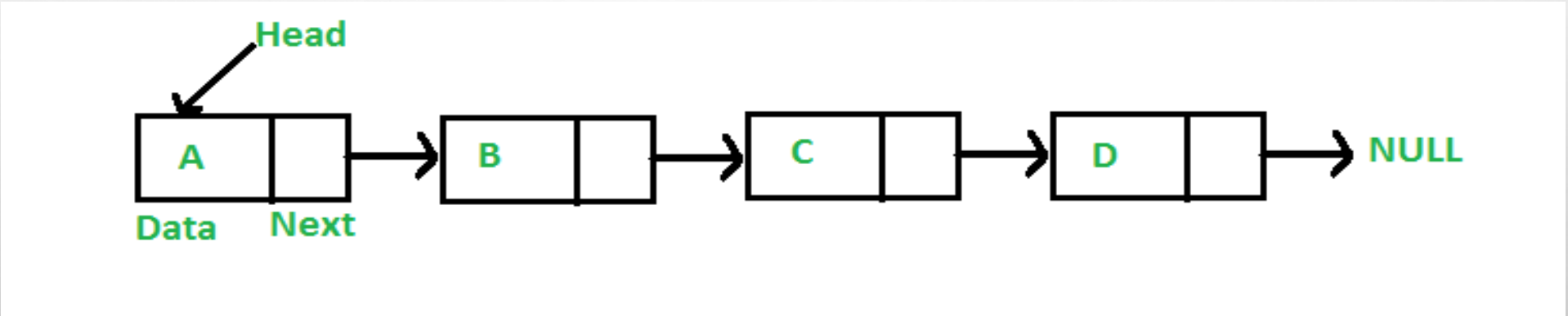
ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

- ΜΙΑ ΛΙΣΤΑ ΕΊΝΑΙ ΕΝΑ ΜΙΑ ΔΟΜΗ ΠΟΥ ΠΕΡΙΧΕΙ ΣΤΟΙΧΕΙΑ ΤΑ ΟΠΟΙΑ ΠΑΡΕΧΟΥΝ ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΚΑΙ ΓΙΑ ΤΟ ΕΠΟΜΕΝΟ ΤΟΥΣ ΣΤΟΙΧΕΙΟ.ΤΑ ΣΤΟΙΧΕΙΑ ΤΗΣ ΛΙΣΤΑΣ ΔΕΝ ΑΠΟΘΗΚΕΥΟΝΤΑΙ ΣΕ ΣΥΝΕΧΟΜΕΝΕΣ ΘΕΣΕΙΣ ΜΝΗΜΗΣ.ΚΑΘΕ ΣΤΟΙΧΕΙΟ ΣΥΝΔΕΕΤΑΙ ΜΕ ΤΟ ΕΠΟΜΕΝΟ (Η ΚΑΙ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΜΕ ΤΗΝ ΧΡΗΣΗ ΔΕΙΚΤΗ).ΣΤΙΣ ΛΙΣΤΕΣ ΟΙ ΚΟΜΒΟΙ ΠΟΥ ΕΙΣΑΓΩΝΤΑΙ ΕΙΝΑΙ ΔΕΙΚΤΕΣ ΓΙΑΤΙ Η ΛΙΣΤΑ ΕΠΗΡΕΑΖΕΙ ΤΗΝ ΜΝΗΜΗ.



- ΑΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ→ΚΑΘΕ ΚΟΜΒΟΣ ΕΧΕΙ ΚΑΙ ΕΝΑ ΔΕΙΚΤΗ ΠΟΥ ΔΕΙΧΝΕΙ ΣΤΗΝ ΘΕΣΗ
- ΜΝΗΜΗΣ ΠΟΥ ΕΙΝΑΙ ΑΠΟΘΗΚΕΥΜΕΝΟ ΤΟ ΕΠΟΜΕΝΟ ΣΤΟΙΧΕΙΟ ΤΗΣ ΛΙΣΤΑΣ,Η ΣΤΟ NULL.

ΑΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

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

Δεδομένο λίστας

Δείκτης στο επόμενο στοιχείο της λίστας.

```
void show(node *curr)
{
    while(curr!=nullptr)
    {
        std::cout<<curr->data<<" ";
        curr=curr->next;
        //Προχωράω στο επόμενο στοιχείο της λίστας.
    }
    std::cout<<std::endl;
}
```

Εμφάνιση στοιχείων της λίστας

Προχωράω κάθε φορά στο επόμενο iteration(επανάληψη),με βάση τον δείκτη next του κάθε κόμβου.Αν ο δείκτης next έχει την τιμή NULL σημαίνει ότι έφτασα στο τέλος της λίστας

ΑΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

```
void pushback(node *cr,int data)
{
    node *newnode=new node{data,nullptr};
    while(cr->next!=nullptr)
    {
        cr=cr->next;
    }
    cr->next=newnode;
}
```

Εισαγωγή στοιχείου στην λίστα.

