ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ-2

STRUCTS

STRUCTS-ΔΟΜΕΣ

- STRUCT→Ένας τύπος δεδομένων που καθορίζεται από τον χρήστη και επιτρέπει των συνδυασμό διαφορετικών πρωτογενών τύπων.
- Πρωτογενείς τύποι→int,double,float,char
- Πρακτικά τα structs χρησιμέυουν ώστε να μπορώ να κατασκευάσω παραπάνω τύπυς δεδομένων που περιέχουν περισσότερα του ενός στοιχεία.
- ΠΑΡΑΔΕΙΓΜΑ:

```
typedef struct
{
    char *title;
    char *author_name;
    double price;
}Book;
```

- typedef→Με την χρήση της δεσμευμένης λέξης typedef ορίζω το όνομα του τύπου της δομής(Το όνομα που έχω δώσει πριν το ερωτηματικό)!!![Χωρίς κάθε φορά να χρησιμοποιώ την λέξη struct].
- ΠΑΡΑΔΕΙΓΜΑ:

```
typedef struct
{//πραγματοποιώ ορισμό του struct.
    //Δηλαδή για δηλώσω μία μεταβλητή της δομής
    //μπορώ να το κάνω τοποθετώντας σαν τύπο
    int id;
    char *name;
    double grade;
}student;
struct student
{
    //Για να χρησιμοποιήσω ένα τύπο της δομής
    //πρέπει να έχω σαν όνομα μεταβλητής struct student
    int id;
    char *name;
    double grade;
};
```

- Ταξινόμιση στοιχείων μίας δομής.

```
typedef struct
{
    char *title;
    char *author_name;
    double price;
}Book;
```

!!!Μία μεταβλητή τύπου Book δεν μπορεί να συγκριθεί με μία άλλη μεταβλητη τύπου Book κάθώς μία μεταβλητή τύπου Book περιέχει παραπάνω από ένα στοιχείο.Γενικά η τελεστές (είτε αριθμητικοί είτε σύγκρισης δεν λειτουργούν σε μεταβλητές που η αναπαράσταση τους πραγματοποιήται με struct).Άρα πρέπει να συγκρίνω με βάση ένα από τα πεδία του struct!!!

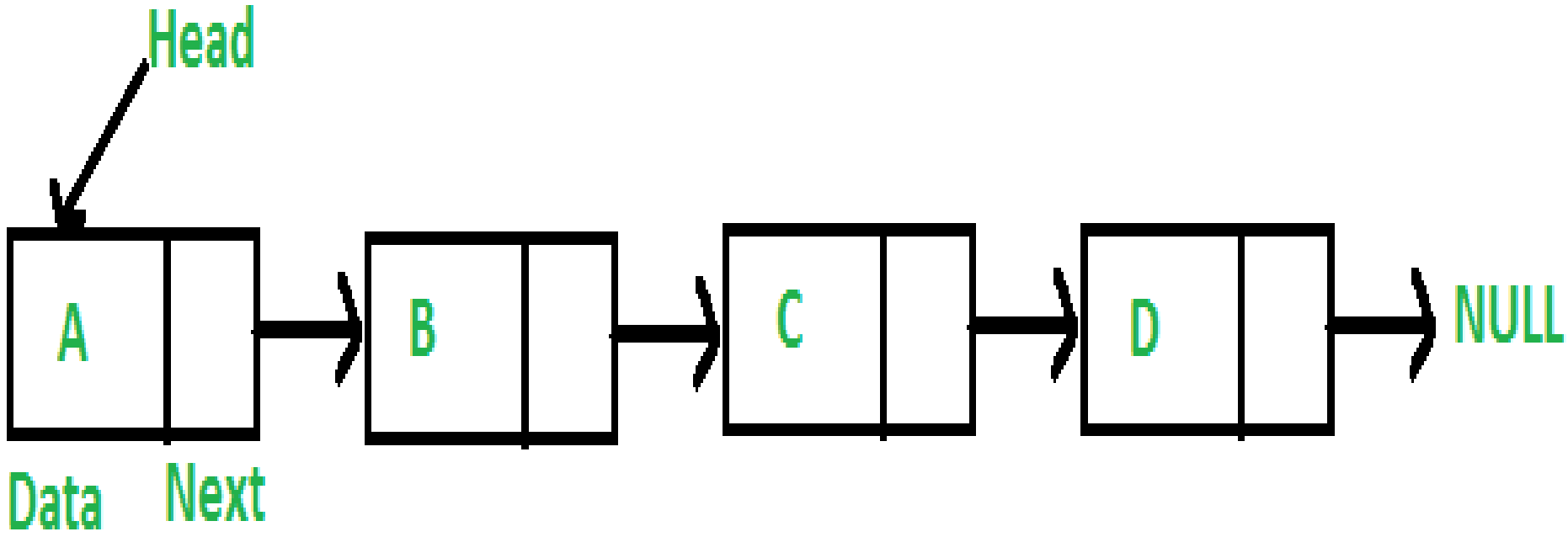
Π.χ αν η ταξινόμιση γινόταν με βάση την τιμή θα σύγκρινα την τιμή για κάθε βιβλίο(Book b1,B2; b1.price<b2.price).[github link]

