লিংকগুলো

টিউটোরিয়াল
প্রোগ্রামিং রিসোর্সেস

বেসিক সি++

ভারশন শূণ্য - সর্বশেষ আপডেট নভেম্বর ৪, ২০০৯

প্রথম অংশ - বেসিক সি++

দ্বিতীয় অংশ - স্ট্যান্ডার্ড টেম্প্লেট লাইব্রেরী

আমরা যারা কন্টেস্ট প্রোগ্রামার, আমরা যতটা না একজন প্রোগ্রামার তার চে' অনেক বেশি হচ্ছি একজন প্রবলেম সলভার। আমরা কিন্তু ঠিক প্রথাগত কম্পিউটার সায়েন্টিস্ট ও নই, যারা মোটামুটি একটা কঠিন প্রবলেম পৃথিবীর কোণাকান্চি থেকে খুঁজে বের করে, সেটার পেছনে একটা বছর ধাই করে উড়িয়ে দেবে, তারপর গালটা অনেকক্ষণ চুলকায়ে টুলকায়ে মনে করতে বসবে শেষ কবে দাঁড়ি কাটার সৌভাগ্য হয়েছিল। আবার আমরা ইনডাস্ট্রি প্রোগ্রামারদের মতও নই - যারা মোটামুটি একমাস ধরে একই চুইংগাম চাবাতে চাবাতে একটা বোরিং কোড অলস হাতে বসে বসে লিখতে থাকে। কোড লিখতে লিখতে ঘুমাই পড়ে, আবার ঘুম থেকে উঠে আবার একই কোড লিখতে থাকে।

আমরা কন্টেস্টে করি অনেক বেশি চাপের মধ্যে, আমাদের প্রবলেম পড়ে বুঝতে হয় কি চাওয়া হচ্ছে, আমাদের অ্যালগরিদম ডিজাইন করতে হয়, আমাদের একটা কোড ডিজাইন করতে হয়, এবং আমরা সেটা করি মোটামুটি কোন ধরণের ফ্লোচার্ট হাবিজাবি না এঁকেই। এবং তারপর আমাদের কোডগুলো বসে বসে আমাদের ডিবাগও করতে হয়, টেস্টিং ও করতে হয়। তো ব্যাপারটা দাড়াচ্ছে, আমরা প্রবলেম সলভার, আমরা কম্পিউটার সায়েন্টিস্ট, আমরা কোডার, আমরা সফটওয়্যার আর্কিটেক্ট, আমরা ডিবাগার - কিন্তু আমরা পুরোপুরি এগুলোর কিছুই নই। কিন্তু একই সাথে আমরা এগুলোর সবকিছু।

পেতর মিত্রিচেভ বলতো, কন্টেস্ট প্রোগ্রামাররা হচ্ছে ফরমুলা ওয়ান রেসের ড্রাইভারদের মত। যে লোকটা একটা ট্রাক চালায়, সেও একটা ড্রাইভার, আর যে ফরমুলা ওয়ানে গাড়ি চালায় সেও ড্রাইভার, কিন্তু দুটার মধ্যে আকাশ পাতাল পার্থক্য। আমরা ভাবতেই পারি ফরমুলা ওয়ানে একটা গাড়ি নিয়ে ফালাফালি করার মধ্যে এমন কি আছে, বরং ট্রাক চালানোই অনেক কাজের। কিন্তু দুইটার মধ্যে ঠিক এভাবে তুলনা করা যায় না। যারা একবার ফরমুলা ওয়ানে গাড়ি ছোটানো শুরু করে, শুধু তারাই বুঝতে পারে এক্সাইটমেন্টটা কেমন, এবং সেটা কতটা আলাদা একটা বোকাসোকা হাইওয়েতে অন্ধকার রাতে তারা দেখতে দেখতে আর

ঘুমাতে ঘুমাতে একটা স্টুপিড ট্রাক চালানোর সাথে।

এই লেখাটাতে আমি সি++ নিয়ে লিখবো। আর লিখবো কন্টেস্ট প্রোগ্রামারদের ঠিক কতটুকু সি++ জানা দরকার সেটা নিয়ে, আর ঠিক সেইটুকুই। কন্টেস্ট প্রোগ্রামাররা আর সবার মতো না, যে তাদেরকে একটা মোটাসোটা স্বাস্থ্যবান সি++ এর বই ধরিয়ে দিয়ে "নাচো" টাইপের কিছু একটা বলে আমি উধাও হয়ে যাবো।

