## Content

# 1. Introduction to Programming

Programming ဆိုတာ Code တွေရေးပြီး Computer ကို ခိုင်းစေခြင်းဖြစ်ပါတယ်။ အဲ့ဒီ Code တွေကို ရေးနိုင်ဖို့အတွက် Programming Language တွေကို နားလည် တက်ကျွမ်းဖို့လိုအပ်ပါတယ်။ Computer မှာ Programming Language တွေ များစွာရှိပါတယ်။ သို့သော် Programmer တစ်ယောက် ဖြစ်ဖို့အတွက် Language အကုန်လုံးကိုလေ့လာထားဖို့ မလိုအပ်ပါဘူး။ Programming Language တစ်ခုခုကိုသာလျှင် သေသေချာချာ ဖြစ်ဖြစ်မြောက်မြောက်နားလည် တက်ကျွမ်းဖို့ပဲလိုအပ်ပါတယ်။ Language တစ်ခုကို တတ်မြောက်လျှင် အခြား Language များကို လွယ်လွယ်ကူကူ နားလည် သဘောပေါက်နိုင်ပါတယ်။

**Programming နဲ့ ဘာတွေဖန်တီးလို့ရမလဲ?။** Programming ကို အသုံးပြုပြီး ကျွန်တော်တို့ နေ့စဉ် အသုံးပြုနေတဲ့ Mobile App များ၊ Website များ ကိုဖန်တီးနိုင်ပါတယ်။ ဒါအပြင် အခြား Hardware ထိန်းချုပ်မှုများ၊ Robotics System များ၊ AI Development, Game Development စသည့်တို့ကို ဖန်တီးလို့ရပါတယ်။

## Programming Language အမျိုးအစားများ

Programming Language တွေရဲ့ Level of Abstraction, Syntax, အသုံးပြုပုံတွေအပေါ်မူတည်ပြီးတော့ Language တွေကို ခွဲခြားထားနိုင်ပါတယ်။

| | |
|---|---|
| High-Level Programming Languages | Python, Java, Ruby |
| Low-Level Programming Languages | System-Level Programming |
| Object-Oriented Programming languages | Java, Python and C++ |
| Functional Programming Languages | Haskell, Lisp |
| Scripting Languages | Python, PHP, Perl and Ruby |
| Markup Languages | HTML, XML and CSS |
| Domain-Specific Languages | SQL, MATLAB, R |

Languages တွေကို အထက်ပါအတိုင်း ခွဲခြားထားသော်လည်း အဓိကအားဖြင့် **Complied Language** နဲ့ **Scripting Language** ဆိုပြီး များသောအားဖြင့် Language တွေကို ခွဲခြားနိုင်ပါတယ်။ Scripting Language တွေက များသောအားဖြင့် လွယ်လွယ်ကူကူလေ့လာ တက်မြောက်နိုင်ပါတယ်။

**Complied Languages**

Complied Languages ဆိုတာက Code တွေကို ရေးပြီး Run ကြည့်ဖို့အတွက် Compiler တစ်ခု လိုအပ်တဲ့ Languages တွေကိုဆိုလိုတာပါ။ Compiler က ရေးလိုက်တဲ့ Code တွေကို Machine Code (Computer နားလည်တဲ့ Code) အဖြစ်ပြောင်းလဲပေးပါတယ်။ Complied Language တွေက Machine Code အဖြစ် ပြန်ပြောင်းပေးတာကြောင့် Scripting Language တွေထက် ပိုပြီးတော့ မြန်တယ်၊ Efficiency ပိုများတယ်ပေါ့။ Complied Languages များး **– C, C++, Java, Fortran, Swift, Rust, …**