ΠΑΡΑΔΕΙΓΜΑ_1

```c
#include <stdio.h>
#include <stdlib.h>
//Ορισμός δομής που θα αναπαριστά μαθητές
//Μία δομή χρησιμοποιήται ώστε να πραγματοποιήσουμε αναπαράσταση
//Ενός τύπου(π.χ students) που ορίζεται με παραπάνω από μία
//πρωτογενείς μεταβλητές.
typedef struct  {
    char name[100];
    double grade;
}student;
void readstruct(student *pin,int size)
{
    for(int i=0;i<size;i++)
    {
        fflush(stdin);
        printf("Give student name:");
        gets(pin[i].name);
        printf("Give students grade:");
        scanf("%lf",&pin[i].grade);
    }
    //Κάθε μεταβλητή student αναπαρείσταται και από 2 πεδία
    //name,grade άρα για κάθε struct διαβάζω 2 μεταβλητές.
}
```

```c
void printstudents(student *s,int size)
{
    //Εκτύπωση μαθητών
    for(int i=0;i<size;i++)
    {
        printf("Name:%s \t Grade:%.3lf\n",s[i].name,s[i].grade);
    }
}
int main()
{
    student *a;
    a=(student *)malloc(5 * sizeof(student));//Δυναμική δέσμευση πίνακα
    readstruct(a,5);//κλήση συνάρτησης
    printstudents(a,5);//κλήση συνάρτησης
    free(a);//Αποδέσμευση της μνήμης που δέσμευσα για τον πίνακα.
}
```

ΛΙΣΤΕΣ



- ΔΥΝΑΜΙΚΗ ΣΥΝΔΕΣΗ ΚΟΜΒΩΝ ΜΕΤΑΞΥ ΤΟΥΣ ΑΡΑ ΧΡΗΣΗ ΔΕΙΚΤΩΝ
- !!!Εφόσον δεν υπάρχει τύπος που να αναπαριστά τον κόμβο θα χρησιμοποιήσω struct για να ορίσω τον τύπο.
- Για μία απλά συνδεδεμένη λίστα(δείχνει μόνο στο επόμενο στοιχείο) η δομή θα έχει την εξής μορφή.

```
struct node
{
    int data;
    struct node *next;
};
```

1.ΔΗΛΩΣΗ ΚΟΜΒΩΝ ΚΑΙ ΕΥΡΕΣΗ ΜΗΚΟΥΣ ΛΙΣΤΑΣ

```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
    char data;
    struct node *next;
};
int length(struct node *curr)
{
    if(curr==NULL)
    {
        perror("empty list");
        return -1;
    }
  int l=0;
  while(curr!=NULL)
  {
      l++;
      curr=curr->next;
  }
  return l;
}
```

2.Τοποθέτηση στοιχείου στο τέλος της λίστας.

