

# Εργαστήριο

# Εισαγωγή στον Προγραμματισμό

Εργαστήριο 09 (10<sup>ο</sup> εργαστήριο)

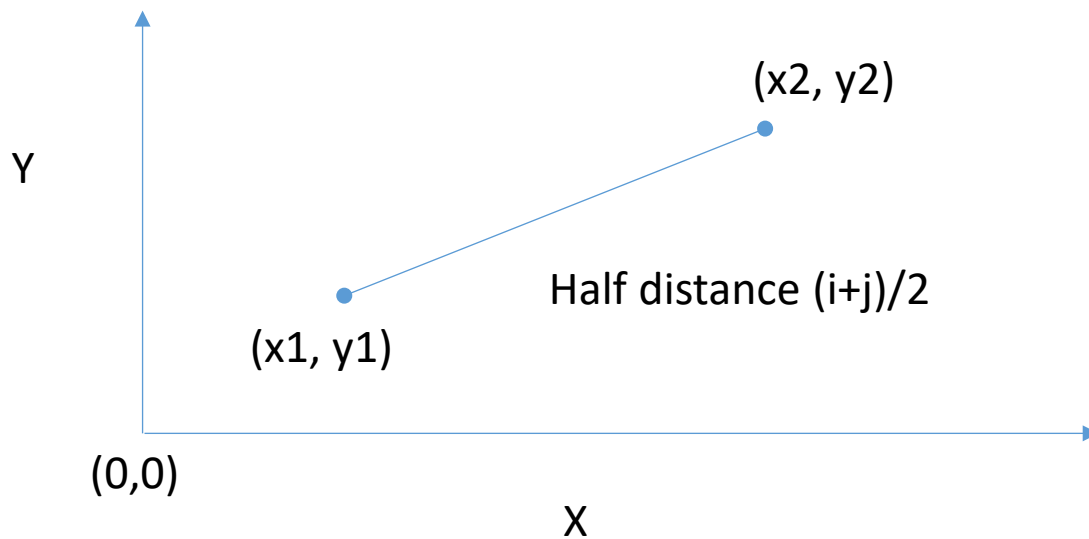
## Δομές (structures)

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών

# Εργαστήριο 09

## point.c (structure)



Ένα σημείο στο επίπεδο μπορεί να οριστεί ως ένα ζεύγος τιμών (x,y)

# Εργαστήριο 09

## point.c

//Create a structure that defines a point  
in a 2-dimensional space

```
struct point {  
  
double x;  
double y;  
  
}; //end struct point
```

```
struct point middle(struct point a, struct point b) {
```

```
struct point median;
```

```
median.x=(a.x+b.x)/2;  
median.y=(a.y+b.y)/2;
```

```
return median;
```

```
} //end function middle
```

# Εργαστήριο 09

## point.c

```
int main() {  
    struct point a,b, result;  
  
    a.x=1.2;a.y=5.4;  
    b.x=7.3;b.y=1.8;  
  
    //  struct point a = {1.2, 5.4};  
    //  struct point b = {7.3, 1.8};  
  
    result=middle(a,b);  
    printf("The middle of the line has the following coordinates:  
x=%f , y=%f\n", result.x, result.y);  
  
    return 0;  
}
```

# Εργαστήριο 09

## person.c (structure + pointer)

```
struct person *person_init(char *firstname, char *lastname, char *middlename){
```

```
struct person *newperson; ← Δείκτης σε structure
```

```
newperson=malloc(sizeof(struct person));
```

```
newperson->fname=firstname;
```

```
newperson->sname=lastname;
```

```
newperson->mname=middlename;
```

```
return newperson;
```

```
}
```

# Εργαστήριο 09

## person.c

```
struct person *childof(struct person father, char *newname)
{

    struct person *newperson;

    newperson=malloc(sizeof(struct person));
    newperson->fname=newname;
    newperson->sname=father.sname;
    newperson->mname=father.fname;

    return newperson;

}
```

# Εργαστήριο 09

## person.c

```
int main() {
    struct person *father, *child;

    father=person_init("Dimitrios", "Vassilopoulos", "Dionysios");
    child=childof(*father, "Dionysios");

    printf("Father name: %s, Father surname: %s, Father middlename: %s\n", (*father).fname, (*father).sname, (*father).mname);
    printf("Child's name: %s, Child's surname: %s, Child's middlename: %s\n", child->fname, child->sname, child->mname);

    free(father);                //Deallocate memory
    free(child);

    return 0;

}
```

# Εργαστήριο 09

## grades.c (συνδεδεμένες λίστες – linked lists)

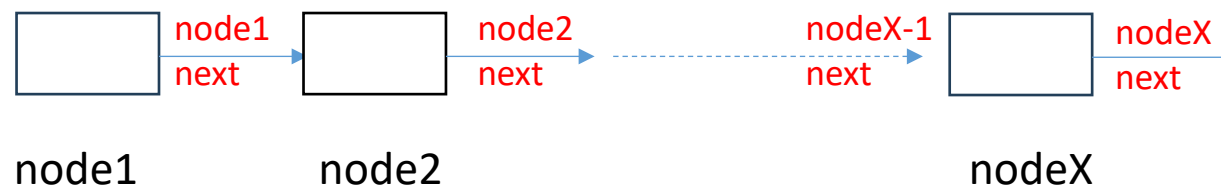
```
typedef struct listnode *Listptr;           //ΟΡΙΖΩ ΝΕΟ ΤΥΠΟ ΔΕΔΟΜΕΝΩΝ  
                                           //Είναι pointer στη δομή (struct) listnode
```

