Λίστες

Διονύσης "dionyziz" Ζήνδρος

Camp Προετοιμασίας ΠΔΠ

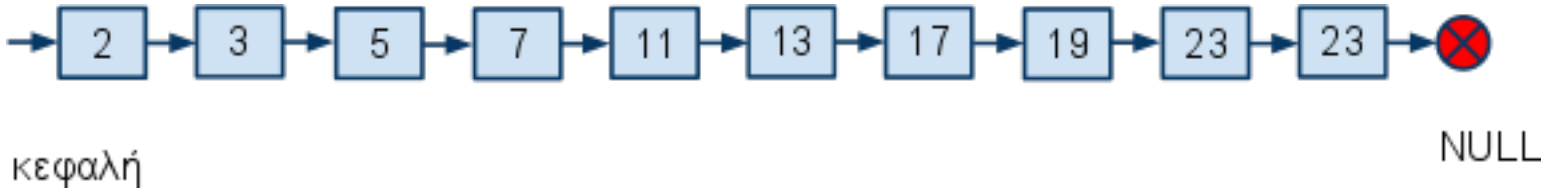
Άνοιξη 2011

Λίστες

- Αντικείμενα ίδιου τύπου σε σειρά
- Παρόμοιες με τους πίνακες με κάποιες διαφορές
- Πλήθος στοιχείων μεταβλητό
- Όχι συνεχείς θέσεις μνήμης

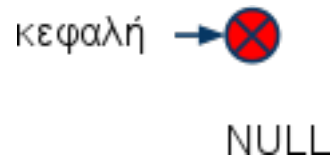
- Οι πρώτοι 10 πρώτοι αριθμοί:
  2, 3, 5, 7, 11, 13, 17, 19, 23, 29

Λίστες



- Κάθε στοιχείο αποθηκεύεται σε διαφορετική θέση μνήμης (όχι συνεχείς)
- Με κάθε στοιχείο αποθηκεύεται δείκτης στο επόμενο
- Εμείς κρατάμε μόνο ένα δείκτη στο πρώτο (κεφαλή)
- Ο δείκτης του τελευταίου στοιχείου δείχνει στο NULL

Η κενή λίστα



- Η κεφαλή δείχνει στο NULL

Τα καλά με τις λίστες

- Προσθέτεις και διαγράφεις άμεσα O(1)
- Έχουν μεταβλητό μήκος

Τα κακά με τις λίστες

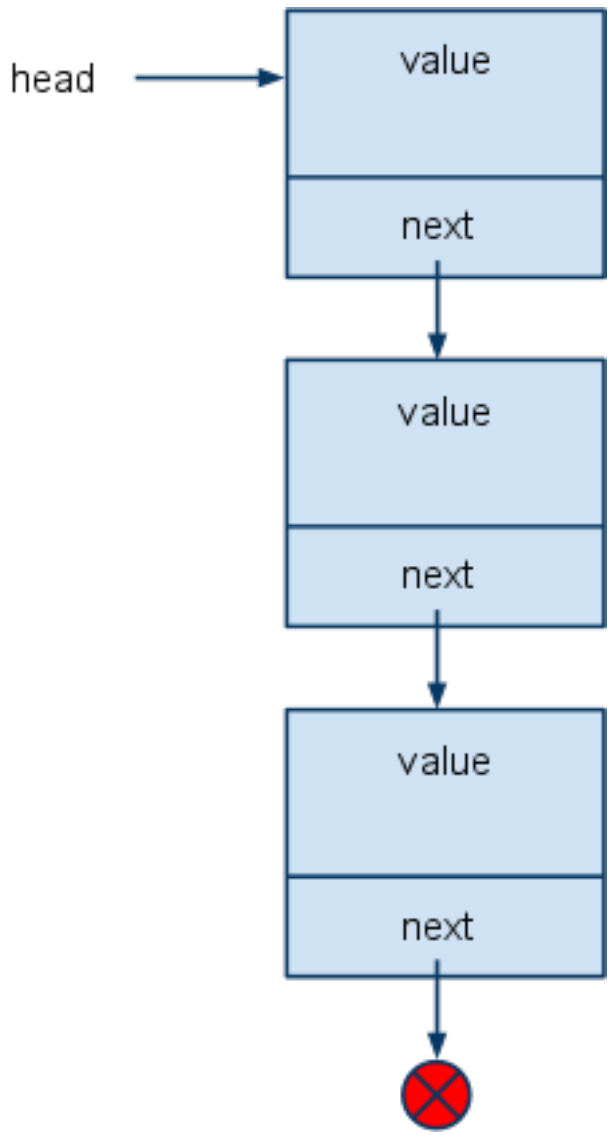
- Δεν μπορείς να πας γρήγορα στο στοιχείο αν ξέρεις τη θέση του.
- Πιάνουν περισσότερο χώρο αφού πρέπει να αποθηκεύσουν δείκτες.

