

ভূমিকা:

বইটি লেখার শুরু থেকে প্রকাশ হওয়া পর্যন্ত এত বেশি প্রতিবন্ধকতার মুখোমুখি হয়েছিলাম যে, এক সময় ভেবেই নিয়েছিলাম আমার আরও একটা স্বপ্ন আর বাস্তবায়ন হলো না,কিন্তু আজ মনে হচ্ছে এত প্রতিবন্ধকতা ছিল বলেই বইটি প্রকাশ করে আজ আমি এত বেশি আনন্দিত, স্বপ্নের পালে কতটা বাতাস লাগলে সেই স্বপ্নের ডিঙা বাস্তবতার বন্দরে নোংড়া ফেলে- তা অনুধাবন করতে পেরে আজ আমি আনন্দিত। C++ programming শেখার একটা গতানুগতিক ধারাকে ভাঙতে পেরে আজ আমি সত্যিই খুব আনন্দিত। অনেকে মনে করেন C program না জেনে C++ শেখা যায় না। আমার বিশ্বাস আমার লেখায় আমি এই ধারণাটি ভুল প্রমাণ করতে পেরেছি। আপনি programming এর ভূবনে সম্পূর্ণ নতুন হলেও C++ শিখতে এই বইটি আপনার সহায়ক হবে বলে আমার বিশ্বাস। তবে সময়ের স্বল্পতার কারণে C++ এর কিছু গুরুত্বপূর্ণ topics ও বাদ দিতে হয়েছে। পরবর্তী সংস্করণে সেগুলো যোগ করব। C++ programming এর কিছু কঠিন ও দুর্বোধ্য বিষয়কে বোঝানোর জন্য মাঝে মাঝে কিছু নিজের বানানো গল্পের আশ্রয় নিয়েছি। আশা করি C++ এর এই কঠিন বিষয়গুলো এই গল্পের সাথে মিলিয়ে আপনি অনেক দিন মনে রাখতে পারবেন ও সেগুলো গল্পের মতই সহজ মনে হবে। বইটিতে যাতে কোন রকমের ভুল না থাকে সেজন্য আন্তরিকতার কোন ঘাটতি ছিল না, তারপরও যদি কোন ভুল আপনার চোখে পড়ে তাহলে email করুন:

sattar0703061@yahoo.com এ।

তাছাড়া আপনার যেকোন পরামর্শও যথাযথ গুরুত্বের সাথে বিবেচনা করা হবে এবং পরবর্তী সংস্করণে সেগুলো যোগ করা হবে। অবশেষে পাঠকের প্রতি আমার পরামর্শ হলো:

“বইটিকে আপনি একটি গল্পের বই ভাবুন। মনে করুন আপনি একটি ভাষা শিখছেন সেটি কোন মানুষের ভাষা নয়, কোন সমাজ বা গোষ্ঠীর ভাষা নয়, সেটি সম্পূর্ণ ভিন্ন একটি ভাষা। কখনো পাখির গানের মতো, কখনো নদীর ঢেউয়ের মৃদ শব্দের মতো, আর কখনো কম্পিউটারের ভাষা C++ এর মতো।”

আবদুস ছাত্তার ভূঁইয়া
তড়িত ও ইলেক্ট্রনিক কৌশল(৪র্থ বর্ষ)
খুলনা প্রকৌশল ও প্রযুক্তি বিশ্ববিদ্যালয়।

উৎসর্গঃ

আব্দু-আব্দু কে আর

রবিন, শান্ত এবং নাঈম ভাইয়ার উদাসীন মুহূর্তগুলোকে।

প্রথম অধ্যায়ঃ C++ এর ভূবনে যাবা সম্পূর্ণ নতুন।

“সহজ কথা বলতে আমায় বল যে

সহজ কথা যায়না বলা সহজে”.....রবি ঠাকুর।

আমার মনে হয় C++ এর মত সহজ ও মজার একটা programming language কে তথ্যীয় বর্ণনায় কঠিন করা হয়েছে। তাই এই বইয়ের প্রত্যেকটা topics উদাহরন দিয়ে সহজে ব্যাখ্যা করার চেষ্টা করবো। ছোট

ছোট program লিখে সেগুলো run করে দেখব কিভাবে c++ দিয়ে একটা problem solve করা যায়।

এজন্য প্রথমে আপনার pc তে C++ এর যেকোন version এর compiler (যেমনঃ Turbo C, Borland C++, Microsoft visual studio C++ etc) install করার জন্য অনুরোধ করা হলো।

যারা নতুন তাদের জন্য “Microsoft visual studio C++ “ compiler টা operate করা একটু ঝামেলা হতে পারে। তাই অন্য version এর compiler (যেমনঃ Turbo C) ব্যবহার করুন।

Turbo C compiler ব্যবহার করলে এটা install করতে হবে না।

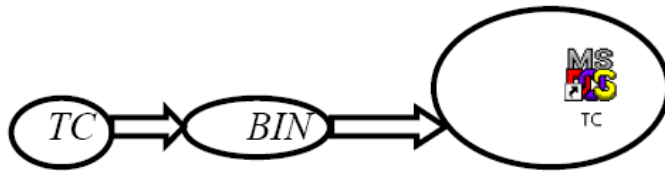
Turbo C compiler টা সাধারণত TC নামে থাকে।


Example 1.1: মনে করুন এমন একটা program লিখতে হবে যেটা run করলে output এ নিচের লেখাটা দেখায়ঃ

Santo is a new programmer.

Solution:

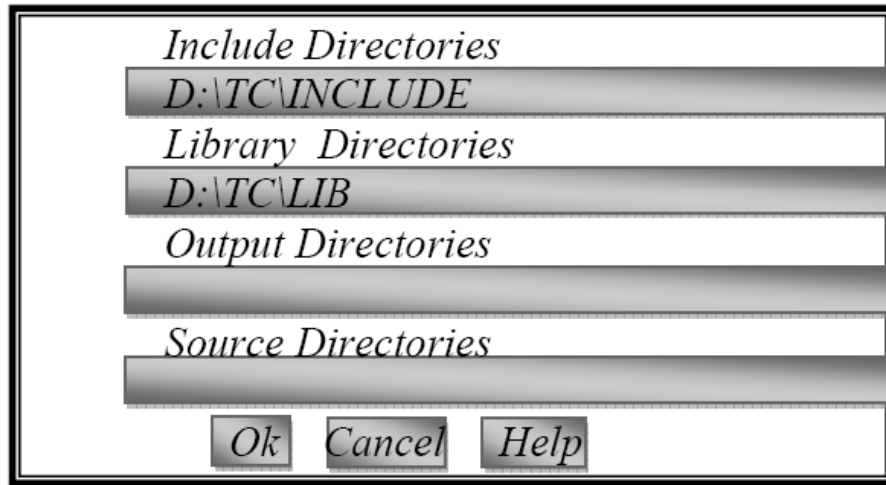
প্রথমে program টা লিখার জন্য একটা নতুন page খুলতে হবে। নিচের প্রবাহ চিত্র অনুসরণ করে একটা নতুন page খুলুন।