πλέον μπορώ να ορίσω ότι μία μεταβλητή είναι τύπου Listptr (όπως π.χ. θα δήλωνα ότι μία μεταβλητή είναι τύπου int)

```
Listptr mylist=NULL;
```

```
struct listnode {  
    int data;  
    Listptr next;  
};
```

Δομή που περιέχει και ένα τύπο δεδομένων τύπου pointer (Listptr) που «δείχνει» σε διεύθυνση που έχει τον ίδιο τύπο δεδομένων

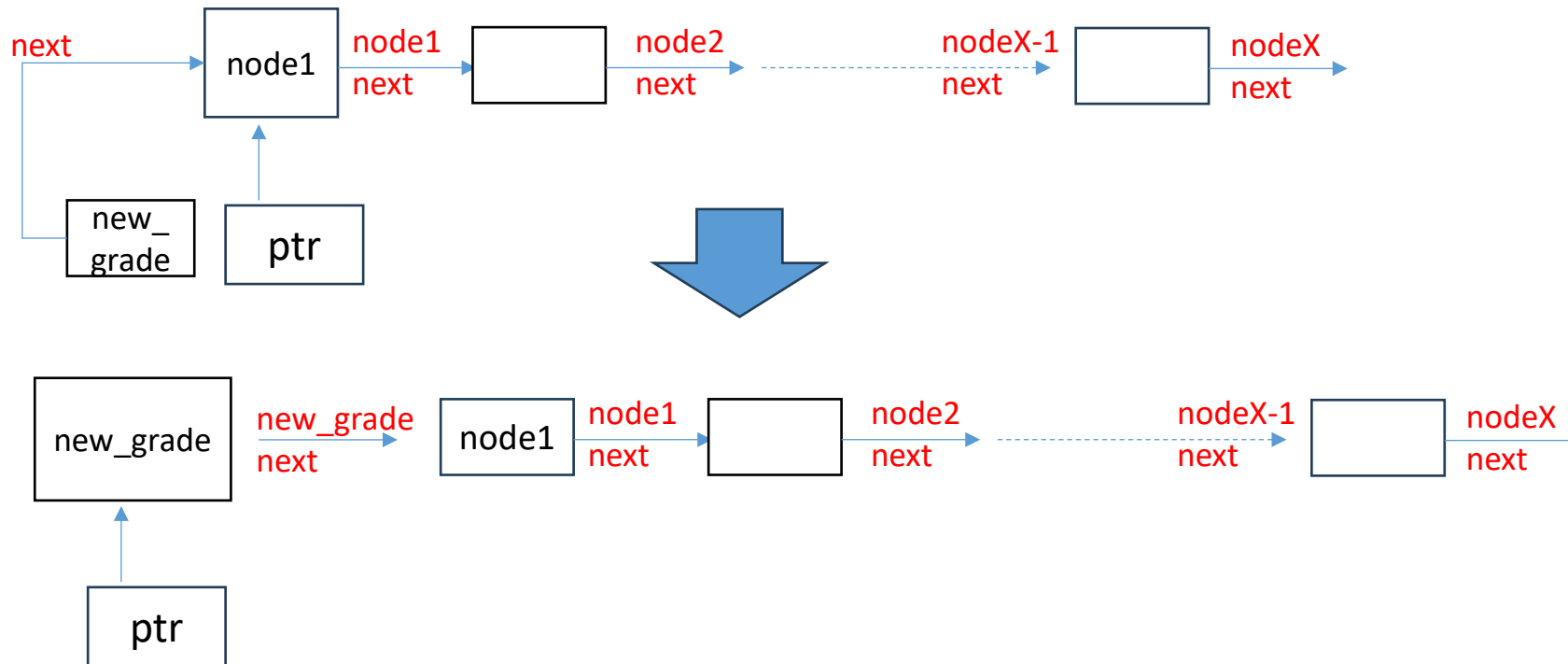




# Εργαστήριο 09

## grades.c (συνδεδεμένες λίστες – linked lists)

```
void insert_at_start(Listptr *ptr, int grade)
```



**ΠΡΕΠΕΙ ΠΑΝΤΑ ΝΑ ΓΝΩΡΙΖΕΤΕ  
ΤΗΝ ΑΡΧΗ ΤΗΣ ΛΙΣΤΑΣ**

# Εργαστήριο 09

## grades.c (συνδεδεμένες λίστες – linked lists)

```
void print_list(Listptr ptr){
```

```
    int i=0;  
    while (ptr!=NULL) {
```

```
        printf("Grade %d: %d\n",++i, ptr->data);  
        ptr=ptr->next;
```

```
    }  
}
```

```
void free_mem(Listptr ptr){
```

```
    while (ptr!=NULL) {  
        Listptr ptr1;
```

```
        ptr1=ptr->next;  
        free(ptr);  
        ptr=ptr1;
```

```
    }  
}
```

# Εργαστήριο 09

## tree.c (δυναμικά δένδρα)

```
typedef struct tnode *Treeptr;
```

```
struct tnode {  
    int data;  
    Treeptr left;  
    Treeptr right;  
};
```