Υλοποίηση

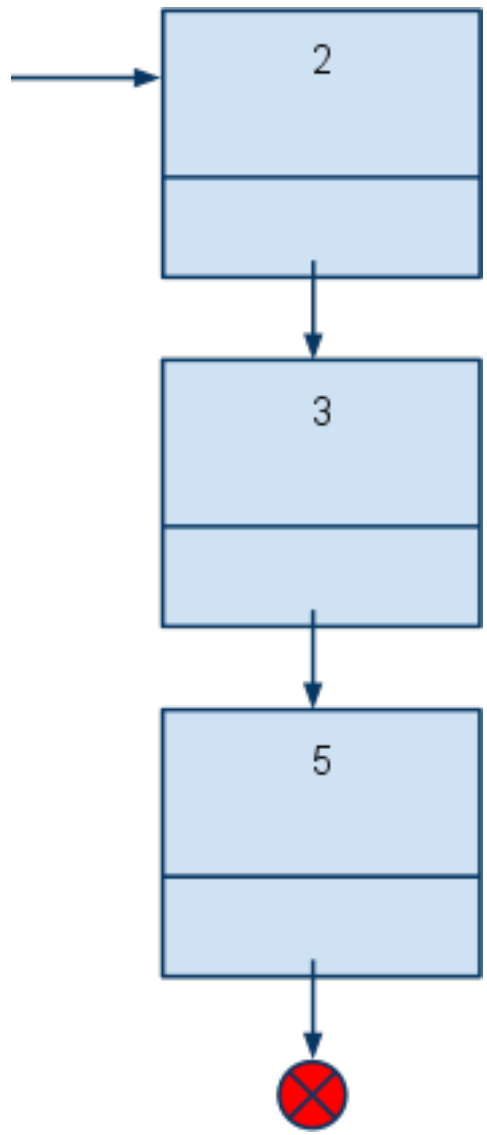
- Συνήθως χρησιμοποιούμε έτοιμες λίστες της C++
- Μπορούμε να τις υλοποιήσουμε και μόνοι εύκολα:

```
struct list_item {
    int value;
    list_item* next;
};

list_item* head;
```

λίστα 3 στοιχείων



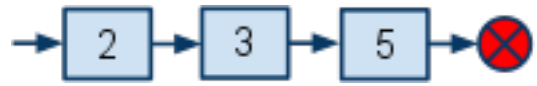
παράδειγμα



Υλοποίηση

```
struct list_item {
    int value;
    list_item* next;
};

list_item* head;
```

```
one = new list_item;
one->value = 2;
two = new list_item;
two->value = 3;
three = new list_item;
three->value = 5;
head = one;
one->next = two;
two->next = three;
three->next = NULL;
```