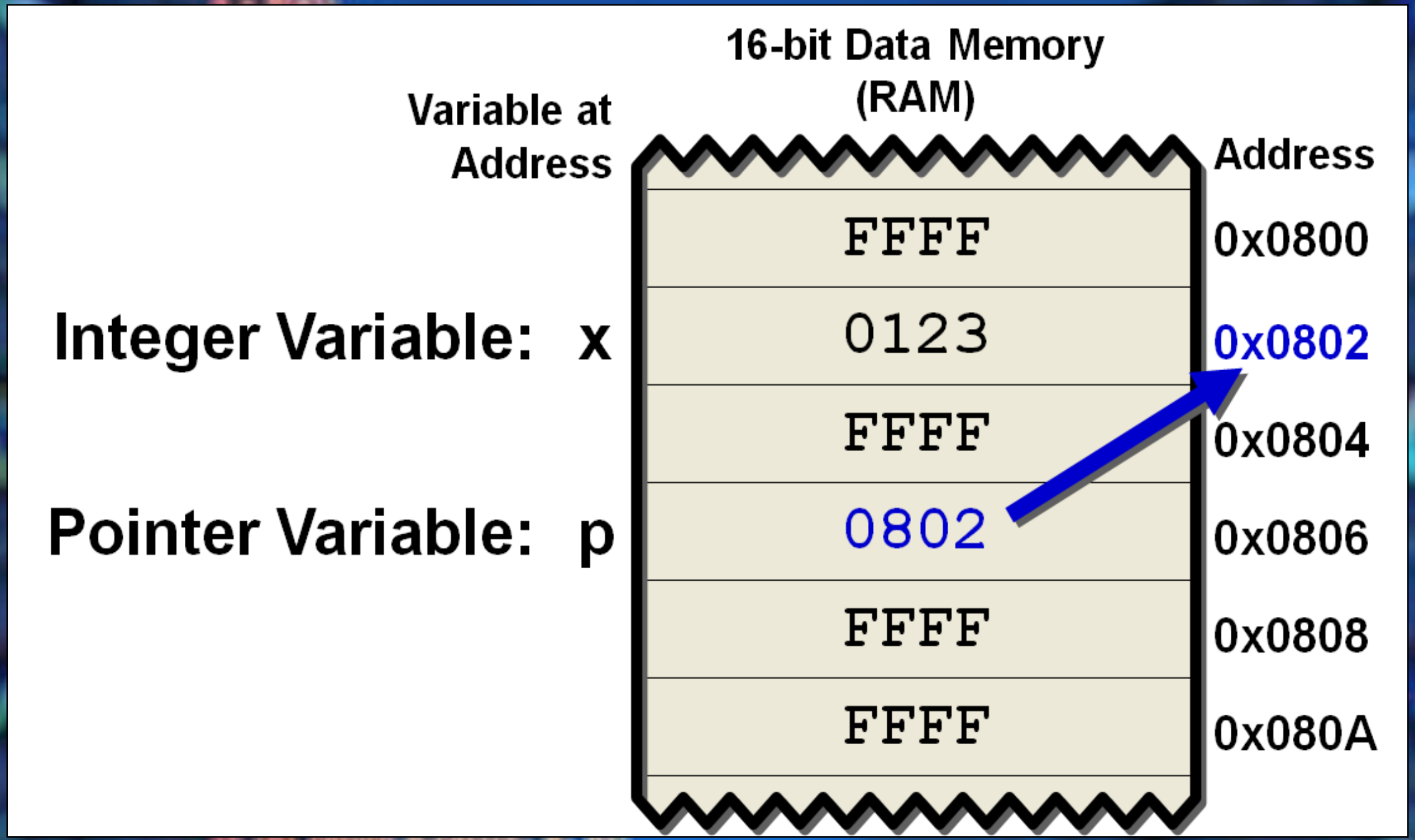
```
void pushback(struct node *curr,char x)
{
    struct node *newnode=(struct node *)malloc(sizeof(struct node)
    newnode->data=x;
    int size=length(curr);
    int i=1;
    while(i<size)
    {
        curr=curr->next;
        i++;
    }
    curr->next=newnode;
    newnode->next=NULL;
}
```

3.ΕΚΤΥΠΩΣΗ ΛΙΣΤΑΣ.

```
void print(list *l)
{
    struct node *curr=l->head;
    while(curr!=NULL)
    {
        printf("%c-",curr->data);
        curr=curr->next;
    }
    printf("\n");
}
```

!!!!!ΕΙΣΑΓΩΓΗ ΣΤΟΙΧΕΙΟΥ ΣΤΗΝ ΑΡΧΗ ΤΗΣ ΛΙΣΤΑΣ

```
struct node
{
    //Αναπαράσταση λίστας ακεραίων με struct.
    int data;
    struct node *next;
}
//**headref-->διπλό *,γιατί θέλω να πειράξω την θέση μνήμης που είναι αποθηκευμένο το
//πρώτο στοιχείο της λίστας θέλω να αλλάξω την διεύθυνση μνήμης του πρώτου κόμβου
//της λίστας,δηλαδή θέλω <<αναφορά>> στην διεύθυνση μνήμης του αρχικού κόμβου,ώστε να την αλλάξω.Ο μονός δείκτης
//θα έδειχνε μόνο στην διεύθυνση μνήμης,άρα θα μπορούσα να τροποποιήσω το περιεχόμενο εκείνης της διεύθυνσης.Με
//διπλό δείκτη έχω προχωρήσει ένα επίπεδο παρακάτω.Αλλάζω το περιεχόμενο της μεταβλητής head.
//όχι όμως και να αλλάξω την διεύθυνση.[Βασική διαφορά αναφοράς και δείκτη].
//Ο Δείκτης μπορεί να επαναοριστεί η αναφορά όχι.
//Επίσης στους δείκτες μπορώ να πραγματοποιήσω αριθμητική δεικτών,δηλαδη να προχωρήσω σε επόμενες θέσεις
//μνήμης εφόσον ένας δείκτης δείχνει σε παραπάνω από μία θέσεις μνήμης,στις αναφορές όχι.
void push_front(struct node **head_reference,int d)
{
    struct node *ref=*head_reference;
    //Δείκτης που θα δείχνει στην προηγούμενη αναφορά
    //που υπήρχε για το πρώτο στοιχείο στην λίστα
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    (*newnode).data=d;//εισαγωγή στην μνήμη νέου στοιχείου
    newnode->next=(*head_reference);
    (*head_reference)=newnode;
}
```


