Προγραμματισμός

Δομές Δεδομένων

Δομές Δεδομένων (Data Structures)

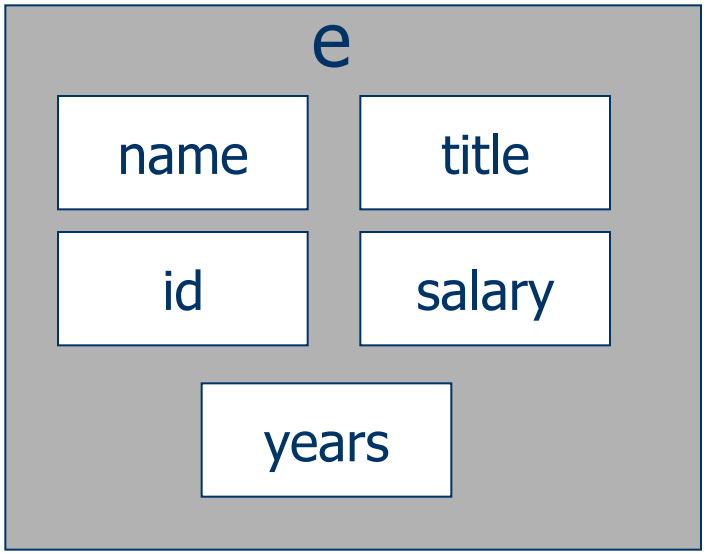
- Καινούργιοι τύποι δεδομένων που αποτελούνται από την ομαδοποίηση υπαρχόντων τύπων δεδομένων
- Ομαδοποίηση πληροφορίας που δεν μπορεί να αποθηκευτεί σε μία μόνον μεταβλητή από τους δοσμένους τύπους της C
- Συλλογή μη διατεταγμένων και ομοιογενών τιμών
- Πλεονέκτημα: Καλύτερη οργάνωση – διαχείριση πληροφορίας πιο ευέλικτος σχεδιασμός

Όνομα	Τίτλος	#Ταυτότητας	Μισθός	Χρόνια
Κώστας	διευθυντής	A322442	1500	4
ΠΛ	Υπάλληλος	B323144	1000	5
ΑΛ	Υπάλληλος	B321421	800	2
ΒΛ	Υπάλληλος	B134325	750	1

Ορισμός Δομών

- Ορισμός νέου τύπου δομής

```
struct employee
{
        char *name;
        char title[20];
        char id[8];
        double salary;
        int years;
};
```



- Τα name, title, id, salary, years ονομάζονται τα μέλη ή πεδία της δομής
- Το employee ονομάζεται η ετικέτα της δομής

Αναπαράσταση στη μνήμη

struct employee

Δήλωση Μεταβλητών Τύπου Δομής

```
struct employee e1, e2;

struct employee *pe;

struct employee manager = {"AB", "manager", "A11111", 2000, 2};

struct employee e3 = {0}; // μηδενίζει όλα τα στοιχεία
```

Πρόσβαση στα Περιεχόμενα της Δομής

```
struct point
{
    int x;
    int y;
} ;
struct point p;
```

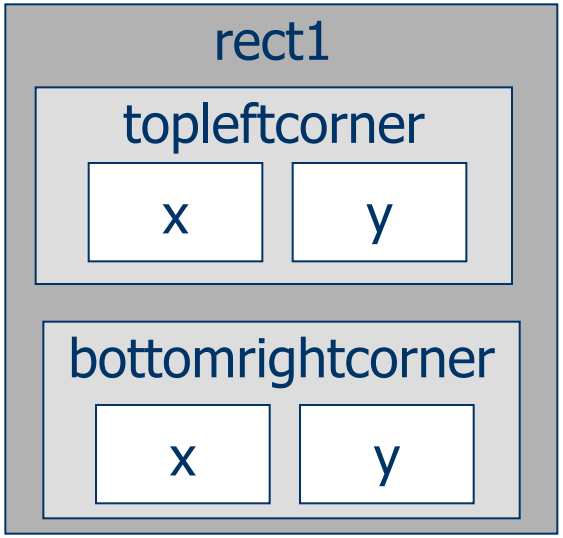
- Με τον τελεστή ‘.’ (τελεία) αποκτούμε πρόσβαση στα μέλη της δομής

```
p.x = 10;
p.y = 20;
```