Δημιουργία νέου κόμβου.

Ψάχνω το τελευταίο στοιχείο στην λίστα.Δηλαδή αυτό που ο δείκτης στο επόμενο στοιχείο του έχει την τιμή NULL.

Ο νέος κόμβος θα είναι το επόμενο στοιχείο του κόμβο που ήταν τελευταίος.

ΔΙΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

- ΔΙΑΦΟΡΑ ΜΕ ΑΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ→ΕΧΟΥΝ ΚΑΙ ΔΕΙΚΤΗ ΠΟΥ ΔΕΙΧΝΕΙ ΣΤΟ ΤΕΛΕΥΤΑΙΟ ΣΤΟΙΧΕΙΟ ΤΗΣ ΛΙΣΤΑΣ.

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

ΔΕΔΟΜΕΝΟ ΤΟΥ ΚΟΜΒΟΥ

ΔΕΙΚΤΗΣ ΣΤΟ ΠΡΩΤΟ ΣΤΟΙΧΕΙΟ ΤΗΣ ΛΙΣΤΑΣ

ΔΕΙΚΤΗΣ ΣΤΟ ΤΕΛΕΥΤΑΙΟ ΣΤΟΙΧΕΙΟ ΤΗΣ ΛΙΣΤΑΣ

ΔΙΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

```
void push_back(node **tail,int data)
{
    node *newnode=new node{data,nullptr};
   *tail->next=newnode;
   *tail=newnode;
}
```

Κατασκευή καινούργιου κόμβου

Επόμενο στοιχείο του τελευταίου στοιχείου γίνεται ο καινούργιος κόμβος

Ο δείκτης στο τελευταίο στοιχείο δείχνει στον καινούργιο κόμβο.

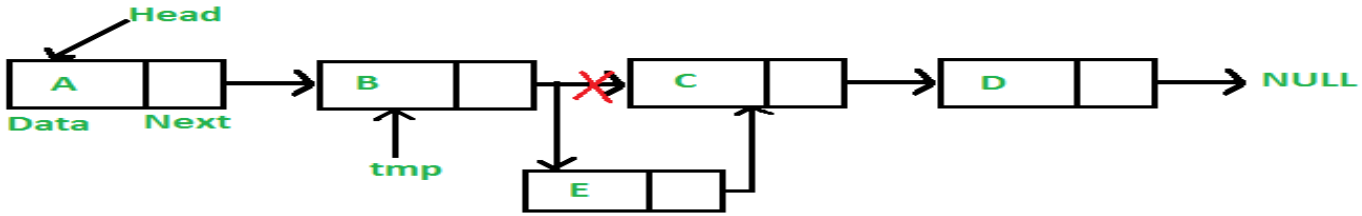
ΠΟΛΥΠΛΟΚΟΤΗΤΑ

- ΑΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ
  - ❑ ΕΙΣΑΓΩΓΗ→O(1)
  - ❑ ΔΙΑΓΡΑΦΗ→O(1)
  - ❑ ΑΝΑΖΗΤΗΣΗ→O(N)

- ΔΙΠΛΑ ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ
  - ❑ ΕΙΣΑΓΩΓΗ→O(1)
  - ❑ ΔΙΑΓΡΑΦΗ→O(1)
  - ❑ ΑΝΑΖΗΤΗΣΗ→O(N)

ΑΛΛΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΛΙΣΤΩΝ

- ΔΙΑΓΡΑΦΗ




```
void delete(node *curr,int pos)
{
    int i=1;
    while(i<pos-1)
    {
        curr=curr->next;
        i++;
    }
    node *delnode=curr->next;
    node *repnode=curr->next->next;
    curr->next=repnode;
    delete delnode;
}
```

Εύρεση προηγούμενο στοιχείου από αυτό που θα διαγραφεί

Ευρεση επόμενου στοιχείου από αυτό που θα διαγραφεί

Αποδέσμευση μνήμης για το στοιχείο που θέλω να διαγράψω και σύνδεση προηγούμενου στοιχείου με το επόμενο του διαγραμένου στοιχείου.

ΚΩΔΙΚΕΣ ΣΕ ΛΙΣΤΕΣ

CUSTOM LISTS

- A.LIST1.CPP
- B.LIST2.CPP
- C.LIST3.CPP
- D.LIST4.CPP
- E.LIST5.CPP

STL

- A.STL1.CPP
- B.STL2.CPP
- C.STL3.CPP