**Scripting Languages**

Scripting Languages (သို့) Interpreted Languages တွေကို Compiler မပါဘဲ Run နိုင်ပါ တယ်။ Scripting Languages တွေက ရေးလိုက်တဲ့ Code တွေကို တစ်ခုချင်းစီ Line by Line, interpreted လုပ်ပြီးတော့ Code တွေကို Executes လုပ်ပါတယ်။ အဲ့တာကြောင့် Scripting Languages တွေကို Interpreted Languages လို့လဲခေါ်ကြပါတယ်။ Scripting Languages တွေကို အဓိကအားဖြင့် Web Development ပိုင်းနဲ့ Data Analysis ပိုင်းမှာ အသုံးပြုတာများပါတယ်။ Scripting Languages များး – **Python, JavaScript, Ruby, PHP, …**

## Program တစ်ခုကို စရေးဖို့အတွက် အနည်းဆုံး ဘာTools တွေလိုအပ်မလဲ?

**– Text Editor or IDEs**

**– Compiler or Interpreter**

**– Command–line Interface (CLI)**

Program တစ်ခုကိုစရေးဖို့အတွက် အထက်ပါ Tools တွေလိုအပ်ပါတယ်။ Code တွေကို Computer မှာ Built In ပါတဲ့ Text Editor တစ်ခုခုနဲ့ အလွယ်တကူရေးလို့ရပါတယ်။ **IDEs (Integrated Development Environments) ဆိုတာ Code ရေးဖို့အတွက် Text Editor တွေကို Compiler/Interpreter, Debugger တွေနဲ့ အထူးပြုလုပ်ထားတဲ့ Software တွေဖြစ်ပါတယ်။** IDE တွေက Code တွေကို လျှင်လျှင်မြန်မြန်ရေးနိုင်အောင်ကူညီပေးပါတယ်။ နောက်တစ်ခုက Code တွေကို Executes လုပ်ဖို့အတွက် Compiler/Interpreter တွေပါ။ **Command–Line Interface ဆို တာ Compiler မှ Executes လုပ်လိုက်တဲ့ Input/Output တွေကို ပြသပေးတဲ့ Tools ဖြစ်ပါတယ်။** ဒီလောက်ဆို Program တစ်ခုကိုစရေးလို့ရပါပြီ။

## 1.1 Overview of C++

C++ ကို 1980 အစောပိုင်းက Bjarne Stroustrup ဆိုသူက C Languages ကိုအခြေခံပြီးဖန်တီးခဲ့ပါတယ်။ ထို့ကြောင့် C Language မှာရှိတဲ့ Syntax နဲ့ Code Structure တွေက C ++ နဲ့ Similar ဖြစ်ပါတယ်။ System Level Programs , Operating Systems တွေကို ရေးသားရာ၌ C++ ကို ရွေးချယ် ပါတယ်။ C++ ရဲ့ Speed, Flexibility တွေက ကောင်းသောကြောင့် Finance တွေ၊ Game Development တွေနဲ့ High–Performance Computing တွေမှာ အဓိကအသုံးပြုကြပါတယ်။ C++ မှာMemory Management ကို အခြား Language တွေထက် ပိုပြီး Manual လုပ်နိုင်တာပါ။ Memory Manage လုပ်နိုင်တဲ့အတွက်ကြောင့် C++ နဲ့ ရေးတဲ့ Application တွေက ပိုပြီးတော့ Performance ကောင်းတာကို တွေ့နိုင်ပါတယ်။ C++ က Compiled Language တစ်ခုဖြစ်ပါတယ်။ OOP ကို အခြေခံထားတာကြောင့် Complex Program တွေကို ရေးတဲ့အခါမှာ လွယ်ကူစွာ Manage လုပ်နိုင်မှာဖြစ်ပါတယ်။