Εμφωλιασμένες Δομές

- Δομές ως μέλη άλλων δομών είναι επιτρεπτά:

```
struct point
{
        int x;
        int y;
};
struct rectangle
{
    struct point topleftcorner;
    struct point bottomrightcorner;
} rect1;
```



- Πρόσβαση με διαδοχικές τελείες (.)

  rect1.topleftcorner.x = 10;

- Παίρνουμε το πεδίο x του πεδίου topleftcorner της μεταβλητής rect1

Πίνακες από Δομές

- Όπως και για κάθε άλλο τύπο:

```
struct point
{
    int x;
    int y;
} ;
struct point p;
struct point points[100];
points[2].x = 10;
p.x = 42;
```

Χρήση Δομών

- Σαν κανονικές μεταβλητές για την αποθήκευση πληροφορίας
- Σαν ορίσματα σε Συναρτήσεις:

```
void printEmployee( struct employee e);
```

- Σαν επιστρεφόμενες τιμές από συναρτήσεις:

```
struct employee newEmployee(char *name,
                            char *pos,
                            char *id,
                            double sal,
                            int y);
```

- Σαν επιστρεφόμενες τιμές από συναρτήσεις ως δείκτες:

```
struct employee *newEmployee(char *name,
                             char *pos,
                             char *id,
                             double sal, int y);
```

Σε ποια περίπτωση θα χρησιμοποιούσατε το κάθε ένα;

Δείκτες σε Δομές

```
struct employee
{
        char *name;
        char title[20];
        char id[8];
        double salary;
        int years;
}; // Δήλωση της δομής employee
struct employee *e;
// Δήλωση της μεταβλητής e ως δείκτης σε δομή.

struct employee d; // Δήλωση της μεταβλητής d ως δομή τύπου employee

e = &d; // το e παίρνει τιμή την διεύθυνση της δομής d
int *pi = &(d.years);

// το e είναι δείκτης σε πίνακα 100 στοιχείων struct employee
e = new employee[100]; // C++
e = (struct employee *) malloc(100*sizeof(struct employee));  /* C */
e[0].salary = 1000;
```

Δείκτες σε Δομές

- struct employee *e;
- Πρόσβαση στα περιεχόμενα του δείκτη
  - (*e) τα περιεχόμενα του δείκτη είναι η δομή
  - (*e).title το πεδίο title της δομής που δείχνει ο e
  - **e[0].title συνώνυμο του παραπάνω**
  - **e->title συνώνυμο του παραπάνω**
- Προσοχή στις προτεραιότητες των τελεστών:
  - (*e).title
  - *e.title είναι ισοδύναμο με *(e.title) που δεν είναι αποδεκτό για την συγκεκριμένη δήλωση του e

Αριθμητική δεικτών

- struct employee *e;
- e++;
- Προχωράει κατά sizeof(struct employee);

Επιτρεπτές Πράξεις σε Δομές

```
struct point
{
    int x;
    int y;
} p1, p2;
```

- Αντιγραφή p1 = p2;
- Απόδοση τιμής σαν σύνολο με λίστα σταθερών τιμών για τα μέλη: p1 = {5, 6};
- Εξαγωγή διεύθυνσης: &p1
- Προσπέλαση των μελών: p1.x

- ΔΕΝ ΕΠΙΤΡΕΠΕΤΑΙ η σύγκριση δομών

- fread(), fwrite()

