

<τύπος> *<όνομα-δείκτη> = <διεύθυνση>;

π.χ.
```
int num;
int *num_ptr = &num;
```

Επιτρεπόμενοι Τελεστές

= ανάθεση τιμής σε δείκτη

***** περιεχόμενο διεύθυνσης που δείχνει ο δείκτης

& διεύθυνση μεταβλητής

Ο τελεστής & εφαρμόζεται **μόνο** σε αντικείμενα που είναι στη μνήμη δηλαδή σε μεταβλητές και στοιχεία πινάκων. Δεν μπορεί να εφαρμοστεί σε παραστάσεις, σταθερές.

Παράδειγμα

```
int x=1;
int y=2;
int *p;
p=&x; /*  η p δείχνει τη διεύθυνση της x      */
y=*p; /*  η y γίνεται 1, δηλαδή η τιμή της x   */
*p=0; /*  η x γίνεται 0  */
*p=*p+10; /*  x=x+10 */
```

Πέρασμα Μεταβλητών σε Συναρτήσεις (By Value)

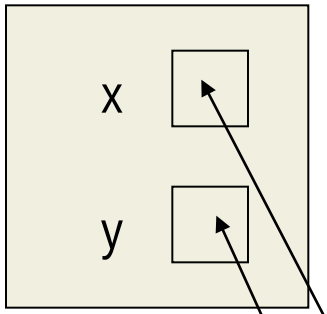
```
#include <stdio.h>
void swap( int, int );

int main()
{
     int x=2, y=3;
     printf("x:%d, y:%d \n", x,y);
     swap(x,y);
     printf("x:%d, y:%d \n", x,y);
}

void swap( int x, int y )  /* λάθος */
{
     int temp;
     temp=x;
     x=y;
     y=temp;
}
```

Πέρασμα Μεταβλητών σε Συναρτήσεις (By Reference)

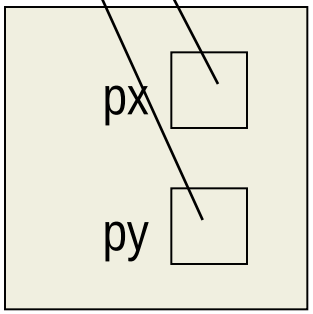
```
#include <stdio.h>
void swap(int*, int*);

int main()
{
     int x=2, y=3;
     printf("x:%d, y:%d \n", x,y);
     swap(&x,&y);
     printf("x:%d, y:%d \n", x,y);
}

void swap(int *px, int *py)  /*σωστή υλοποίηση */
{
     int temp;
     temp=*px;
     *px=*py;
     *py=temp;
}
```

Πέρασμα Μεταβλητών

```
int main()
{
}
```




```
void swap(int *px, int *py)
{
}
```



Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

functions

Άσκηση

Πέρασμα μεταβλητών - By Value

Δημιουργήστε ένα πρόγραμμα που θα υπολογίζει το τετράγωνο μίας τιμής με τη βοήθεια μίας συνάρτησης (sqr(…))

Η συνάρτηση θα επιστρέφει το τετράγωνο της τιμής που δέχεται σαν όρισμα

Άσκηση

Πέρασμα μεταβλητών - By Reference

Τροποποιήστε το προηγούμενο πρόγραμμα ώστε

η συνάρτηση να μην επιστρέφει τιμή

το όρισμα της συνάρτησης να είναι δείκτης σε float, και να χρησιμοποιήσετε το ίδιο το όρισμα της συνάρτησης για να επιστρέψετε την τιμή