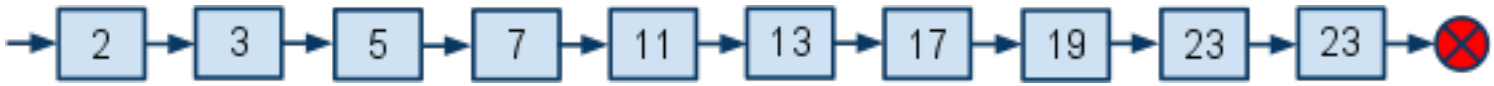


Εκτύπωση λίστας

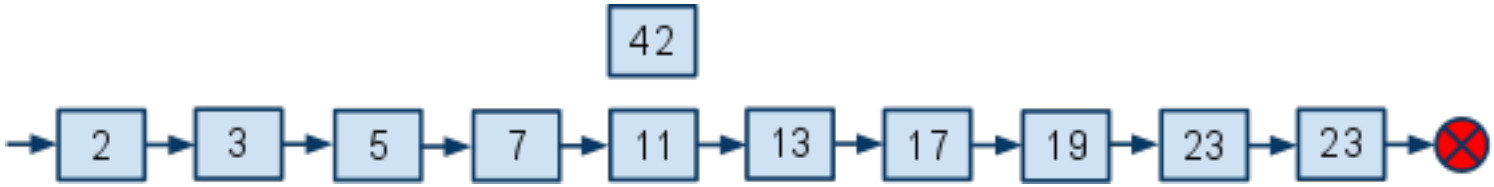
```
struct list_item {
    int value;
    list_item* next;
};

list_item* head;
```

```
void print( list_node* head ) {
    while ( head != NULL ) {
        printf( "%i\n", head->value );
        head = head->next;
    }
}
```

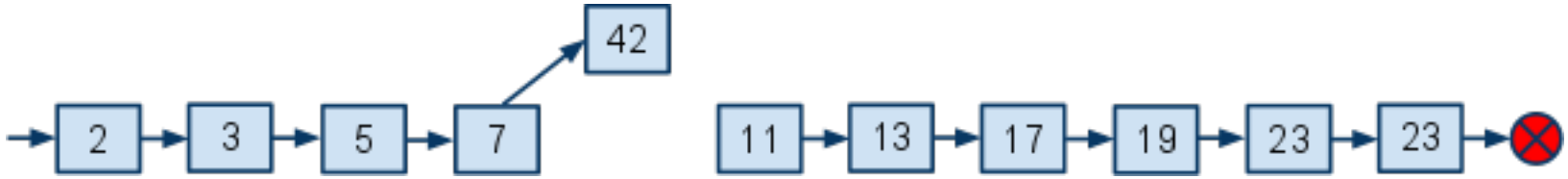
Προσθήκη στοιχείου



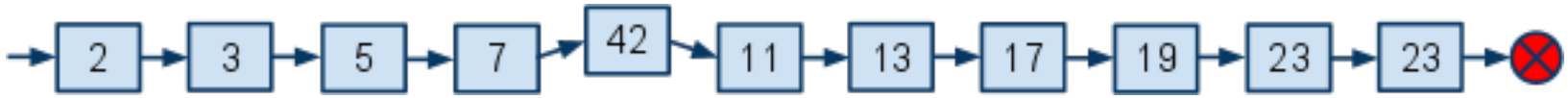
- Φτιάχνω ένα νέο κουτάκι



- Αλλάζω το βέλος του προηγούμενου να δείχνει στο νέο

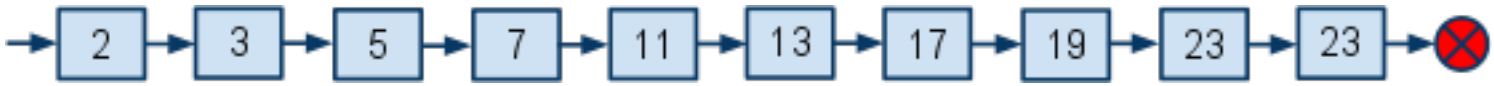


- Αλλάζω το βέλος του νέου να δείχνει στο επόμενο

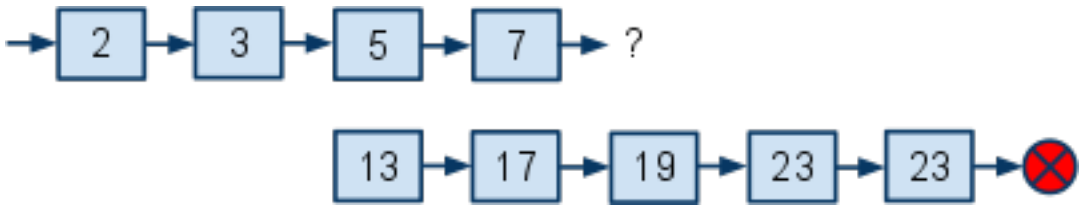


Χρόνος O(1)