আর STL নিয়ে যা কিছু লিখবো তার বেশিরভাগ জুড়ে থাকবে ডাটা স্ট্রাকচার। আসলে কি, কম্পিউটারে আমরা প্রচুর ডাটা জমা করি। এখন ডাটাগুলোকে কম্পিউটারে কিভাবে রাখলে আমাদের কাজ করতে সুবিধা হবে, আমরা যদি সেটা ভেবে সেভাবে ডাটাগুলোকে সাজিয়ে রাখি, সেই বুদ্ধিমানের মত করে ডাটা সাজানোটাকেই আসলে ডাটা স্ট্রাকচার বলে। একজন নতুন কন্টেস্ট্যান্টের কাছে ডাটা স্ট্রাকচার নিয়ে লাফালাফি করাটা খুব অদ্ভূত দেখাবে। এবং আমি যদি পুরোপুরি সবকিছুই ব্যাক্ষা করতে যাই, যে কোনটা কেন দরকার, কিভাবে দরকার, তাহলে এই লেখাটা বিশাল হয়ে যাবে আর আমার আঙ্গুল ব্যাথা করবে এটা লিখতে। তো আমি সব লিখে যাচ্ছি, যাতে তুমি শুধু জিনিসগুলো জেনে রাখতে পারো, কোন না কোন দিন হঠা□ দেখবে একটা প্রবলেম সলভ করতে তোমার কিছু জানা ডাটা স্ট্রাকচার ব্যবহার করতে হচ্ছে! :)

প্রথম অংশ - সি++ নিয়ে বকরবকর

স্ট্রাকচার

আমাদের মাঝে মাঝেই একসাথে অনেকগুলো ডাটা রাখতে হয়। ধরো, তোমার ফোনবুকে প্রতিটা নামের সাথে একটা করে নাম্বার আছে। খুব সহজ। সমস্যাটা হয়ে গেলো, পৃথিবী অনেকটুকু এগিয়ে গেছে, এখন আর কেউ বাটন গুতোগুতি করে ফোন করে না, সবাই ধাই ধাই করে মেইল করে। তো তোমার ফোন নাম্বারের পাশাপাশি মেইল অ্যাডরেসও রাখার দরকার হচ্ছে। তো ব্যাপারটা কি দাঁড়াচ্ছে, তুমি একই সাথে দুইটা ডাটা রাখছো।

আমরা এই জিনিসটাকে বলি স্ট্রাকচার। এটা লিখি অনেকটা এভাবে।

```
struct data {
    char name[20];
    int number, email_num;
};
```

অবশ্যই ইমেইল আইডি ইনটেজার হয় না - আমি শুধু বোঝানোর লিখছি।

তো আগে আমাদের কিছু ডাটা টাইপ ছিল int, char, double এই ধরনের। আমরা এইমাত্র আমাদের নিজেদের জন্য একটা ডাটা টাইপ তৈরী করলাম, যার নাম হচ্ছে data । ডাটা হচ্ছে

একটা মিষ্টির প্যাকেট যার মধ্যে বিভিন্ন ধরণের অনেকগুলো মিষ্টি আছে।এভাবে চিন্তা করলে যদি দেখো তোমার খুব মিষ্টি খেতে ইচ্ছে করছে, তাহলে চিন্তা করো, data হচ্ছে একটা বাক্স যেটার ভিতর আরো কিছু বাক্স(char, int) আছে।

মাঝে মাঝে আমাদের একই জিনিস বারবার ব্যবহার করতে হয়। ধরো, তোমার প্রতিদিন কিছু নতুন মানুষের সাথে দেখা হচ্ছে, ওরা হচ্ছে তোমার নতুন ডাটা - কিন্তু নতুন মানুষগুলোই তোমার সবকিছু নয়, তোমার অনেক কাজ আছে করার মতো আর অনেক অনেক কিছু। তো তোমার দিনটা অনেকটা এরকম কিছু একটা হবে।

```
struct day {
    data new_people[100];
    int total_new_people;
    worklist todo[100];
};
```

যেখানে worklist হয়তো আরেক টাইপের ডাটা। day হলো একটা বাক্স যেটার ভিতরে data টাইপের কিছু বাক্সও আছে।