### C++ Compilers

Compiler ဆိုတာ အထက်မှာ ပြောခဲ့သလိုမျိုး၊ Written Code တွေကို Executable Code အဖြစ် ပြောင်းလဲပေးတာဖြစ်ပါတယ်။ C++ မှာ **Open Source Compiler** တွေကော **Commercial Compiler** တွေကော အများကြီးရှိပါတယ်။ Popular ဖြစ်တဲ့ Compiler တွေကို အောက်မှာ ဖော်ပြထားပေးပါတယ်။

| Compiler | Description |
|---|---|
| **GCC (GNU Compiler Collection)** | Free and open-source compiler that supports multiple programming languages, including C++. |
| **Clang** | It is designed to be fast and flexible and is widely used in the development of Apple's operating systems. |
| **Microsoft Visual C++** | Commercial compiler, It is widely used in the development of windows application and games |
| **Intel C++** | Commercial compiler, It is designed to produce high-performance code and used in scientific and engineering communities |

| | |
|---|---|
| **Borland C++** | Commercial compiler, It is used in the development of Windows application and games. |

Compiler တွေကို ရွေးချယ်တဲ့ နေရာမှာ Compiler ရဲ့ Performance, Compatibility, Support Platforms and Technologies စတဲ့ အချက်တွေကို ကြည့်ပြီးရွေးချယ်ရပါမယ်။

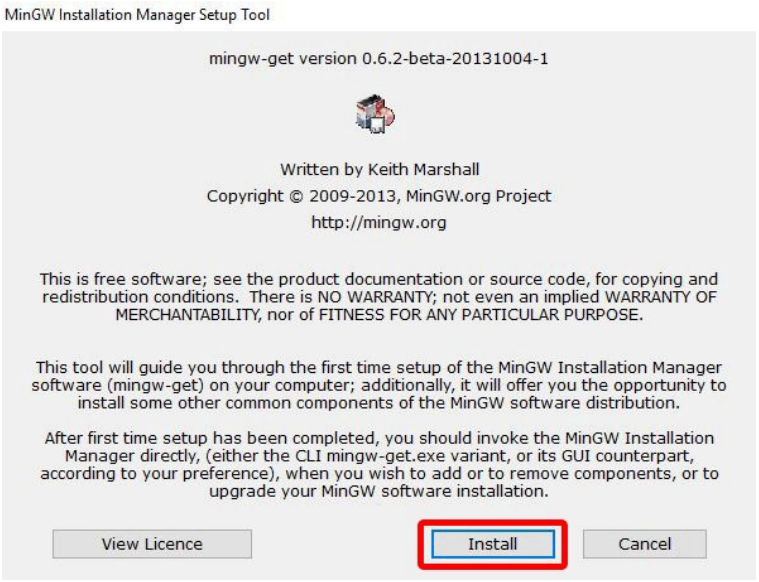
## 1.2 Environment Setup for C++

C++ Program တစ်ခုကိုမရေးခင် သင်၏ Computer မှာ C++ Compiler တစ်ခုကို Install လုပ်ထားဖို့လိုပါတယ်။ ကျွန်တော်ကတော့ ဒီ စာအုပ်မှာ GNU (GCC) Compiler ကိုပဲ အသုံးပြုသွားမှာ ဖြစ်တာကြောင့် GNU Compiler ကိုပဲ Install လုပ်ပြမှာဖြစ်ပါတယ်။

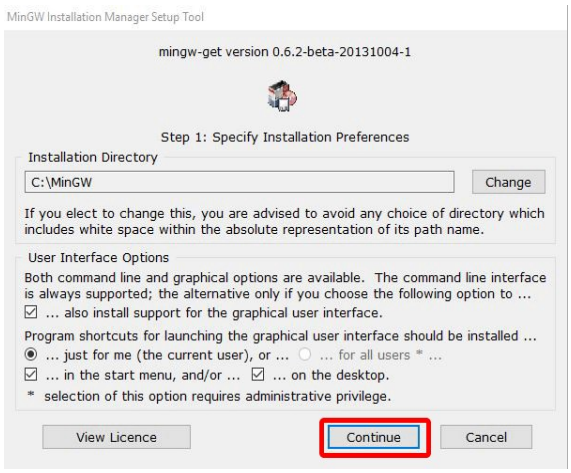
1. GNU (GCC) Compiler ကို Install လုပ်ဖို့ အတွက် အောက်က Link ကို နှိပ်ပြီး ဒေါင်းလုတ်ဆွဲပါ။