এখন  তে *double click* করলে নিচের মত একটা মেনু দেখা যাবেঃ

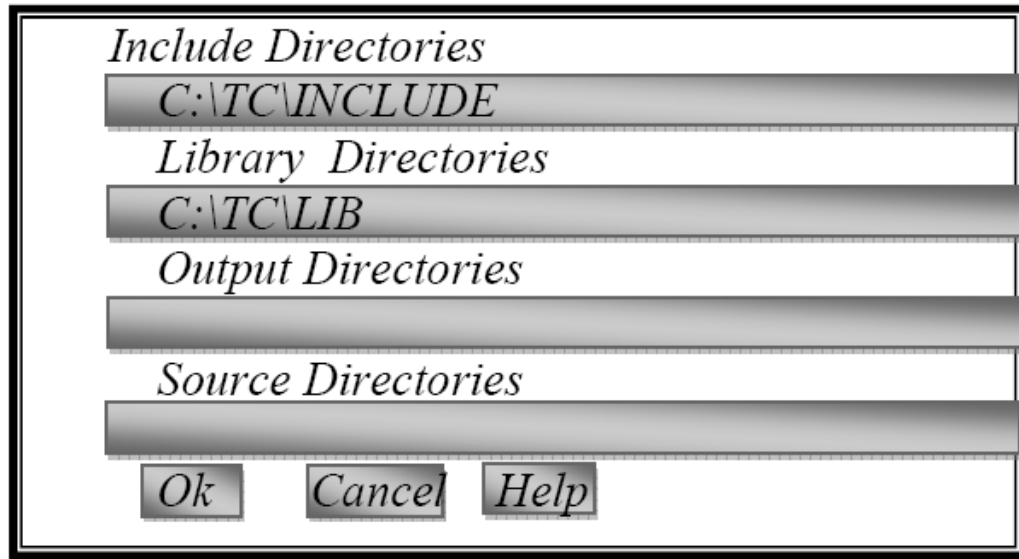
≡File Edit Search Run Compile Debug Project Option Window Help.

উপরের মেনু গুলো থেকে *Option* মেনু টা তে *click* করুন। এখন আপনি একটি বক্সে অনেক গুলো অপশন দেখতে পাবেন। সেগুলো থেকে *Directories...* অপশন টা তে *click* করুন। তাহলে আপনি বড় একটি বক্সের মধ্যে নিচের মত একটা লেখা দেখতে পাবেনঃ



আপনি TC compiler টা আপনার computer এর কোন drive এ রেখেছেন??????

বুঝার সুবিধার জন্য মনে করি যে আপনি D drive এ রেখেছেন। তাহলে উপরের বক্সের মধ্যে কোন পরিবর্তন করতে হবে না। কিন্তু যদি অন্য drive এ রেখে থাকেন তাহলে D এর জায়গায় অন্য drive টি লিখতে হবে। যেমন TC compiler টা যদি C drive এ থাকে তাহলে উপরের বক্সের D এর জায়গায় C হবে। অর্থাৎ বক্সটা কে নিচের মত পরিবর্তন করে OK press করতে হবে।

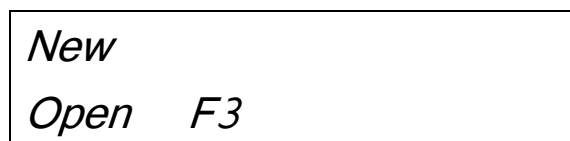


একবার এই setting ঠিক হয়ে গেলে আর ঠিক করতে হবে না। তবে আপনি যদি TC compiler টা অন্য drive এ replace করেন সেক্ষেত্রে করা লাগবে।

এখন আবার নিচের মেনু থেকে File এর উপর click করুন:



File এর উপর click করলে নিচের মত একটা বক্স দেখতে পাবেন:



Save F2
Save as....
Save all
Change dir.....
Print
DOS Shell
Quit Alt+X

উপরের অপশনগুলো থেকে New তে click করলে NONAME00.cpp নামে একটা খালি File Open হতে পারে। NONAME00.cpp এর পরিবর্তে NONAME01.cpp বা NONAME02.cpp বা NONAME03.cpp File Open হতে পারে। তবে যে নামেই File টা Open হোক না কেন File টা যদি খালি না হয়ে পুরাতন কোন program লেখা File Open হয় তবে আপনাকে আবার New Press করে আরেকটি নতুন File Open করতে হবে। এখন সেই File এ নিচের program টা লিখে ফেলুনঃ

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
clrscr();
```

```
cout<<"Santo is a new programmer.";
```

```
getch();
```

```
return 0;
```

```
}
```

≡File Edit Search Run Compile Debug Project Option Window Help.

এখন উপরের Run অপশন থেকে রান করুন অথবা একসাথে Ctrl এবং F9 চাপুন।

Output:

Santo is a new programmer.

উপরের program এর প্রত্যেকটি লাইন এর ব্যাখ্যা:

“iostream.h” এবং

“conio.h” কে বলা হয় header file. এদের কাজ কি তা একটু পরে জানতে পারবেন।

int main():

যেকোন program লিখার জন্য int main() এই function টা লিখতে হয়।

এর পর যেখান থেকে “{ “ শুরু হয় তার পর থেকে program টা এক্সিকিউট করা শুরু করে এবং একটি একটি করে লাইন এক্সিকিউট করার পর যেখানে “}” আছে সেখানে টার্মিনেট করবে। লক্ষ্য রাখতে হবে যে int এবং main এর মাঝে যেন কমপক্ষে একটা space থাকে।

***কোন লাইন এক্সিকিউট করা মানে হলো সেই লাইন এর যে কাজ সেই কাজটি করা।

clrscr(); লিখার কারনে output page এ আগের program এর output কে মুছে নতুন program এর output প্রিন্ট করবে। আর clrscr(); এই function টা কাজ করার জন্য যে command দরকার সেগুলো

“conio.h” এই header file এ দেওয়া আছে।

cout<<”Santo is a new programmer.”; লেখার কারনঃ

কোন message যদি print করতে হয় তাহলে `cout<<` লিখার পর ইনভার্টেট কমার (“ ”) ভিতর সেই message টা লিখতে হয়। এখানে “cout” কে বলা হয় identifier আর “<<” কে বলা হয় insertion to operator.

উপরের program এ `cout<<` লেখার মাধ্যমে output এ message প্রিন্ট করার জন্য যেই command দরকার তা “`iostream.h`” header file এ দেওয়া আছে।

`getch()`; লিখার কারনে output দেখার পর আপনি যতক্ষণ পর্যন্ত keyboard থেকে কোন button press না করেন ততক্ষণ পূর্যন্ত output page টা আপনি দেখতে পাবেন। `getch()`; function এর এই কাজটার জন্য যে command দরকার সেগুলো “`conio.h`” এই header file এ দেওয়া আছে।

`return 0;` এই লাইন এর ব্যখ্যাটা আপনি চতুর্থ অধ্যায় পরলে বুঝতে পারবেন।

Analysis of Example 1.1: এখন `getch()` টা কেটে দিন এবং program টা run করুন। কি কোনো output ই দেখাচ্ছে না?????

≡File Edit Search Run Compile Debug Project Option Window Help.

অপশনগুলো থেকে Window তে click করুন। তাহলে আপনি একটি বক্সে অনেকগুলো অপশনের মাঝে দেখুন “output” নামে একটা অপশন দেখতে পাবেন। এখন “output “ এ click করলে আপনি output দেখতে পাবেন।

****মনে রাখবেন C++ language টা case sensitive . সব কিছু small letter এ লিখতে হবে।

যেমনঃ আপনি `#include<iostream.h>` না লিখে যদি

`INCLUDE<IOSTREAM.H>` লিখেন তাহলে error দেখাবে।

Example 1. 2: এমন একটা program লিখুন যেটা run করলে নিচের লেখাটা output এ দেখায়।

Shanto is a new programmer.

He is really interested to know it deeply.

Solution:

```
#include<iostream.h>
```

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
    clrscr();
```

```
    cout<<"Shanto is a new programmer.\n";
```

```
    cout<<"He is really interested to know it deeply.\n";
```

```
    getch();
```

```
    return 0;
```

```
}
```

Output:

Shanto is a new programmer.

He is really interested to know it deeply.

ব্যাখ্যাঃ

ইনভার্টেট কমার ভিতরে যদি \n (ব্যাক স্পেস n) থাকে তাহলে এর পরের লেখাটা নতুন লাইন থেকে প্রিন্ট করবে।

Analysis : এখন যেখানে \n আছে সেখান থেকে \n কেটে দিয়ে program টা রান করে output টা একটু নিজে নিজে analysis করুন। আবার পরপর দুই বার ব্যাক স্লেশ n (\n) দিয়ে রান করুন। আবার \n কেটে দিয়ে \t দিলে একটা ট্যাব এর সমপরিমাণ space ফাঁকা রেখে message প্রিন্ট করবে।

দ্বিতীয় অধ্যায়ঃ Tokens , Expression & comment.