16-bit Data Memory
(RAM)
Variable at
Address
Address
Integer Variable: x
Pointer Variable: p
FFFF
0123
FFFF
0802
FFFF
FFFF
0x0800
0x0802
0x0804
0x0806
0x0808
0x080A

ΑΣΚΗΣΕΙΣ

1.Να κατεσκευαστεί δομή η οποία θα αποθηκεύει πληροφορίες για ένα φορτίο.Συγκεκριμένα ένα φορτίο περιέχει:

- Id→Αναγνωριστικός αριθμός ο οποίος είναι ακέραιος
- weight→βάρος φορτίου(σε κιλά).
- destination→προορισμός φορτίου.
- price→Τιμή φορτίου

I. Να κατασκευαστεί συνάρτηση η οποία με βάση το id θα διαγράφει ένα φορτίο από τον πίνακα διαχείρησης φορτίων.

II. Να κατασκευαστεί συνάρτηση η οποία θα επιστρέφει το ακριβότερο φορτίο.

III. Να κατεσκευαστεί συνάρτηση η οποία θα αποθηκεύει τα φορτία σε ένα αρχείο.

IV. Στην κύρια συνάρτηση ο χρήστης θα ανοίγει ένα αρχείο και θα διαβάζει δεδομένα από το αρχείο και θα τα αποθηκεύει σε πίνακα,θα πραγμοτοποιεί τις διαγραφές των id.Επίσης για θα κάνει την ακόλουθη μετατροπή

→Για κάθε φορτίο,θα αυξάνει την τιμή κατά 0,1 ανά 200 γραμμάρια.

Τέλος να καλεί την συνάρτηση αποθήκευσης δεδομένων σε φορτίο.

→Χρησιμοποιήσται τον παρακάτω κώδικα:

https://github.com/vasnastos/PROGRAMMING-TO-C-2/blob/master/Course5/EX_1/cargostruct.c

2.Για μία απλά συνδεδεμένη λίστα να υλοποιηθούν οι παρακάτω συναρτήσεις:

- int length(struct node *curr)→η οποία βρίσκει το μήκος της λίστας μας
- int sum(struct node *curr)→η οποία βρίσκει το άθροισμα των στοιχείων της λίστας.
- void printlist(struct node *curr)→Εκτύπωση στοιχείων της λίστας.
- void push_back(struct node *curr,int d)→τοποθέτηση στοιχείου στο τέλος της λίστας.

→Χρησιμοποίηστε τον παρακάτω κώδικα:

https://github.com/vasnastos/PROGRAMMING-TO-C-2/blob/master/Course5/EX_2/list.c

3.Να κατασκευαστεί μία δομή λίστας για μαθητές που σαν δεδομένα θα έχει το id του μαθητή ,το όνομα μαθηματος και τον βαθμό του.

Να κατεσκευαστούν οι ακόλουθες συναρτήσεις:

- void push_back()→Η οποία θα εισάγει ένα μαθητή στο τέλος της λίστας.
- double average()→Η οποία θα επιστρέφει τον μέσο όρο της λίστας.
- int length()→Η οποία θα επιστρέφει το μήκος της λίστας.
- void delete_student(int id)→Η οποία με βάση το id ενός μαθητή θα τον διαγράφει από την λίστα[Θεωρείσται ότι το κάθε id υπάρχει μία φορά στην λίστα].

→Χρησιμοποίηστε τον παρακάτω κώδικα:

https://github.com/vasnastos/PROGRAMMING-TO-C-2/blob/master/Course5/EX_3/studentslist.c