Δομή Δεδομένων: Πίνακας

- Οργάνωση των δεδομένων:
  - το ένα μετά το άλλο στην μνήμη
- Τελεστές:
  - Δημιουργία πίνακα με συγκεκριμένο αριθμό στοιχείων
  - Αποδέσμευση πίνακα με συγκεκριμένο αριθμό στοιχείων
  - Εισαγωγή στοιχείου σε συγκεκριμένη θέση
  - Ανάκτηση ενός στοιχείου από δεδομένη θέση
- Αλγόριθμοι: τους περιέχει ο μεταγλωττιστής της C
  - int a[100]; ή a = (int *)malloc(100 * sizeof(int));
  - με το πέρας της εμβέλειας του a ή free(a);
  - a[34] = 7;
  - b = a[36];

Πίνακας

- Οργάνωση των δεδομένων: θεωρούμε τα δεδομένα μας διατεταγμένα (το ένα ακολουθεί το άλλο)

```
//Δημιουργία
struct employee *newEmployee(int number)
{
	struct employee *e;
	int i;
	e = new employee [number];
	for (i = 0; i < number; ++i)
		e[i].salary = 1000;

	return e;
}


//Διαγραφή…
//Αναζήτηση…
//Εκτύπωση…
```

Πρόγραμμα

- Δημιουργεί ένα δυναμικό πίνακα από τυχαία σημεία του χώρου.
- Υπολογίζει όλα τα δυνατά τρίγωνα που δημιουργούνται από τα σημεία αυτά και τα εκτυπώνει.
- Μετά διαγράφει τυχαία κάποια μέλη του πίνακα και τυπώνει ξανά τα τρίγωνα.
- Μετά αποδεσμεύει τη μνήμη του πίνακα.

typedef

- **Ορισμός νέων ονομάτων για τους τύπους των μεταβλητών**
- typedef τυπος όνομα;
  - Κάνει συνώνυμο το όνομα με τον τύπο
  - typedef unsigned long int size_t
    - Το size_t γίνεται συνώνυμο του unsigned long int
    - Αντί unsigned long int var1;
      - size_t var;

Παραδείγματα typedef

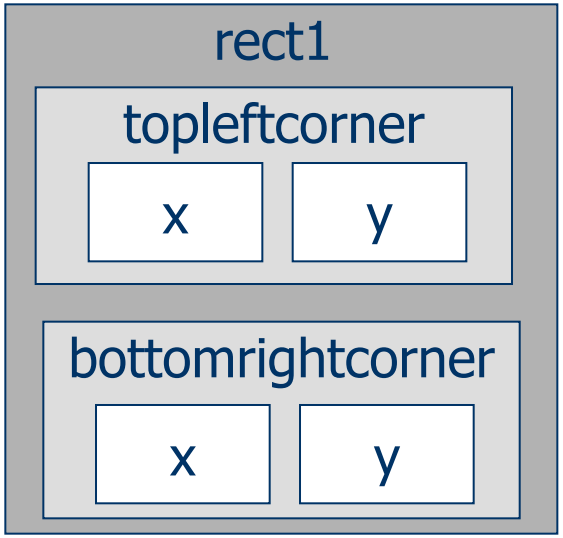
```
typedef struct employee employee;
employee e;

typedef struct employee
{
        char *name;
        char title[20];
} Employee;

typedef struct employee
{
        char *name;
        char title[20];
} employee;
```

Χωρίς την typedef μπροστά δηλώνει μια μεταβλητή τύπου struct employee. Με την typedef δηλώνει ότι το όνομα Employee είναι συνώνυμο του struct employee.

