24th February 2013

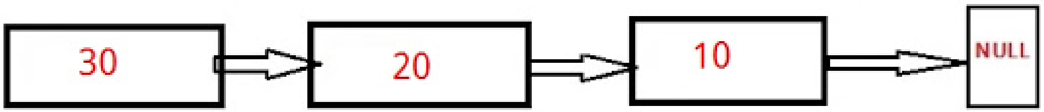
কোডিং লিংকড লিস্ট

আগের পোস্ট [http://alavolacoder.blogspot.com/2013/02/blog-post_19.html] পড়ার পর যাদের মাথায় লিংকড লিস্ট ঢুকে গেছে আর হাত চুলকানো শুরু হয়ে গেছে কোডিং করার জন্য তাদের জন্যই এই পোস্ট। :)
আবারো স্মরণ করিয়ে দেয়া দরকার, লিংকড লিস্ট কোড করতে হলে অবশ্যই স্ট্রাকচার আর পয়েন্টার জানতে হবে।
আর এই পোস্টে শুধু লিংকড লিস্টই না একই সাথে এ্যারেতেও insert, delete, search অপারেশন কোড দেখিয়ে দিচ্ছি, যাতে করে দুইটা ডাটা স্ট্রাকচারের তুলনা করে বুঝতে সুবিধা হয়।

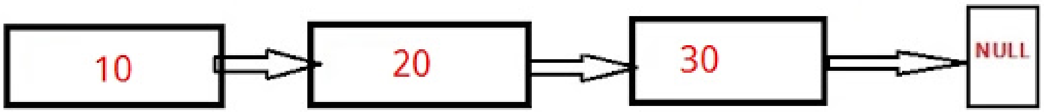
তো শুরু করি-

Insert:

লিংকড লিস্ট বোঝানোর সময় বলেছিলাম আমাদের যখনই কোনো নতুন একটা ভ্যালু লিস্টে যোগ করার দরকার পড়বে, আমরা একটা নতুন নোড তৈরি করে সেটাকে লিস্টের শেষে যোগ করে দিবো। কিন্তু কোডে আমরা যদি একাজ করি তাহলে একটা সমস্যা হয়। সেটা হলো, আমাদের লিস্টটা আমরা যেরকম চাই সেরকম না হয়ে উল্টা হয়ে যায়। মানে আমরা যদি ১০,২০,৩০ ঢোকাতে চাই আমাদের লিস্টে, তখন আমাদের লিস্টটা দেখতে হবে এই রকম-



[http://1.bp.blogspot.com/-dS2Z71JXnhM/USjwyIAsL5I/AAAAAAAAAEw/_l0oMdMWmg8/s1600/linkedlist.jpg]
ইচ্ছা করলে এভাবেও কোডে এটাকে ইমপ্লিমেন্ট করা যায়। কিন্তু আরো কিছু কাজের সুবিধার জন্য আমরা কাজটা এভাবে না করে অন্যভাবে করবো। আমরা আমাদের নতুন ভ্যালুকে শেষে যোগ না করে প্রথমে যোগ করে দিবো। মানে NULL এর জায়গায় নতুন ভ্যালু যোগ করে সেটার শেষে NULL যোগ করে দিবো। যেটা এরকম হবে-



[http://4.bp.blogspot.com/-1goAX6NUoKs/USjyJuzQ3VI/AAAAAAAAAE8/T95N9ryVrK8/s1600/linkedlist2.jpg]
এটাকে এ্যারের মত মনে করেই কাজ করা সহজ হবে।

কোড করার জন্য প্রথমেই আমাদের কিছু জিনিস লাগবে।

```
#include<stdio.h>
#include<stdlib.h>
```