Token & Expression : আপনাকে যদি প্রশ্ন করি যে বাক্য কাকে বলে? তাহলে আপনি নিশ্চই বলবেন—

“কত গুলো শব্দ পাশাপাশি বসে যদি মনের ভাব প্রকাশ করে তাহলে তাকে বাক্য বলে।

এখন যদি আবার প্রশ্ন করি যে শব্দ কাকে বলে? তাহলে আপনার উত্তরটা নিশ্চই হবে—

“কত গুলো বর্ণ পাশাপাশি বসে যদি কোন অর্থ প্রকাশ করে তাহলে তাকে শব্দ বলে “।

এখন যদি বলি বর্ণ কাকে বলে? তাহলে আপনার উত্তর হবে:

ভাষায় ব্যবহৃত সবচেয়ে ক্ষুদ্রতম একক কে বর্ণ বলে।

C++ ও যেহেতু একটা ভাষা এই ভাষারও একটা ক্ষুদ্রতম একক আছে।

C++ ভাষার ক্ষুদ্রতম একক কে বলা হয় Token.

****token কে আপনি বর্ণের সাথে তুলনা করতে পারেন।*

আর কতগুলো Token মিলে যখন program এর একটা একক কাজ (unit task) সম্পন্ন করবে তখন সেই Token গুলোকে একসাথে একটা statement

বলা হবে। এক একটা statement একটা শব্দের সাথে তুলনা করতে

পারেন। কতগুলো শব্দ মিলে যেমন বাক্য হয় তেমনি কতগুলো

statement নিয়ে একটা program হয়। বাক্য ছাড়া যেমন মনের ভাব

প্রকাশ করা যায় না তেমনি statement ছাড়াও কোন program লিখা যায়

না। অর্থাৎ একটা program হলো কতগুলো statement এর

সমষ্টি। কোন statement এ যখন mathematical term থাকবে তখন

তাকে statement না বলে Expression বলা হয়। অর্থাৎ সব Expression

ই statement কিন্তু সব statement ই Expression না। program এ

প্রত্যেকটা statement এর পরে একটা সেমিকোলন(;) থাকে। তাই আপনি

যদি program এর কোন জায়গায় ভুলে সেমিকোলন(;) না দেন তাহলে রান করার পর আপনাকে “statement missing” নামে একটা error দেখাবে।

যেমনঃ `cout<<”santo is not quiet in nature”;` একটা statement.
`2+3;` এটাও একটা statement কিন্তু mathematical statement তাই একে statement না বলে Expression বলতে হবে।

Example :2.1

```
#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr(); →statement-1
    cout<<”Robin is really genoius.”; →statement-2
    getch(); →statement-3
    return 0; →statement-4
}
```

Comment: উপরের program টা রান করলে error দেখাবে। কারণ .

→statement-1

→statement-2

→statement-4

→statement-3

এগুলো এই program এর কোন Token না। কিন্তু program এর coding এর ব্যাখ্যা করার জন্য coding এর পাশে যখন এ ধরনের information লেখার দরকার হয় তখন information টার আগে “//” বসিয়ে দিতে হবে।

“// information ” কে comment বলা হয়।

এখন উপরের program টা নিচের মত লিখলে রান করবে।

```

#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr(); //→statement-1
    cout<<"Robin is really genoius."; // →statement-2
    getch(); //→statement-3
    return 0; // →statement-4
}

```

আবার *Comment* যদি একসাথে কয়েক লাইন এর হয় তাহলে বার বার *//* লিখার পরিবর্তে */** এবং **/* এর মাঝখানে *Comment* টা লিখা হয়।
যেমন: নিচের program টা তে মূল প্রগ্রাম এর সাথে উপরের দুটি লাইন কে */** এবং **/* এর মাঝখানে লেখার কারনে compiler এই লেখাটা কে *inactive* হিসাবে ধরবে।

```

/* this is a very simple program
   to illustrate the use of “/*” and “*/”
*/
#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr(); //→statement-1
    cout<<"Robin is really genoius."; // →statement-2
    getch(); //→statement-3
    return 0; // →statement-4
}

```

লক্ষ্য করুন *“//”* এর পরবর্তী লাইন টা program এ *inactive* করে দিবে।

আর “/*” এবং “*/” এর মাঝখানের সবগুলো লাইন কে program এ inactive করে দিবে।

Comment কেন লেখা হয়????????

উত্তরঃ অনেক দিন পরে আপনি নিজেই আপনার লেখা program এর Coding বুঝবেন না। তাই Coding এর পাশে Comment লিখে রাখলে বুঝতে সুবিধা হবে।

অথবা অন্য কেউ আপনার program দেখলে Coding এর পাশে যদি Comment ব্যবহার করলে বুঝতে সুবিধা হবে।

তৃতীয় অধ্যায়ঃ Data type & variable.

Data type & Variable: নিচের table টি লক্ষ্য করুনঃ

Name	Beginning letter of the name	Age (year)	Hight (ft)	Salary (TK)
1.Shanto	S	24	5.6	40,000/-
2.Robin	R	23	5.7	35,000/-
3.Rashed	R	22	5.4	32,000/-
4.Masud	M	22	5.6	30,000/-
5.Kamruzzaman	K	23	5.7	30,000/-
6.Kawser	K	24	5.8	30,000/-
7.Abdus Sattar	A	24	5.5	30,000/-

উপরের লক্ষ্য করুন “Shanto” কে আপনি “word” type এর data বলতে পারেন।

একই ভাবে 24(শান্ত'র বয়স) কে আপনি integer(পূর্ণ সংখ্যা) type এর Data এবং 5.6 (শান্ত'র উচ্চতা)কে float(দশমিক) type এর Data বলতে পারেন। আর S,R,M,K,A কে character (বর্ণ) type এর Data বলতে পারেন।

*** সহজ কথায় Data type মানে হলো কোন ধরনের বা কোন Type এর Data.

C++ এ data type কেন দরকার??????

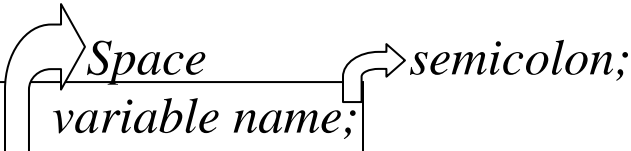
উত্তরঃ মনে করুন C++ program এর মাধ্যমে দুটি পূর্ণ (integer) সংখ্যা যোগ করতে হবে। তাহলে আপনাকে ধাপে ধাপে কাজটি করতে হবে।

১ম ধাপঃ আপনাকে keyboard থেকে দুটি সংখ্যা input দিতে হবে।

২য় ধাপঃ সংখ্যা দুটি কে computer এর একটি নির্দিষ্ট address এ রাখতে হবে। এখন সংখ্যা দুটি রাখার জন্য আপনি computer এ নতুন address কিভাবে সৃষ্টি করবেন????

এ কাজটি করার জন্যই Data type ডিক্লেয়ার(প্রচার) করা হয়।

Data type ডিক্লেয়ার করার নিয়মঃ

Data type name 

Data type name হলো একটি keyword. অর্থাৎ C++ এ সংরক্ষিত কিছু সাংকেতিক শব্দ। যেমন Data যদি integer(পূর্ণ সংখ্যা) টাইপ এর হয় তাহলে

Data type name হবে int

এক্ষেত্রে আপনি int কে অন্য কোনভাবে (যেমনঃ Int) লিখতে পারবেন না।

কিন্তু variable name আপনি আপনার ইচ্ছামত লিখতে পারবেন।
যেমনঃ

int santo;

তবে কিছু শর্ত আপনাকে মানতে হবেঃ

১. Variable name এর শুরুতে কোন digit থাকতে পারবেনা।

যেমনঃ int 1santo; লেখা যাবেনা।

২. একটা keyword কে আপনি variable name হিসাবে লিখতে পারবেন না।

int float; লিখতে পারবেন না। কারণ float হলো একটা keyword

৩. ডলার সাইন (\$) ব্যবহার করতে পারবেন না।

৪.