ব্যাপারটা কঠিন লাগছে? হুমম, এটার নামই হলো অবজেক্ট অরিয়েন্টেড প্রোগ্রামিং। যারা প্রথম প্রথম এটা বুঝতে পারে, তারা তার পরের মোটামুটি কয়েক সপ্তাহ যেখানে সেখানে এটা নিয়ে আঁতলামি করতে খুব ভালোবাসে। তো জিনিসটা কঠিন লাগাটাই স্বাভাবিক।

কিন্তু আমার মনে হয় না, জিনিসটা একটুও কঠিন। জিনিসটা ফালতু রকমের সোজা। আমরা একটা বাক্সের ভিতর আরেকটা বাক্স ভরতেই পারি, আমাদের যত ইচ্ছে খুশি, ততগুলো বাক্সের ভিতর ততগুলো করে বাক্স ভরে, তারপর ওই বাক্সগুলোর ভিতর আরো গাদা গাদা বাক্স ভরতেই পারি। তারপর যখন মনে হবে "উরি বাবা, পাগল হয়ে যাচ্ছি", তখন উইনঅ্যাম্প খুলে অর্ণবের গান শুনতেই পারি, (যে অলরেডি অবজেক্ট অরিয়েন্টেড প্রোগ্রামিং পড়তে গিয়ে পাগল হয়ে গেছে)

বাক্সে বাক্সে বন্দি বাক্স

বাক্সে বাক্সে বন্দি বাসা,

বাক্স নিয়ে টাক show করা,

বাক্সতে সব স্বপ্ন আশা।

অর্ণবের মত একটা ননকোডার মানুষ যদি এই জিনিস বুঝতে পারে, আমরা কেন পারবো না? আরে, আমরা হলাম গিয়ে কন্টেস্ট প্রোগ্রামার, এলিট ক্লাসের মানুষজন!

বেশতো, মূল ব্যাপারটা কিন্তু এরকমই। কিন্তু আরেকটু ভেবে দেখো, প্রতিদিন কি তুমি বসে বসে মাথার ভিতর খালি ডাটাই ঢুকাও, নাকি কিছু করোও? যদি কিছু করো, তবে সেটা অবশ্যই

কিছু কাজ, তাই না? হয়তো তুমি ক্লাসে বসে বসে ক্লাসওয়ার্কও করো, হয়তো প্রতিদিন তোমার মিস তোমাকে এক গাদা অংক দেয়, তুমি বসে বসে যোগ করো। তো তখন ব্যাপারটা কি হচ্ছে

```
struct day {
    data new_people[100];
    int total_new_people;
    worklist todo[100];

    void charm_your_miss() {
        while( true ) do_math();
        while( true ) smile();
    }

    void do_math() {
        int a, b;
        scanf("%d %d",&a,&b);
        printf("%d\n", a+b );
    }

    void smile() {
        printf(" :) ");
    }
};
```

এটা হচ্ছে তোমার দিনেরই একটা অংশ, তাইনা? ডাটার পাশাপাশি তোমাকে কিছু কাজও রাখতে হচ্ছে। নাইলে আসলে মিসকে চার্ম করা যাচ্ছে না, এমনকি হাসাও যাচ্ছে না।

এখন যদি আমার শুধু একটা স্ট্রাকচারের কোন একটা ডাটা অ্যাকসেস করা লাগে। আমরা সেটা করি ডট অপারেটর দিয়ে। ধরো আমার শুধু দরকার total_new_people কে। সেটাকে ধরে আমি শূণ্য করে দিবো। তাহলে আমার মেইন এর ভিতর কোডটা হবে এমন।

```
int main() {
    day a_bright_new_day, mora_day;
    mora_day.total_new_people = 0;
    return 0; // already more gesi! :(
}
```

আচ্ছা, এখন ধরো, আমার দিনটা খুব সুন্দর, আমার মিসকে চার্ম করা লাগতেসে না, কারণ স্কুল ছুটি। তো আমি কি করবো, আমি সারাদিন বসে বসে বাবল ফুলাবো আর হাসবো। তো আমি একটা কাজ করবো, হাসবো। (বাবল ফুলানো অবশ্যই কোন কাজের কাজ না, তাই না? ;)) তখন আমি কাজটাও করবো একই ভাবে। একটা ডট অপারেটর ধরে হাসি মেরে দিবো।

```
int main() {
    day a_bright_new_day, mora_day;
    while( true ) a_bright_new_day.smile();
    return 0; // are baba, din to shesh hobei!
}
```