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

struct node *head=NULL,*last=NULL;
```

node structure টা কি সেটা আগের পোস্টেই বলেছি। এখানে node structure টাইপের ২ টা পয়েন্টার লাগবে আমাদের লিস্টের শুরু আর শেষ বোঝার জন্য। পয়েন্টার ২টা প্রথমে NULL, মানে তারা কাউকে পয়েন্ট করছে না। পয়েন্টার ২ টা গ্লোবাল রেখে দিলাম, ইচ্ছে করলে লোকাল ও নেয়া যাবে, তবে তখন ফাংশনে এই পয়েন্টার গুলো নিয়ে কাজ করতে হলে এই পয়েন্টারগুলোকে ফাংশনে প্যারামিটার হিসেবে পাঠাতে হবে। আর <stdlib.h> টা লাগবে পরবর্তীতে malloc ব্যবহার করার জন্য।

এখন আমাদের insert এর জন্য আমরা একটা ফাংশন বানাতে পারি এরকম-

```
void insert(int value)
{
    struct node *tmp;

    tmp=(struct node *)malloc(sizeof(node));

    tmp->val=value;
    tmp->next=NULL;

    //for the first element in the list
    if(head==NULL)
    {
        head=tmp;
        last=tmp;
    }
    else
    {
        last->next=tmp;
        last=tmp;
    }
}
```

যেই value টা লিস্টে যোগ করবো সেটা parameter হিসেবে ফাংশনে পাঠাবো। এখানে আমরা প্রথমে একটা node বানিয়ে নিচ্ছি tmp নামে যেটার জন্য memory allocate করা হচ্ছে আর সেটায় ভ্যালু রাখা হচ্ছে। তারপর সেটাকে পয়েন্টার দিয়ে পয়েন্ট করার মাধ্যমে লিস্টে যোগ করা হচ্ছে। এখানে প্রথমে একটা চেকিং এর কাজ করার হচ্ছে, কারণ আমরা যখন আমাদের লিস্টে প্রথম ভ্যালুটা যোগ করবো তখন আমাদের আদৌ কোনো লিস্ট নেই। তাই সেক্ষেত্রে head and last দুইটা পয়েন্টারই একই নোডকে পয়েন্ট করছে। এরপর নতুন কোনো ভ্যালু যোগ করতে হলে, head এর কোনো পরিবর্তন হবে না, শুধু last এর সাথে নতুন নোড যোগ করে, last কে update করা হচ্ছে।
একই insert এর কাজ যদি আমরা Array দিয়ে করতাম, তাহলে এরকম হতো-

```
void insert(int value)
{
    arr[last]=value;
```

```
    last++;
}
```

last হচ্ছে এ্যারের last position যেখান পর্যন্ত ডাটা ঢোকানো হয়েছে।
কি, array দিয়ে খুব সহজ, তাই না? তাহলে তো array use করলেই হয়।
ধীরে বৎস, এখনো অনেক কাজ বাকি। আগে শেষ করি, তারপর দেখা যাবে। :)
insert তো হলো, এখন যদি আমাদের লিস্টটাকে আমরা দেখতে চাই, তাহলে নীচের মত একটা ফাংশন লিখে নিলেই হলো-

```
void printlist()
{
    struct node *tmp=head;
    while(tmp!=NULL)
    {
        printf("%d\n",tmp->val);
        tmp=tmp->next;
    }
}
```

এখানে head হচ্ছে লিস্টের শুরু, সেখান থেকে শুরু করে লিস্টের শেষ অর্থাৎ NULL পাওয়া পর্যন্ত লিস্টে যা আছে সেগুলোকে পাওয়া যাবে। প্রতিবার একটা নোডের ভ্যালু প্রিন্ট করার পর নোড পয়েন্টারটাকে tmp=tmp->next দিয়ে পরের নোডে পয়েন্ট করা হচ্ছে। একটা জিনিস লক্ষ্যণীয়, এখানে প্রথমে কিন্তু tmp এর জন্য কোনো memory allocate করা হচ্ছে না। কারণ, আমরা এখানে কোনো নতুন নোড বানাচ্ছি না, আগের বানানো নোড head কে পয়েন্ট করছি।

Search:
আমরা যদি চাই, আমাদের লিস্টে একটা ভ্যালু আছে কি না সেটা দেখতে, তাহলে নীচের মত করে একটা ফাংশন লিখতে পারি-