৫.

.

কোন কোন বইয়ে Variable ডিক্লেয়ার করতে এরকম প্রায় ১০ টি শর্তের কথা উল্লেখ করেছে। শুধু ১ টি variable ডিক্লেয়ার করতে এই ১০ টি শর্ত মুখস্থ করতে হবে!!!!!!

Short cut technique:

Variable name লিখতে শুধু নিচের ৪ শর্ত মনে রাখুন:

১. a হতে z পর্যন্ত বা A হতে Z পর্যন্ত যেকোন বর্ণ ব্যবহার করতে পারেন।
২. Digit ব্যবহার করতে পারেন (তবে শুরুতে না)।
৩. Underscore ব্যবহার করতে পারেন।
৪. একটা keyword কে আপনি variable name হিসাবে লিখতে পারবেন না।

প্রশ্ন: `int robin+;` লিখা যাবে???

উত্তর: না। কারণ '+' ব্যবহার করা হয়েছে। নিচের মত ডিক্লেয়ার করতে হবে:

```
int robin;
```

প্রশ্ন: `float santo robin;` লিখা যাবে???

উত্তর: না। কারণ space ব্যবহার করা হয়েছে। নিচের মত ডিক্লেয়ার করতে হবে:

```
float santo_robin;
```



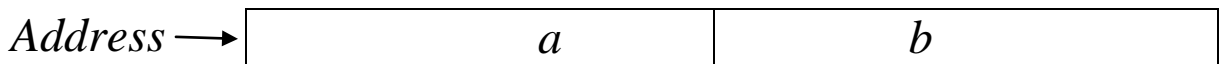
Variable ডিক্লেয়ার করলে কি হয়???

যেমন: আপনি নিচের মত `integer` (পূর্ণ সংখ্যা) ডিক্লেয়ার করতে পারেন।

```
int a;
```

```
int b;
```

উপরের দুটি লাইন লেখার কারণে computer এ নতুন দুটি address সৃষ্টি হবে যা নিচের চিত্রে দেখানো হলো:



এখানে *a* ও *b* কে বলা হয় variable. অর্থাৎ variable হচ্ছে memory address এর নাম যেখানে কোন information সংরক্ষণ করে রাখা হয়। memory address এর নাম কে কেন variable বলা হয় তা একটু পরেই বুঝতে পারবেন।

এখন দেখা যাক কিভাবে keyboard থেকে দুটি পূর্ণ সংখ্যাকে এই দুটি address এ রাখবেন। ইতিমধ্যেই আপনি জানেন যে

“`cout<<`” লেখার পর ইনভার্টেট কমার ভেতর যে message টা লেখা হয় সেটা output screen এ

দেখায়। একই ভাবে “`cin>>`” লেখার পর কোন variable এর নাম লেখলে

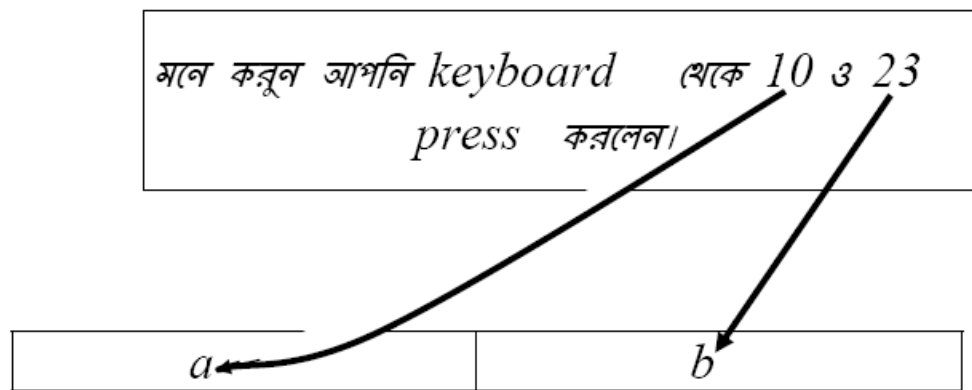
Keyboard থেকে input দেয়া যায়।

যেমন: আপনি যদি লেখেন:

```
cin>>a>>b;
```

তাহলে program রান করার পর keyboard থেকে দুটি মান input দিলে

যথাক্রমে a ও b address এ রাখবে। variable এ কোন মান রাখাকে বলা হয় Initialization of variable. নিচের চিত্রটি লক্ষ্য করুন:



Keyboard থেকে মান না দিয়ে মূল program এ variable initialize করা যায়। যেমনঃ `cin>>a>>b;` না লিখে `a=10;b=23;` লিখলেও হবে।

এখন মনে করুন a ও b কে যোগ করতে হবে। তাহলে $a+b$ লিখলেই যোগ হয়ে যাবে। কিন্তু যোগফলটা কে তো নতুন একটা address এ রাখতে হবে। তাই

`int sum;` লিখে নতুন একটি address সৃষ্টি করে নিন।

এখন যদি লিখেন

`sum=a+b;`

তাহলে যোগফলকে `sum` address এ রাখবে। এখানে লক্ষ্য করুন “=” (সমান সমান) চিহ্ন কে C++ এর ভাষায় assignment operator বলা হয়। assignment operator মানে হলো যেটা assign করে। assign করা মানে হলো কোন কিছু আরোপ করা

আরো সহজ ভাবে বলা যায় যে কারো উপর কোন কিছু চাপিয়ে দেওয়া।

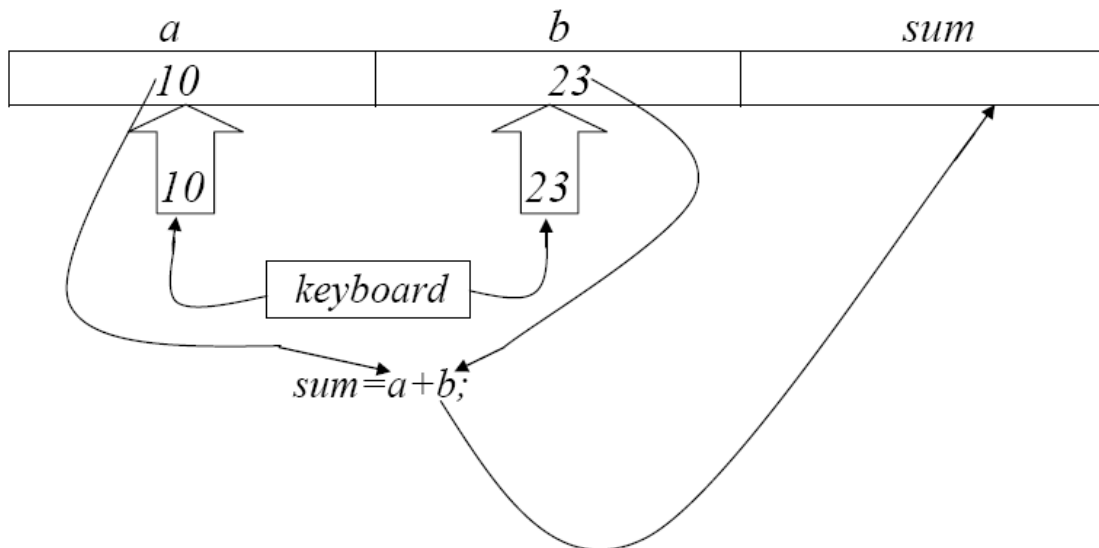
Assignment operator (“=”) এর ডান পাশের address কে বাম পাশের address এ চাপিয়ে দেওয়া হয়।