Εμφωλιασμένες Δομές

- Δομές ως μέλη άλλων δομών είναι επιτρεπτά:

```
struct point
{
        int x;
        int y;
};
struct rectangle
{
    struct point topleftcorner;
    struct point bottomrightcorner;
} rect1;
```



- Πρόσβαση με διαδοχικές τελείες (.)

  rect1.topleftcorner.x = 10;

- Παίρνουμε το πεδίο x του πεδίου topleftcorner της μεταβλητής rect1

```
struct point
{
      int x;
      int y;
} p1, p2;

    p1.x = 10;
    p2.y = 20;
```

- Εξαγωγή διεύθυνσης: &p1
- Προσπέλαση των μελών: p1.x

Unions

- Παρόμοια με τα structs

```
union time
   {
      long simpleDate;
      double perciseDate;
    } mytime;
```

- Κρατείται ο «μεγαλύτερος» τύπος

Unions: Example

```
int main()
    {
     union data
        {
          char a;
             int x;
             float f;
     } myData;

     int mode = 1;
     myData.a = 'A';
     printf("Here is the
        Data:\n%c\n%i\n%.3f\n", myData.a,
        myData.x, myData.f );

     myData.x = 42;
     mode = 2;
     printf("Here is the Data:
        \n%c\n%i\n%.3f\n", myData.a,
        myData.x, myData.f );

     myData.f = 101.357;
     mode = 3;
     printf("Here is the Data:
     \n%c\n%i\n%.3f\n", myData.a,
     myData.x, myData.f );

     if( mode == 1 )
       printf("The char is being used\n");
else if( mode == 2 )
       printf("The int is being used\n");
else if( mode == 3 )
       printf("The float is being used\n");
return 0;
}
```

ΗΥ-150
Προγραμματισμός

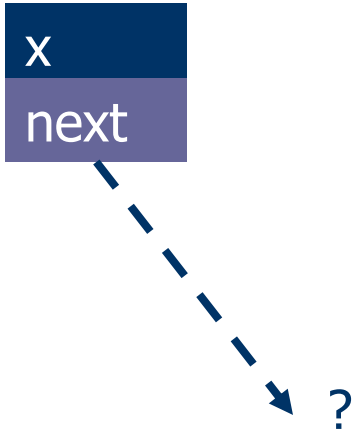
Αυτοαναφορικές Δομές
Δυναμικές Δομές Δεδομένων: Λίστες

Αυτοαναφορικές Δομές

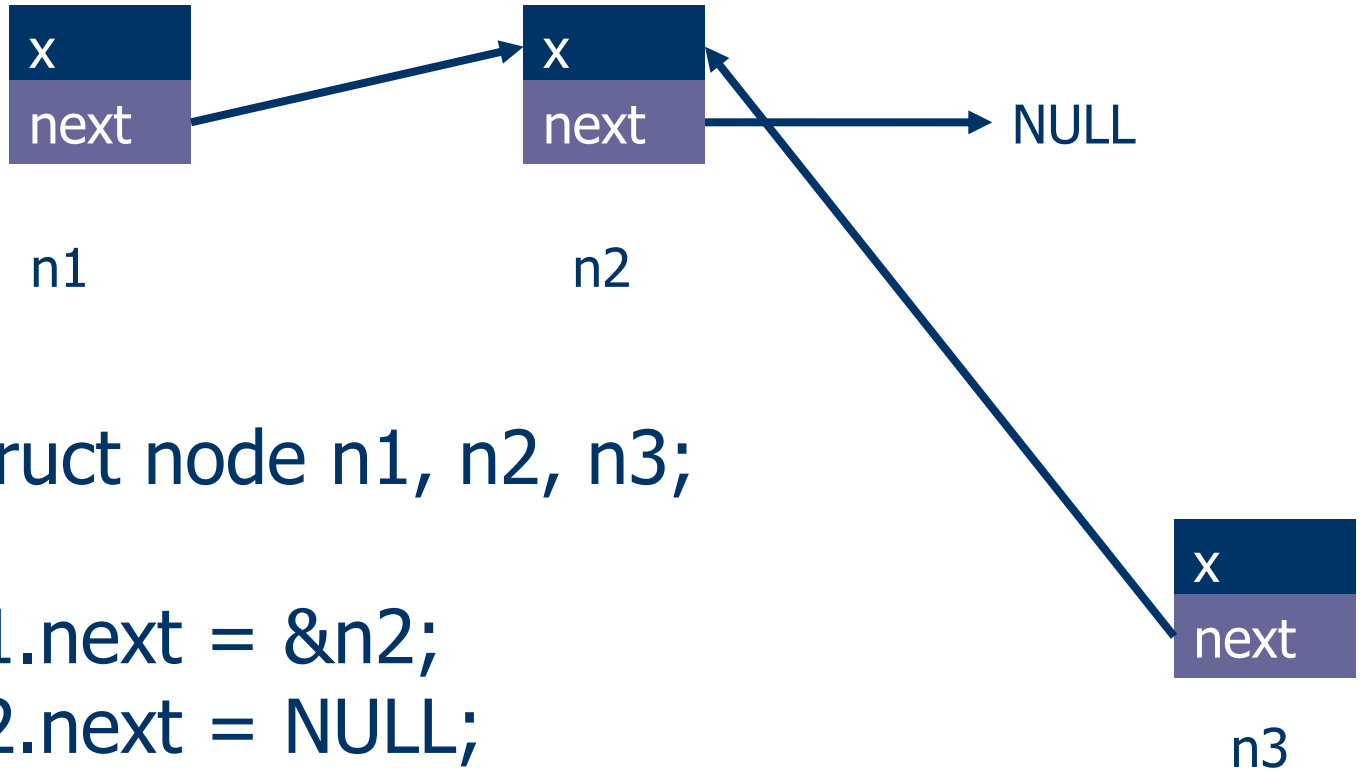
- Δομές που περιέχουν ως μέλη δείκτες σε δομές ίδιου τύπου