```
int search(int value)
{
    struct node* tmp=head;

    while(tmp!=NULL)
    {
        if(tmp->val==value)
            return 1;

        tmp=tmp->next;
    }

    return 0;
}
```

যেই ভ্যালুটাকে search করতে চাই সেটাকে ফাংশনে প্যারামিটার হিসেবে পাঠিয়ে দিলেই হলো। এই কাজটা printlist() ফাংশনের মতই প্রায়। যদি কোনো নোডে কাংখিত ভ্যালুটা পাওয়া যায় তাহলে ফাংশন থেকে একটা 1 or true ভ্যালু রিটার্ন হবে, নাহলে একটা 0 or false ভ্যালু রিটার্ন হবে। যেটা দেখে বোঝা যাবে লিস্টে ওই ভ্যালুটা আছে কিনা।

এই কাজটাই এ্যারেতে করা যাবে এভাবে-

```
int search(int value)
{
    int i;
    for(i=0;i<last;i++)
    {
        if(arr[i]==value)
            return 1;
    }

    return 0;
}
```

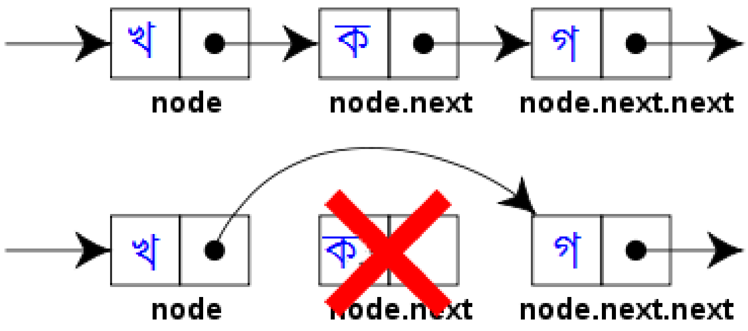
Delete:

মনে হতে পারে যে, লিংকড লিস্ট থেকে একটা ভ্যালু delete করা খুবই কঠিন। কারণ একটার পর একটা লিংক করা, এখন এখানে মাঝখান থেকে একটা লিংক থেকে আলাদা করলে যদি বাকিগুলোও আলাদা হয়ে যায়?

হুমম, ব্যাপারটা একটু চিন্তারই।

কিন্তু একটু মাথা খাটালেই আশা করি ব্যাপারটা সহজ হয়ে যাবে। আমরা যদি আগে খুঁজে বের করি কোন ভ্যালুটাকে delete করবো, মনে করি সেই নোডটা 'ক', আর 'ক' এর ঠিক আগের নোড হচ্ছে 'খ' এবং 'ক' এর ঠিক পরের নোড হচ্ছে 'গ'। তাহলে আমরা যদি 'খ' এর next পয়েন্টার 'ক' থেকে সরিয়ে সেখানে 'গ' কে বসিয়ে দেই, তাহলে তো মাঝখান থেকে 'ক' গায়েব হয়ে যাচ্ছে। কারণ, আমরা যখন 'খ' তে আসবো, তখন 'খ' এর next এ গেলে তো 'গ' পাবো, 'ক' কে আর পাবো না।

ঠিক এরকম-



[http://3.bp.blogspot.com/-WjkHhy_NCU4/USkY-RyB7aI/AAAAAAAAAFQ/Ud84fnBlCe8/s1600/delete.png]

তারমানে, লিস্ট থেকে delete করাও আসলে তেমন কঠিন কিছু না। আমরা যদি কোড করি, তাহলে এরকম হবে-

```
void deletenode(int value)
{
    struct node* tmp=head;
    struct node* prev=NULL;

    while(tmp!=NULL)
    {
        if(tmp->val==value)
        {
            if(prev==NULL)
            {
                head=tmp->next;
            }
```