“ $sum=a+b;$ ” লেখার কারনে কি হয় তা একটু analysis করা যাক:

১ম ধাপ: $a+b;$ নির্ণয় করার জন্য a ও b এর address এ যে value রাখা হয়েছে সেগুলো কম্পিউটার এর memory থেকে call করে নিয়ে আসবে।।

২য় ধাপ: এখন যোগফল কে sum address এ রাখবে।

সমগ্র প্রক্রিয়াটা আরও ভালো ভাবে বুঝার জন্য নিচের চিত্রটা লক্ষ্য করুন:



*** আপনি কি উপরের চিত্রটি দেখে কিছু বুঝছেন???? না বুঝলে হতাশ হবেন না। পরবর্তীতে আরও কয়েকটি উদাহরণ দেখলে সহজে বুঝতে পারবেন।

এখন আপনি যদি যোগফল দেখতে চান তাহলে তাহলে যোগফল যে variable এ রেখেছেন `cout<< লিখার পর সেই variable টা লিখলেই হবে। অর্থাৎ`
`cout<<sum;` লিখলেই হবে।

এখন আপনার প্রশ্ন হতে পারে `cout<<sum;` আর `cout<<"sum";` মধ্যে পার্থক্য কি???

উত্তরঃ `cout<< লিখার পর` আপনি যদি ইনভার্টেট কমার(“”) ভিতরে কিছু লিখেন তাহলে Output স্ক্রিনে ইনভার্টেট কমার ভেতরের লেখাটা দেখাবে। আর `cout<< লিখার পর` আপনি যদি ইনভার্টেট কমা না দিয়ে কোন কিছু লিখেন তাহলে compiler সেটাকে একটা memory address এর নাম হিসাবে বিবেচনা করবে এবং সেই memory address এ যে মান রাখা আছে সেই মানটা output স্ক্রিনে দেখাবে।

অর্থাৎ `cout<<sum;` লেখার কারণে a ও b এর যোগফল কে output এ দেখাবে কেননা a ও b এর যোগফল কে sum নামক variable এর address এ রাখা হয়েছে।

আর `cout<<"sum";` লেখার কারনে `sum` লেখাটা `output` এ দেখাবে।

*** `int a;`

`int b;` এভাবে দুটি `variable` কে আলাদাভাবে ডিক্লেয়ার না করে নিচের মত ও ডিক্লেয়ার করা যায়ঃ

`int a,b;`

অর্থাৎ

data type space variable_1, variable_2, variable_3, variable_n; semicolon

কমা

এখন আপনি জানেন কিভাবে `variable` ডিক্লেয়ার করতে হয়। এখন আপনার প্রশ্ন হতে পারে `program` এর কোথায় `variable` ডিক্লেয়ার করতে হয়??

উত্তরঃ `variable` ডিক্লেয়ার করলে `memory address` সৃষ্টি হয় এবং সেই `memory address` এ ডাটা ইনপুট দেওয়া হয়। আর যেহেতু আগে `memory address` সৃষ্টি করতে হবে তার পর সেই `memory` তে ইনপুট দিতে হবে। তাই ইনপুট দেওয়ায় আগে `variable` ডিক্লেয়ার করে `memory address` সৃষ্টি করে নিতে হবে।

উদাহরনঃ

```
#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr();
    cout<<"Enter a character : ";
    cin>>santo; // input from keyboard to the created memory
    char    santo; // memory is created
    getch();
    return 0;
}
```

এই program এ compiler প্রথমে cin>>santo; এবং পরে char santo; statement টা পাবে।

cin>>santo; —→ এই statement পাওয়ার পর compiler বুঝবে keyboard থেকে

কোন বর্ন ইনপুট দিতে হবে এবং সেটা santo নামক memory address এ রাখতে হবে। কিন্তু santo নামক memory address

সৃষ্টি করা হয়নি। তাই compiler নিচের মত একটা error message

দেখাবেঃ
undefined symbol 'santo'

char santo; —→ এই statement পাওয়ার পর compiler বুঝবে santo নামে একটা

memory address সৃষ্টি করতে হবে যেই memory address এ

character type এর ডাটা অর্থাৎ বর্ন ইনপুট দেওয়া যাবে। এই statement টা cin>>santo statement এর আগে লিখলে

উপরের

error টা দেখাবেনা।

Program টা নিচের মত যেকোন একভাবে লিখলে আর error টা দেখাবে না :

```
#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr();
    char    santo;    // memory is created
    cout<<"Enter a character : ";
    cin>>santo;    // input from keyboard to the created memory
    getch();
    return 0;
}
```

অথবাঃ

```
#include<iostream.h>
#include<conio.h>
int main()
{
    char    santo; //memory is created
    clrscr();
    cout<<"Enter a character : ";
    cin>>santo; // input to a created memory.
    getch();
    return 0;
}
```

অথবাঃ

```
#include<iostream.h>
#include<conio.h>
int main()
{
```

```

        clrscr();
        cout<<"Enter a character : ";
        char    santo; //memory is created
        cin>>santo; //input to a created memory
        getch();
        return 0;
}

```

***অর্থাৎ variable যেখানেই ডিক্লেয়ার করেন না কেন ইনপুট দেওয়ার আগে ডিক্লেয়ার করলেই হবে।

Example2.2: এমন একটা program লিখুন যেটা দুটি সংখ্যা যোগ করে যোগফল কে output এ দেখাবে।

Solution:

```

#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr();
    int a,b;
    cout<<"Enter two integer values : ";
    cin>>a>>b; // assigning two values from keyboard.
    int sum;
    sum=a+b;
    cout<<"sum of "<<a<<" & "<<b<<" = "<<sum<<"\n";
    getch();
    return 0;
}

```

Output:

```

Enter two integer values : 25 45
sum of 25 & 45 = 70

```

Analysis:

clrscr(); লেখার কারনেঃ

আগের program এর Output মুছে যাবে।

int a,b; লেখার কারনেঃ

a ও b variable এর জন্য নিচের মত memory তে দুটি ঘর সৃষ্টি হবে।

a

b

--	--

এখন a ও b এর ঘরে আপনি দুটি পূর্ণ সংখ্যা রাখতে পারবেন।

`cout<<"Enter two integer values : ";` লেখার কারণে:

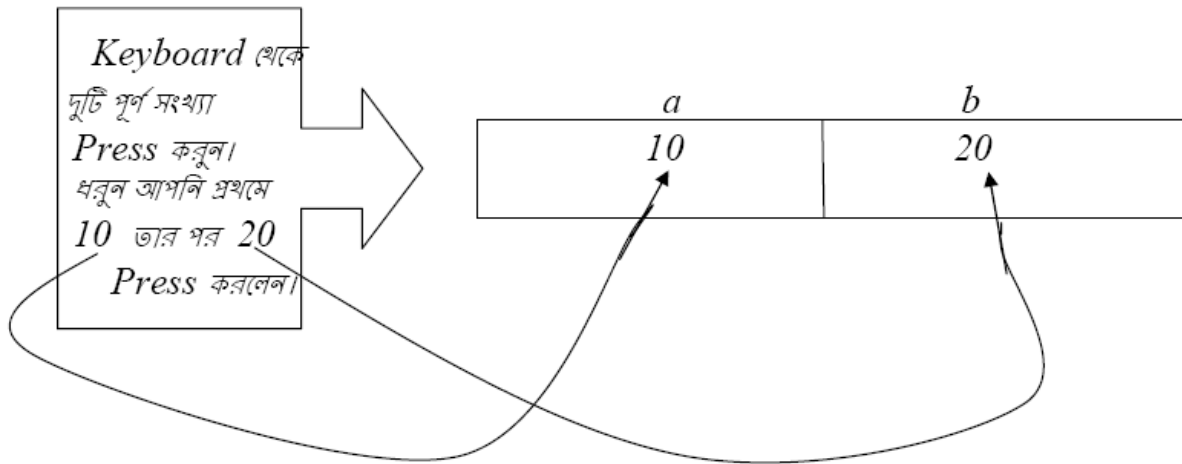
Output স্ক্রিনে নিচের লেখাটা দেখাবে:

Enter two integer values :

`cin>>a>>b;` লেখার কারণে:

keyboard থেকে দুটি পূর্ণ সংখ্যা ইনপুট দিলে সেগুলো যথাক্রমে a ও b এর জন্য উপরে যে দুটি খালি ঘর সৃষ্টি হয়েছে সেই ঘর দুটিতে রাখবে।

এটা বুঝার জন্য নিচের চিত্রটা লক্ষ্য করুন:



`int sum;` লেখার কারনেঃ

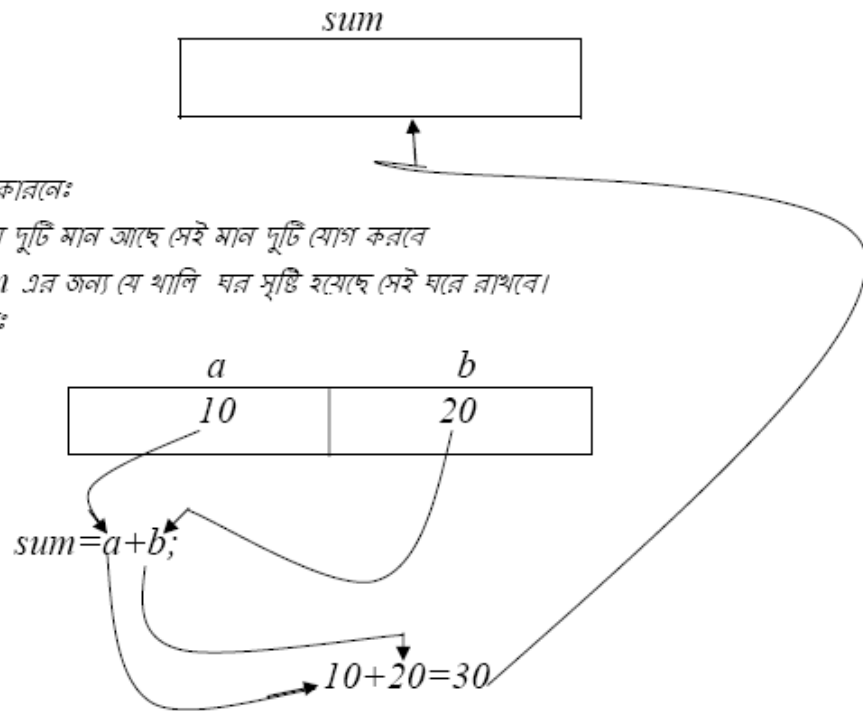
`sum` variable এর জন্য `memory` তে একটি খালি ঘর সৃষ্টি হবে। যা নিচে দেখানো হলো।

`sum = a + b;` লেখার কারনেঃ

প্রথমে `a` ও `b` এর ঘরে যে দুটি মান আছে সেই মান দুটি যোগ করবে

তারপর যোগফল কে `sum` এর জন্য যে খালি ঘর সৃষ্টি হয়েছে সেই ঘরে রাখবে।

নিচের চিত্র টা লক্ষ্য করুনঃ



`cout << "sum of " << a << " & " << b << " = " << sum << "\n";` লেখার কারনেঃ

Output স্ক্রিনে

sum of 10 & b = 30

লেখাটা Output এ দেখাবে।

*** লক্ষ্য রাখবেন `int` ও `a` এর মাঝখানে যেন কমপক্ষে একটি `space` থাকে।

অর্থাৎ `data type` ও `variable name` এর মাঝখানে কমপক্ষে একটি `space` থাকবে।

`data type` এর পরে যেটা লেখা হয় তাকে `variable` বলা হয় কেন???????

মনে করুন আপনি লিখলেনঃ

```
int a,b;
```

```
cin>>a>>b;
```

```
cin>>a>>b;
```

তাহলে কি হবে????????

প্রথমে keyboard থেকে দুটি মান নিয়ে `a` ও `b` এর `address` এ রাখবে তারপর আবার keyboard থেকে দুটি মান নিবে এবং `a` ও `b` আর আগের মান মুছে

দিয়ে নতুন মান a ও b এর address এ রাখবে। এইভাবে যতবার মান পরিবর্তন করা হয় ততবারই নতুন মান assign হয়।
যেহেতু a ও b এর মান সর্বদাই vary করে তাই এদের variable বলা হয়।
এই ব্যাপারটা বুঝার জন্য নিচের program টা রান করে দেখুনঃ

Example 2.3 :

```
#include<iostream.h>
#include<conio.h>
int main()
{
    clrscr();
    int a,b;
    cout<<"Enter value of a & b : ";
    cin>>a>>b;
    cout<<"Enter value of a & b again : ";
    cin>>a>>b;
    cout<<"\n\nNow value of a & b = "<<a<<" & "<<b<<"\n";
    getch();
    return 0;
}
```

Sample Output:

```
Enter value of a & b : 1 2
Enter value of a & b again : 4 5
```

```
Now value of a & b = 4 & 5
```

Analysis: output এ a ও b এর যে মান দুটি দেখাচ্ছে সেগুলো হলো দ্বিতীয় বার আপনি keyboard থেকে যে মান গুলো input দিয়েছেন সেগুলো। তাহলে প্রথম বার input দেওয়া মানগুলো কোথায় হারিয়ে গেলো????? ।

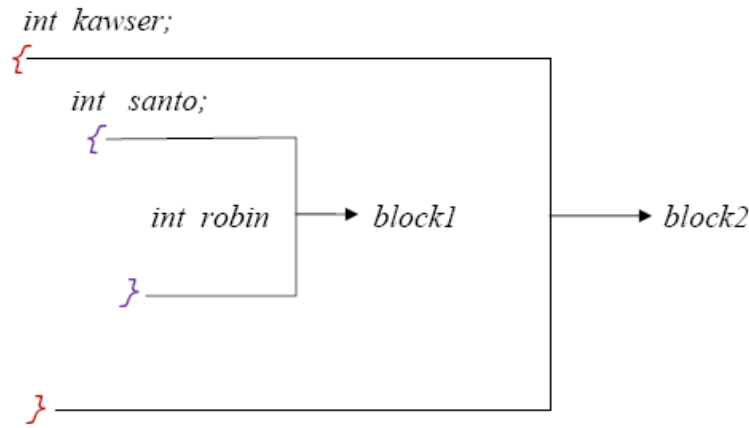
উত্তরঃ প্রথম বার input দেওয়া মানগুলো মুছে দিয়েছে।variable কে পেন্সিলে আঁকা কোন ছবি ভাবুন যা যখন খুশি তখন মুছে ফেলে একই জায়গায় আবার নতুন ছবি আঁকা যায়।

C++ এ data type কিভাবে লিখতে হয়???:

এতক্ষণ আপনি শুধু জেনেছেন কিভাবে integer type এর data নিয়ে কাজ করা যায়। এখন আপনি যদি অন্য data type নিয়ে কাজ করতে চান তাহলে একই ভাবে আপনাকে data type আগে ডিক্লেয়ার করতে হবে। তবে যে data type ই আপনি ডিক্লেয়ার করেন না কেন আপনাকে অবশ্যই সেই data type এর keyword জানতে হবে। যেমনঃ integer(পূর্ণ সংখ্যা) data ডিক্লেয়ার করতে আপনাকে “int” keyword টা ব্যবহার করতে হয়েছিল। আপনি যদি “int” না লিখে “integer” লিখেন তাহলে কিন্তু error দেখাবে। কারণ C++ compiler একমাত্র “int” কে ই integer হিসাবে বিবেচনা করবে। যেমন : আপনি যদি “ক্যালসিয়াম কার্বনেট” এর সংকেত লিখতে চান তাহলে আপনাকে অবশ্যই “CaCO₃” লিখতে হবে। CaCO₃ বা Caco3 অথবা অন্য কোন ভাবেই লিখা যাবে না। chemistry শিখতে হলে যেমন এসব সংকেত মুখস্থ করতে হয়। তবে C++ শিখতে হলে data type মুখস্থ করতে হবে না একটু মাথায় রাখতে হবে। তবে চিন্তার কারণ নেই কারণ C++ এ যেসব basic data type বেশি ব্যবহৃত হয় তার সংখ্যা খুবই কম। তাছাড়া আপনি যত program ই করবেন সব Program এ এসব basic data type ব্যবহার করবেন। কয়েকদিন program করার পর এমনিতেই সব data type মুখস্থ হয়ে যাবে।

Local ও global variable বুঝানোর জন্য একটি মজার উদাহরণঃ

ইতিমধ্যেই আপনি জানেন ‘{ }’ কে একটা block বলা হয়। একটা program একটি বা একের অধিক block থেকে পারে। যেমনঃ



block2 এর মধ্যেই block1 লুকিয়ে আছে। block1 কে বাংলাদেশ, block2 কে এশিয়া মহাদেশ আর block2 এর বাহিরের অংশটাকে সমগ্র বিশ্ব চিন্তা করুন। এখন আপনি যদি বাংলাদেশের নাগরিকত্ব অর্জন করেন, তাহলে কিন্তু বাংলাদেশ ছাড়া এশিয়া মহাদেশের অন্য কোন দেশের নাগরিক সুবিধা পাবেন না। কিন্তু আপনি যদি এশিয়া মহাদেশের নাগরিকত্ব অর্জন করেন তাহলে এশিয়া মহাদেশের সব কয়টি দেশের নাগরিক সুবিধা আপনি পাবেন। আর আপনি যদি সমগ্র বিশ্বের নাগরিকত্ব অর্জন করেন তাহলে সারা বিশ্বের যেকোন দেশের নাগরিক সুবিধা পাবেন।

এখন উপরের চিত্র থেকে আমরা বলতে পারি , ‘kawser’ হলো একটা int টাইপের variable যেটাকে globally ডিক্লেয়ার করা হয়েছে , ‘santo’ হলো আরেকটা int টাইপের variable যেটাকে local in block2 হিসাবে ডিক্লেয়ার করা হয়েছে এবং ‘robin’ হলো আরেকটা int টাইপের variable যেটাকে local in block1 হিসাবে ডিক্লেয়ার করা হয়েছে। তাই ‘kawser’ variable কে program এর যেকোন জায়গায় ব্যবহার করা যাবে। ‘santo’ variable কে block2 তে ব্যবহার করা যাবে এবং block1 এ ব্যবহার করা যাবে। ‘robin’ variable কে শুধু block1 এ ব্যবহার করা যাবে।

N.B: বাস্তবে এশিয়া মহাদেশের নাগরিকত্ব বলতে কিছু নাই। উপরের উদাহরণটা শুধু global variable ও local variable কি সেটা বুঝানোর জন্য।

:

চতুর্থ অধ্যায়ঃ Function

মনে করুন আপনি দুটি সংখ্যা যোগ করার একটি program লিখবেন। তাহলে নিচের মত program টা লিখুনঃ

```
#include<iostream.h>
#include<conio.h>
    clrscr();
    cout<<"Enter two value :";
    int a,b;
    cin>>a>>b;
    int sum;
    sum=a+b;
    getch();
    return 0;
```

এখন program টা রান করলে error দেখাবে। কিন্তু দুটি সংখ্যা যোগ করার জন্য যতগুলো statement লেখা দরকার আপনি সব statement ই তো উপরের program এ লিখেছেন।

তারপরও কেন error দেখায়? মনে করুন একটা টেবিলের উপর কিছু পরীক্ষার খাতা এবং একটা চিরকুট পড়ে আছে। চিরকুটে লিখা আছে শান্তর বার্ষিক পরীক্ষার ফলাফল হলোঃ

বিষয়	প্রাপ্ত নম্বর
বাংলা	৯৫
ইংরেজী	৮৫

গণিত	৭৯
সমাজ	৬৫
ইসলাম শিক্ষা	৮৯

শান্তর মোট নম্বর = ৯৫+৮৫+৭৯+৬৫+৮৫

এখন যদি এই চিরকুট এবং খাতাগুলো কখনো কোন শিক্ষকের চোখে না পড়ে তাহলে কিন্তু শান্তর বার্ষিক পরীক্ষার ফলাফল প্রকাশিত হবে না।

লক্ষ্য করুন চিরকুটে সবকিছু থাকার পরও কোন শিক্ষক সেটা না দেখার কারনে বিবেচনা করা হবে:

শান্ত বার্ষিক পরীক্ষায় অংশগ্রহন করেনি। আসলে এটি ভুল(error)।

একইভাবে উপরের program এ সব কিছু থাকার পরও int main() না থাকার কারনে error দেখায়। এখানে int main() কে আপনি শান্তর শিক্ষকের সাথে তুলনা করতে পারেন।

int main() কে programএর ভাষায় ফাংশন বলা হয়।

এখন যদি বলি উপরের উদাহরনে শান্তর শিক্ষকের কাজ কি?

উত্তর হবে পরীক্ষার ফলাফল তৈরি করে প্রকাশ করা।

এখন যদি বলি program এ ফাংশন এর কাজ কি?

উত্তরঃ যেহেতু program হলো কতগুলো statement এর সমষ্টি।তাই program এ ফাংশনের কাজ হলো কতগুলো statement এক্সিকিউট করে ফলাফল তৈরি করা এবং

ফলাফল প্রকাশ করা।

***কোন লাইন বা statement এক্সিকিউট করা মানে হলো সেই লাইন বা statement এর যে কাজ সেই কাজটি করা।

ফাংশন দুই প্রকারঃ

1. *Userdefined function*
2. *Library function*

Userdefined function হলো *main* ফাংশনের বাইরে *programmer* এর তৈরি করা

ছোট ছোট ফাংশন বা *sub function*.

Userdefined function এর উপকারিতা হলো অনেক বড় একটা *program* কে ছোট ছোট অংশে ভাগ করে। আপনার নিজের লেখা অনেক বড় একটা *program* এর *coding* অনেক দিন পর দেখলে আপনি নিজেই বুঝবেন না। তাই *userdefined* ফাংশনের মাধ্যমে ছোট ছোট অংশে ভাগ করে নিলে পরবর্তীতে বুঝতে সুবিধা হয় এবং সহজে *error* বের করা যায়।



C++ এর *Library function* হলো *C++ compiler* এ সংরক্ষিত কিছু ফাংশন।

যেমনঃ *getch()* ফাংশনের যে কাজ সেই কাজটা এক্সিকিউট করার জন্য *programmer* কে *getch()* ফাংশনের *definition* লিখা লাগে না। শুধু মাত্র *getch();* লিখলেই *C++ compiler* এ সংরক্ষিত ফাংশনটা *call* হয়ে যায়।

যেকোন ফাংশনে চারটা অংশ থাকেঃ

1. *return type*
2. *Function name*
3. *Argument*
4. *Function body*

return type Space semicolon হবে না
function name (argument)
{
 Function body (combination of statement)
}

return type  *Space* *function name (argument)*  *semicolon হবে*

এভাবে লেখাটাকে বলা হয় *function prototyping* . অর্থাৎ *return type* এবং *argument* এর সংখ্যা উল্লেখ্য করে ফাংশন ডিক্লেয়ার করাকেই *function prototyping* বলে