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

Αυτοαναφορικές Δομές



Αυτοαναφορικές Δομές



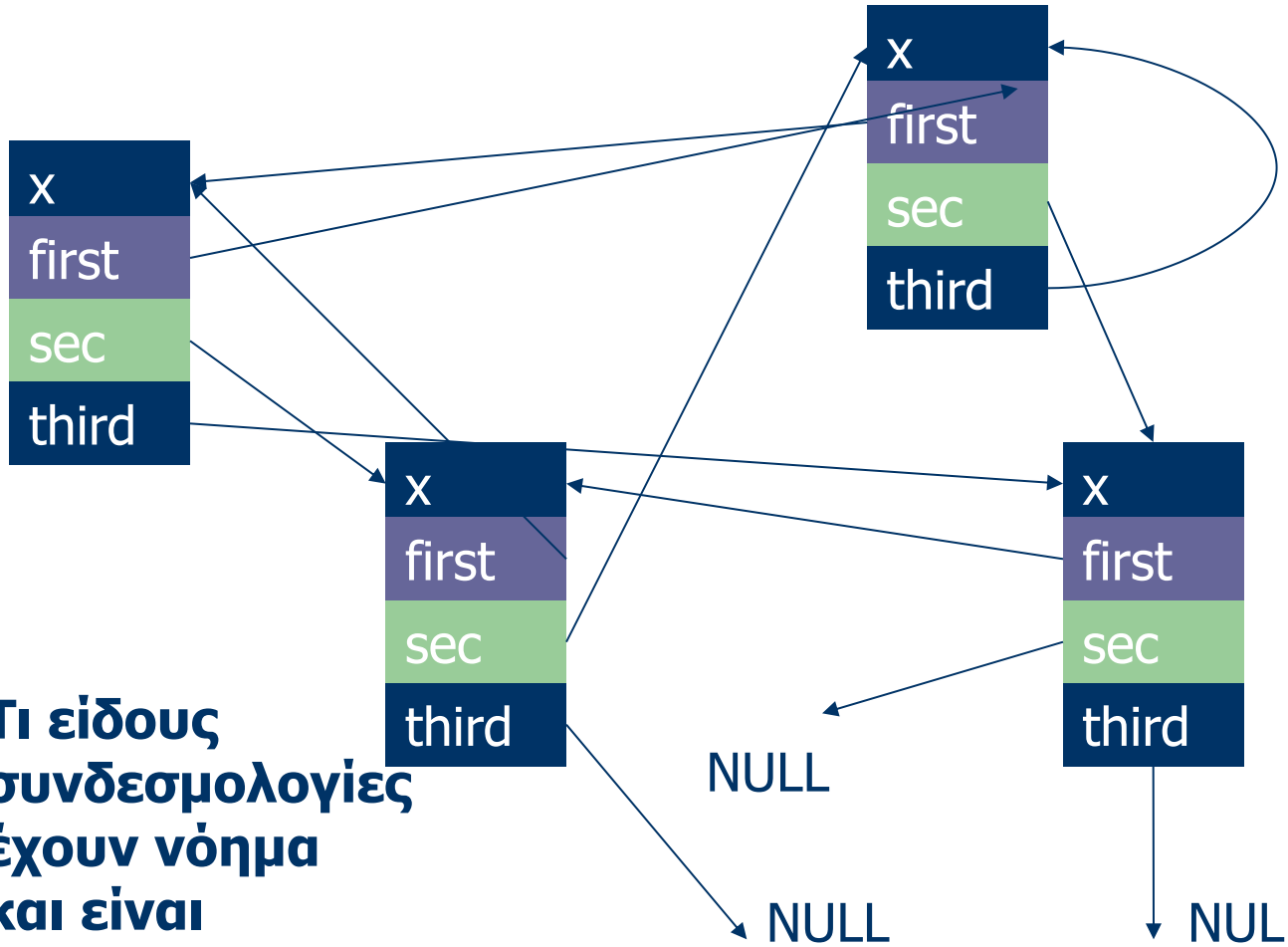

```
struct node n1, n2, n3;

n1.next = &n2;
n2.next = NULL;
n3.next = &n2;
```