তো এই হলো ঘটনা।

এখন সমস্যা হলো, প্রতিদিন কতকিছুই ঘটতে পারে। তার উপর ছেলেপুলে বড় হচ্ছে। এখন সবকিছু তো আর সবাইকে বলা যায় না, তো আমরা আসলে সব ডাটা এমন ভাবে রাখতে পারবো না, যে আজাইরা লোকজন এসে বলবে, "চিচিং ফাঁক! এই নাও ডট অপারেটর", তারপর ডট অপারেটর দিয়ে আমাদের সব ডাটা মেরে দিবে।

এই জিনিসটা এড়ানোর জন্য আমরা একটা কি-ওয়ার্ড ব্যবহার করি। চিচিং ফাঁক এর ঠিক উল্টা। এটা হলো - private। তোমার প্রাইভেট জিনিসপাতিতে কারো সাধ্য নাই হাতাহাতি করার। ;) তো জিনিসটা তখন লিখে এভাবে।

```
struct day {
    data new_people[100];
    int total_new_people;
    worklist todo[100];

    private:
        sweetthings bolbo_keno[100];

};
```

জিনিসপাতি পাবলিক করতে চাইলেও কোন সমস্যা নাই। পাবলিক করে দিলেই হবে। সেটা লিখতে হবে এভাবে।

```
struct day {
    data new_people[100];
    int total_new_people;
    worklist todo[100];

    public:
        sweetthings bolboi_to[100];

};
```

এবার একটু জ্ঞান কপচাই। সি++ এ, একটা ফাউ জিনিস আছে, সেটার নাম হলো ক্লাস। ক্লাস একটু মুরুব্বি টাইপের জিনিস। একইভাবেই লিখেও যদিও তারপরও।

```
class day {
    data new_people[100];
    int total_new_people;
    worklist todo[100];

    public:
        sweetthings bolbo_na_vabsilam[100];
```

```
};
```

ক্লাসের ব্যাপারটা হলো, তুমি যদি কিছু না বলে দাও ক্লাস সম্বন্ধে তাইলে ক্লাসের সব ডাটা প্রাইভেট থাকবে। আর স্ট্রাকচারের ক্ষেত্রে ঠিক উল্টা, ওর সব কিছুই পাবলিক, বলে না দিলে। এইজন্যই বললাম, ক্লাস একটু মুরুব্বি টাইপের জিনিস, বেশি ভাবের উপর থাকে।

ফাংশন ওভারলোড

মাঝে মাঝেই আমাদের একই কাজ করতে হয়, অনেক আলাদা আলাদা ভাবে। ধরো নরমাল হাসির ব্যাপারটাই, আমরা একেকজনের সামনে একেকভাবে হাসি। হেডমাস্টারের সামনে গিয়ে খ্যাঁক খ্যাঁক করে হাসতে হাসতে গড়াই পড়ে যাই না আমরা। আবার বন্ধুদের সাথে মুখ টিপে টিপে লাজুক লাজুক হাসিও আমরা দেই না। তো ব্যাপারটা একই কাজ কিন্তু ভিন্ন ভিন্ন জায়গায় ভিন্ন ভিন্ন রকমের কাজ করতে হচ্ছে।

একটা সহজ উদাহরণ হচ্ছে অ্যাবসলুট ভ্যালু নেয়া। কোন সংখ্যার শুধু মান নেয়াটাকে বলে অ্যাবসলুট ভ্যালু নেয়া। যেমন -২০ এর অ্যাবসলুট ভ্যালু হলো ২০। মাইনাস উড়ে গেছে।

```
int absolute_int( int x ) {
    if( x < 0 ) return -x;
    else return x;
}
```

তো ধরো আমার long long এও একই কাজ করা লাগবে। তাইলে আমি লিখবো

```
long long absolute_ll( long long x ) {
    if( x < 0 ) return -x;
    else return x;
}
```

সি++ এ দুইটা আলাদা নাম লেখা লাগে না। এরকম দুইটা একই নামের ফাংশন লিখে দিলেই হয়। সে নিজে নিজে বুঝে নিবে তোমার ডাটা টাইপ দেখে যে তুমি মুচকি হাসি দিতে চাচ্ছো, না খ্যাঁক খ্যাঁক করে হাসতে চাচ্ছো।

```
long long absolute( long long x ) {
    if( x < 0 ) return -x;
    else return x;
}
```

```
int absolute( int x ) {
    if( x < 0 ) return -x;
    else return x;
}
```