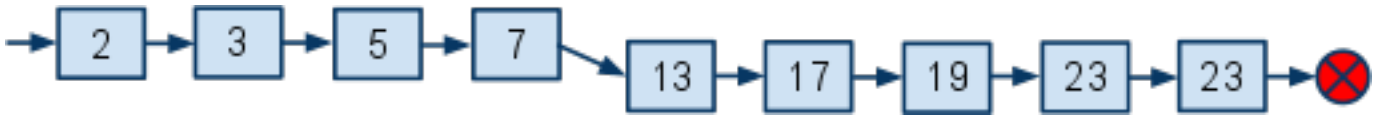
Διαγραφή στοιχείου



- Διαγράφω το κουτάκι



- Αλλάζω το βέλος του προηγούμενου να δείχνει στο επόμενο



Χρόνος O(1)

Χρόνοι διαγραφής/προσθήκης

- Οι καλοί χρόνοι O(1) στην διαγραφή και προσθήκη στοιχείου επιτυγχάνονται μόνο αν γνωρίζω ήδη τον δείκτη στο:
  - Προηγούμενο στοιχείο από εκείνο που θέλω να διαγράψω
  - Στοιχείο μετά από το οποίο θέλω να προσθέσω
- Διαφορετικά θα πρέπει να τα αναζητήσω σε O(N)

Αντιστροφή λίστας

- Δημιουργούμε μια νέα λίστα που έχει τα βέλη ανάποδα

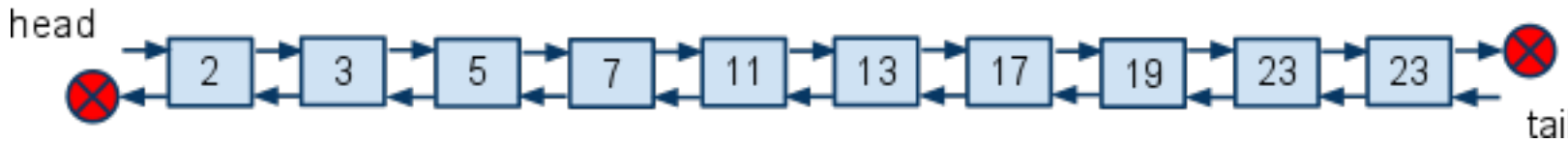
Χρόνος O(N)

Αντιστροφή λίστας

```
list_node* head;
list_node* reversed = NULL;
list_node* prev = NULL;

while ( head != NULL ) {
    reversed = new list_node();
    reversed->value = head->value;
    reversed->next = prev;
    head = head->next;
    prev = reversed;
}
```

Διπλά συνδεδεμένες λίστες

- Ίδιες με τις απλές, μόνο που έχουν βέλη και ανάποδα



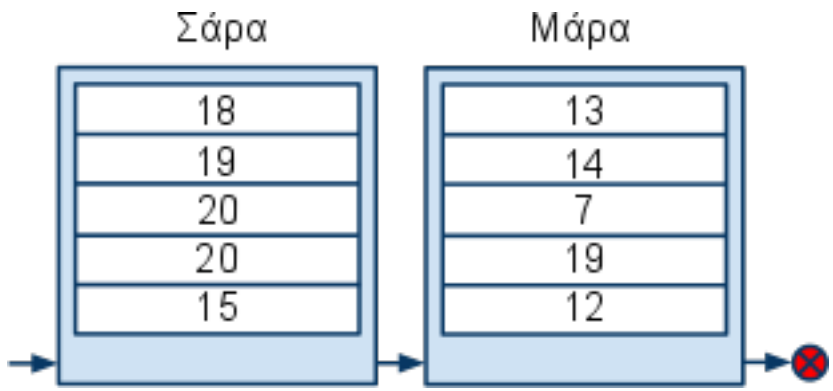

```
struct list_item {
    int value;
    list_item* next;
    list_item* prev;
};

list_item *head, *tail;
```

Λίστες και πίνακες

- Λίστα από πίνακες:

Λίστα μαθητών με βαθμούς σε 5 μαθήματα

```
struct list_item {
    int grades[ 5 ];
    list_item* next;
};
```



Λίστες και πίνακες

- Πίνακας από λίστες:

Ποδοσφαιριστές σε 4 ομάδες

```
struct list_item {
    string player;
    list_item* next;
};

list_item teams[ 4 ];
```