Αυτοαναφορικές Δομές

- Με περισσότερους από ένα δείκτες μπορούμε να έχουμε πολύ πολύπλοκες δομές δεδομένων

```
struct node
{
   int x;
   struct node *first;
   struct node *sec;
   struct node *third;
};
```

Αυτοαναφορικές Δομές



Τα NULL δηλώνουν ότι δεν δείχνει πουθενά ο δείκτης

Τι είδους συνδεσμολογίες έχουν νόημα και είναι χρήσιμες;

Δείκτες, Πίνακες και Δομές

- Δήλωση του τύπου
  - struct somestruct *ptr;
- Τι είναι και πως έχουμε πρόσβαση;

- Περίπτωση 1:
  - **μια μοναδική μεταβλητή τύπου struct somestruct**
  - Πρόσβαση: ptr->member
  - Δέσμευση μνήμης
    - Στατικά:
      - struct somestruct a;
      - ptr = &a;
    - Δυναμικά:
      - ptr = new somestruct();
      - ptr = (struct somestruct *)malloc(sizeof(struct somestruct));

Δείκτες, Πίνακες και Δομές

- Δήλωση του τύπου
  - struct somestruct *ptr;
- Περίπτωση 2:
  - **ένας πίνακας από μεταβλητές τύπου struct somestruct**
  - Πρόσβαση: ptr[index].member
  - Δέσμευση μνήμης
    - Στατικά:
      - struct somestruct a[100];
      - ptr = a; ή ptr = &a[0];
    - Δυναμικά:
      - ptr = new somestruct[100];
      - ptr = (struct somestruct *)malloc(100*sizeof(struct somestruct));

Δείκτες, Πίνακες και Δομές

- Δήλωση του τύπου
  - struct somestruct *ptr;
- Περίπτωση 3:
  - **μια δυναμική δομή δεδομένων με στοιχεία μεταβλητές τύπου struct somestruct**
  - Πρόσβαση: εξαρτάται από την δομή δεδομένων
  - Δέσμευση μνήμης: εξαρτάται από την δομή δεδομένων
- Ο προγραμματιστής πρέπει να ξέρει την σημασιολογία της ptr και να την χρησιμοποιεί ανάλογα!