হেডার ফাইল

সি++ এ নেমস্পেস ব্যবহার করলে হেডার লিখার জন্য .h মারা লাগে না। সেটা যদি সি++ এর হয়, যেমন algorithm, তাহলে এটা লেখা হবে এভাবে।

```
#include <algorithm>

using namespace std;
```

কিন্তু সেটা যদি সি এর কিছু একটা হয়, আমরা শুরুতে c লিখে দিবো। যেমন stdio.h লেখা হয় এভাবে।

```
#include <cstdio>

using namespace std;
```

সত্যি কথা আমি এসবের কিছুই করি না। আমার একটা জায়গায় যা কিছু মনে পড়ে, যার যার কথা মনে পড়ে, সবাইকে একসাথে লিখে রেখেছি। জাস্ট ধরে কথা বার্তা ছাড়াই একটা কপি মারি।

```
#include<cstdio>
#include<sstream>
#include<cstdlib>
#include<cctype>
#include<cmath>
#include<algorithm>
#include<set>
#include<queue>
#include<stack>
#include<list>
#include<iostream>
#include<fstream>
#include<numeric>
#include<string>
#include<vector>
#include<cstring>
#include<map>
#include<iterator>
using namespace std;
```

এই হলো আমার হেডারদের কাহিনী।

কিন কাউট!

আমি অবশ্য উচ্চারণ করি সি-ইন, সি-আউট।

তুমি যদি একজন সি কোডার হও, তুমি অবশ্যই scanf দিয়ে ইনপুট নাও, আর printf দিয়ে আউটপুট মারো। খুব স্বাভাবিক ব্যাপার, আরে এতে লজ্জা পাওয়ার কিচ্ছু নেই, সবাই তাই করে! কিন্তু ব্যাপারটা হলো, সি তে তোমাকে বলে দিতে হয় তুমি কি ধরণের ডাটা নিয়ে খেলছো, মনে রাখতে হয় %lld, %lu, %c, %s, %d, %lf এইরকম বিচ্ছিরি দেখতে অনেক অনেক কিছু।

সিপিপিতে তুমি শুধু একটা সি-ইন মেরে দিবা, কিচ্ছু মনে রাখা লাগবে না।

যেমন এটা হলো একটা সি কোড

```
int main() {
    int murgi;
    double water;
    long long bank_balance;

    scanf("%d %lf %lld",&murgi, &water, &bank_balance);
    printf("%d %lf %lld",murgi, water, bank_balance);


 return 0;


}
```

একই জিনিস সি-ইন সি-আউট দিয়ে লিখলে এরকম হবে।

```
int main() {
    int murgi;
    double water;
    long long bank_balance;

    cin  >> murgi >> water >> bank_balance;
    cout << murgi << water << bank_balance;

    return 0; // ki ase jibone?
}
```

খালি একটাই সমস্যা। এরা একটু স্লো। কাম কাজ একটু বেশি ধীরে সুস্থ্যে করে।

টেম্প্লেট

টেম্প্লেট কি জিনিস সেটা যদি তুমি বুঝতেই চাও, এই কোডটায় একটা উঁকি মারো। ভয় পেয়েছো? এটা হলো পৃথিবীর এখনকার এক নাম্বার কোডারটার কোড।

```
template<class T> inline T gcd(T a,T b)
```

আচ্ছা, এটা মোটামুটি নির্দোষই লাগছে, তাই না?

আমরা একটু আগে ফাংশন ওভারলোডিং দেখলাম না? ওই জিনিসটাই আরেকভাবে করা যায়, টেম্প্লেট ব্যবহার করে। যেমন ওই অ্যাবসলুট ভ্যালুর কোডটা টেম্প্লেট দিয়ে লিখলে এরকম হতো

```
template<class T> T absolute( T x ) {
    if( x < 0 ) return -x;
    else return x;
}
```

মানে আমার দুই তিনবার একই ফাংশন আলাদা আলাদা করে ডাটাটাইপ পাল্টে লিখতে হতো না। আমি টেম্প্লেটটা দেখিয়ে নিলাম, এই জন্য যে, আমরা একটু পরে STL নিয়ে আঁতলামি শুরু করবো, STL এর সবকিছুতে টেম্প্লেট ব্যবহার করে লেখা, যাতে তুমি ইচ্ছামতো ক্লাস/স্ট্রাকচার ব্যবহার করে নিজের ডাটাটাইপ ব্যবহার করতে পারো, আর সেটা নিয়ে ইচ্ছামতো নাচানাচিও যেন করতে পারো।

পরবর্তী অংশ