```
            else
                prev->next=tmp->next;

            break;
        }

        prev=tmp;
        tmp=tmp->next;
    }
}
```

এখানে কোনো নোডের আগের নোডকে locate করার জন্য prev একটা পয়েন্টার ব্যবহার করা হচ্ছে। আর delete করার সময় যদি প্রথম নোডটাই delete করতে হয় মানে prev==NULL হয়, তাহলে শুধু head (প্রথম নোডের পয়েন্টার) কে সরিয়ে next নোড এ পয়েন্ট করলেই হবে, যেহেতু এর কোনো prev নোড নেই।
এছাড়া অন্যক্ষেত্রে, যেভাবে উপরের বুদ্ধিতে prev নোড এর পয়েন্টার prev->next কে বর্তমান নোডের next এ tmp->next এ পয়েন্ট করলেই হবে, মাঝখান থেকে tmp হাওয়া(!) হয়ে যাবে, কারণ কেউ তাকে পয়েন্ট করছে না।
Array তে delete করতে চাইলে-

```
void deleteval(int value)
{
    int i,pos=-1;
    for(i=0;i<last;i++)
    {
        if(arr[i]==value)
        {
            pos=i;
            break;
        }
    }

    if(pos>=0)
    {
        for(i=pos+1;i<last;i++)
        {
            arr[i-1]=arr[i];
        }
        last--;
        printf("%d value deleted\n",value);
    }
    else
        printf("%d value not found\n",value);
}
```

Array তে মাঝখান থেকে delete করতে চাইলে একটা প্রবলেম হয়, সেটা হলো পরের ভ্যালুগুলোকে শিফট করে একঘর আগে নিয়ে আসতে হয়। যেটা অনেক বড় অ্যারেতে করতে হলে inefficient হয়, কারণ সময় নষ্ট হয়। লিংকড লিস্টে এই অসুবিধাটা নেই।

অনেক তো হলো, লিংকড লিস্ট আর এ্যারে নিয়ে বকর বকর। তবে কথা হচ্ছে, এ্যারে এবং লিংকড লিস্ট কোনোটাই হেলা করার মত না। কারণ, যখন যেটা ব্যবহার করা সুবিধাজনক তখন কোডের স্বার্থে সেটাকে ব্যবহার করতে হবে, এছাড়াও অন্য ডাটা স্ট্রাকচারগুলোকে(স্ট্যাক, কিউ, হিপ, ট্রি ইত্যাদি) কোডে ইমপ্লিমেন্ট করতে হলে এ্যারে এবং লিংকড লিস্ট লাগবে।

সুতরাং, চলতে থাকুক কোডিং...আর উপরের পুরো কোড পাবে নীচের লিংকে।
Array [http://ideone.com/E2UPWI]
Linked List [http://ideone.com/yJLS2M]

Happy coding.........................

Posted 24th February 2013 by Ala vola

Labels: data structure, linked list

4 View comments

Hasan Abdullah July 2, 2013 at 11:08 PM

দিলাম শুরু করে লিংকড লিস্ট... যা থাকে কপালে...!!!

Reply

Rayhan May 17, 2015 at 12:26 AM

আপনাকে অনেক অনেক ধন্যবাদ ভাই.... লিঙ্ক লিস্ট বুঝতাম না এখন অন্যদের বুঝাতে পারি পুরা ক্রেডিটই টা আপনার ... :)

Reply

Md. Shafiul Alam Sagor May 20, 2015 at 9:53 PM

osthir............... thanks a lot

Reply

Mohammad Nazmul June 24, 2016 at 1:01 PM

আপনার বোঝানোর কৌশল খুব ভাল। আরো গভীরের প্রোগ্রামিং পড়াশুনা নিয়ে বিস্তরভাবে লিখবেন আশা করি। আপনাকে ধন্যবাদ।

Reply

Enter your comment...

Comment as: Ashikur Rahma ▼ Sign out

Publish Preview ☐ Notify me