*** যেকোন program এ একটি মাত্র int main() ফাংশন থাকবে। এই function এর নাম main() এর পরিবর্তে অন্য কিছু লিখতে পারবেন না।

একটি *program* এ *int main()* ফাংশন বাদে যত ওলো ফাংশন লিখবেন সব ওলো কে বলা হয় *user defined* ফাংশন। *compiler* সবসময় *int main()* ফাংশন হতে এক্সিকিউট করা শুরু করে।

এখন ফাংশনের চারটা অংশ ব্যাখ্যা করা যাকঃ

Function name:

Variable name যেভাবে লিখতে হয় *userdefined function name* ও একই ভাবে লিখতে হয়। অর্থাৎ *userdefined function name* লিখতে আপনি-

১. *a* হতে *z* পর্যন্ত বা *A* হতে *Z* পর্যন্ত যেকোন বর্ণ ব্যবহার করতে পারেন।

২. *Digit* ব্যবহার করতে পারেন (তবে শুরুতে না)।

৩. *Underscore* ব্যবহার করতে পারেন।

৪. একটা *keyword* কে আপনি *variable name* হিসাবে লিখতে পারবেন না।

Argument : একটা ফাংশনের মাধ্যমে আপনি যে কাজটি করতে চান সেই কাজটা করার জন্য আপনাকে ঐ ফাংশনে যে ডাটা পাঠাতে হয় সেই ডাটা কে *argument* বলে।

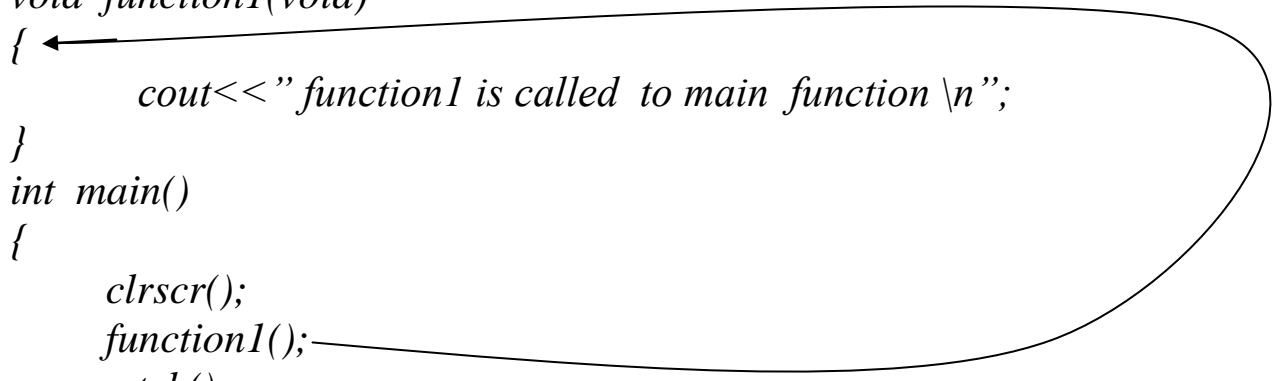
আপনি যদি *int* টাইপ এর ডাটা পাঠান তাহলে *argument* হবে *int* টাইপ। আপনি যদি *float* টাইপ এর ডাটা পাঠান তাহলে *argument* হবে *float* টাইপ। আর আপনি যদি কোন ডাটাই ফাংশনে না পাঠান তাহলে *argument* হবে *void* টাইপ।

return type একটা ফাংশনের সব *statement* এক্সিকিউট করার পরে ফাংশনটা যে টাইপ এর ডাটা *return* করে বা ফেরত দেয় তাকে বলে

return type / যেমন: যদি একটি ফাংশনের কিছু statement দুটি পূর্ণ সংখ্যাকে ভাগ করে ভাগফল হিসাবে কোন ভগ্নাংশ টাইপ এর ডাটা return করে তাহলে return type হবে float. কোন কিছু return না করলে return টাইপ হবে void.

return type ,argument এবং function body সম্পর্কে ভালভাবে বুঝার জন্য নিচের উদাহরণগুলো দেখুন:

```
#include<iostream.h>
#include<conio.h>
void function1(void)
{
    cout<<"function1 is called to main function \n";
}
int main()
{
    clrscr();
    function1();
    getch();
    return 0;
}
```



The diagram consists of two curved arrows. The first arrow starts from the `function1();` line in the `main()` function and points to the opening curly brace of the `function1(void)` function definition. The second arrow starts from the closing curly brace of the `function1(void)` function and points back to the `function1();` line in the `main()` function, illustrating the call and return process.

Output:

function1 is called to main function

উপরের program এ compiler প্রথমে int main() ফাংশনের ভিতরে চুকবে। তারপর clrscr(); statement টা এক্সিকিউট করবে। এরপর যখন function1(); statement টা পাবে তখন compiler টা int main() থেকে বের হয়ে “void function1(void)” ফাংশনের ভেতরে চুকবে। int main() ফাংশনে function1(); statement টা লেখার অর্থ হলো

“ void function1(void) ” কে int main() ফাংশনে call করা হয়েছে।

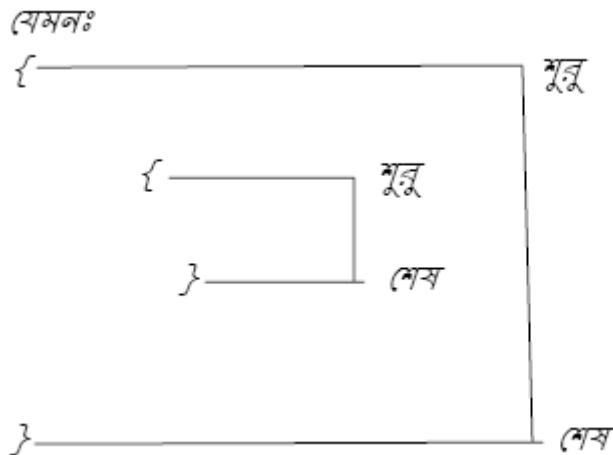
*** লক্ষ্য করুন কোন ফাংশনকে call করতে হলে শুধু মাত্র ফাংশনের নামটো লিখার পর (); লিখলেই ফাংশনটা call হয়ে যাবে। যখন ফাংশনটার ভেতরে কোন মান পাঠাতে হবে তখন () এর ভেতর সেই মানটা দিয়ে দিতে হবে।

void function1(void) ফাংশনের সব statement এক্সিকিউট করা হলে compiler আবার int main() ফাংশনে ফিরে আসবে। এখন int main() ফাংশনে getch(); statement টা এক্সিকিউট করবে। তারপর return 0; statement টা এক্সিকিউট করবে এবং সবশেষে } পাওয়ার পর টার্মিনেট করবে।

*** ‘{’ কে বলা হয় opening brace আর ‘}’ কে বলা হয় closing brace.

*** opening brace থেকে এক্সিকিউশন শুরু হয় আর closing brace এ এক্সিকিউশন শেষ হয়।

*** একটা program এ যতগুলো opening brace থাকবে ঠিক ততগুলো closing brace থাকবে।



প্রশ্নঃ এখন মনে করুন এমন একটা *userdefined function* লিখতে হবে যেন ফাংশনটা call করার সময় একটা *int type* এর ডাটা পাঠানো হয় এবং একটা *int type* এর ডাটা return করে।

উত্তরঃ

function name সিলেক্ট করুন। যেমনঃ *robin*
return type হলো *int*
argument type হলো *int*

অর্থাৎ *userdefined function* টা নিচের মত লিখতে হবেঃ

```
int robin(int a)
```

এখন program টা লিখে ফেলুনঃ

```
#include<iostream.h>  
#include<conio.h>
```

```
int robin(int a)
{
    return a;
}

int main()
{
    clrscr();
    int b;
    b=robin(20);
    cout<<"returning value = "<<b<<"\n";
    getch();
    return 0;
}
```

Sample Output:

```
returning value = 20
_
```