Δομές Δεδομένων

- Ορισμός:
  - Μια συλλογή δεδομένων οργανωμένα με συγκεκριμένο τρόπο + ένα σύνολο τελεστών που επενεργούν πάνω στην συλλογή + αλγόριθμοι που υλοποιούν τους τελεστές

- Δέντρα (δυαδικά και μη, B, R, Red-Black, κτλ)
- Γράφοι (διαταγμένοι, με βάρη, απλοί ή όχι, υπεργράφοι κτλ)
- Λίστες, ουρές, στοίβες, πίνακες, πίνακες κατακερματισμού (hash tables)

Απλά Συνδεδεμένη Λίστα

- Οργάνωση των δεδομένων:
  - θεωρούμε τα δεδομένα μας διατεταγμένα (το ένα ακολουθεί το άλλο)
  - το καθένα δείχνει στο επόμενο του στην μνήμη

```
struct node
{
    int x;
    struct node *next;
};
struct node *root;
```

Απλά Συνδεδεμένη Λίστα

- Τελεστές:
  - Δημιουργία κενής λίστας
  - Αποδέσμευση λίστας
  - Εισαγωγή στοιχείου μετά από ένα στοιχείο
  - Διάσχιση της λίστας
  - Ανάκτηση ενός στοιχείου με συγκεκριμένο περιεχόμενο

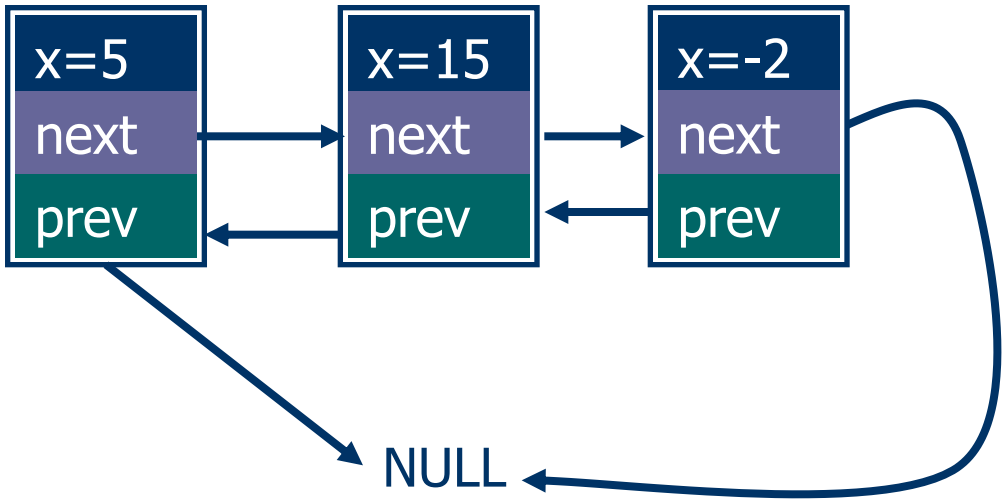
Απλά Συνδεδεμένη Λίστα

- Αλγόριθμοι:
  - Εισαγωγής
  - Διαγραφής
  - Αναζήτησης
  - Εκτύπωση
  - …

Διπλά Συνδεδεμένη Λίστα

- Οργάνωση των δεδομένων:
  - θεωρούμε τα δεδομένα μας διατεταγμένα (το ένα ακολουθεί το άλλο)
  - το καθένα δείχνει στο επόμενο και στο προηγούμενο του στην μνήμη

```
struct node
{
    int x;
    struct node *next;
    struct node *prev;
};
```



Πρόγραμμα

- Δημιουργεί μια απλά συνδεδεμένη λίστα από τυχαίους ακεραίους, τους εκτυπώνει. Μετά διαγράφει τυχαία κάποια μέλη της λίστας και την ξανατυπώνει. Μετά αποδεσμεύει τη μνήμη της λίστας.