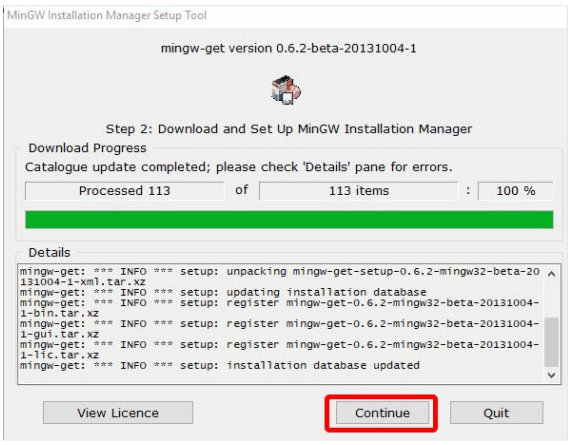
https://sourceforge.net/projects/mingw/files/Installer/mingw-get-setup.exe/download

2. Download ဆွဲပြီးရင် mingw-get-setup.exe file ကို Download Folder ထဲကနေ ဖွင့်ပြီး Install လုပ်ပါ။

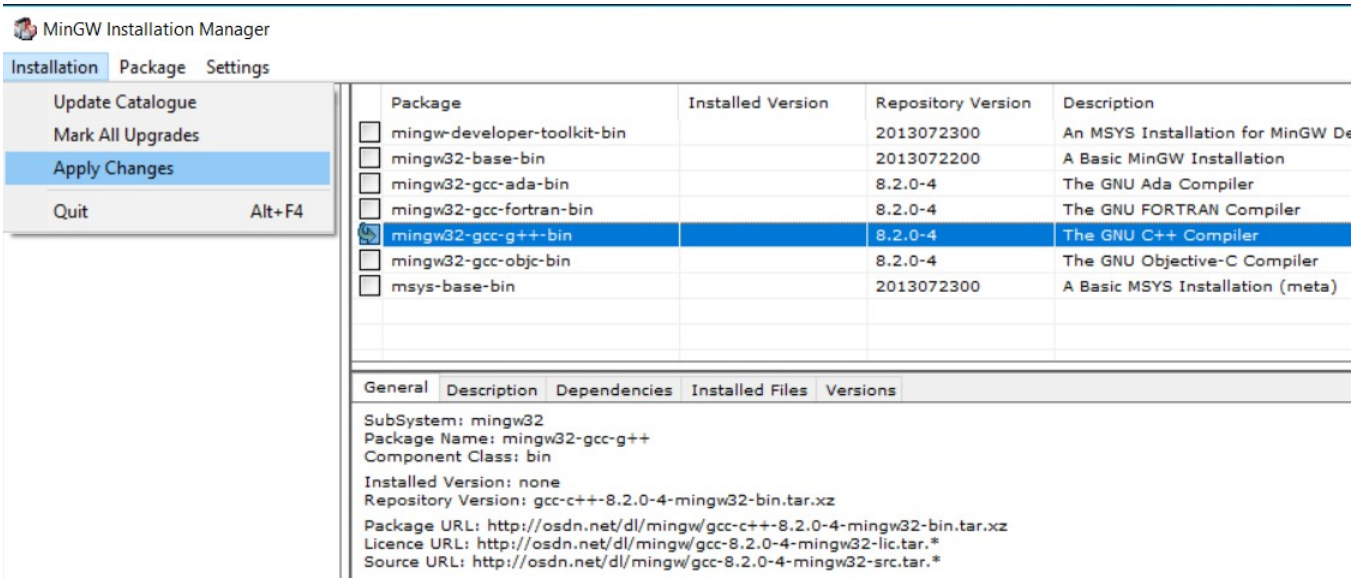



3. Setting တွေကို Default အတိုင်းထားပြီး Continue ကိုသာနှိပ်ပါ။ ပြီးရင် Download ဆွဲနေတာလေးကို ခဏစောင့်လိုက်ပါ။

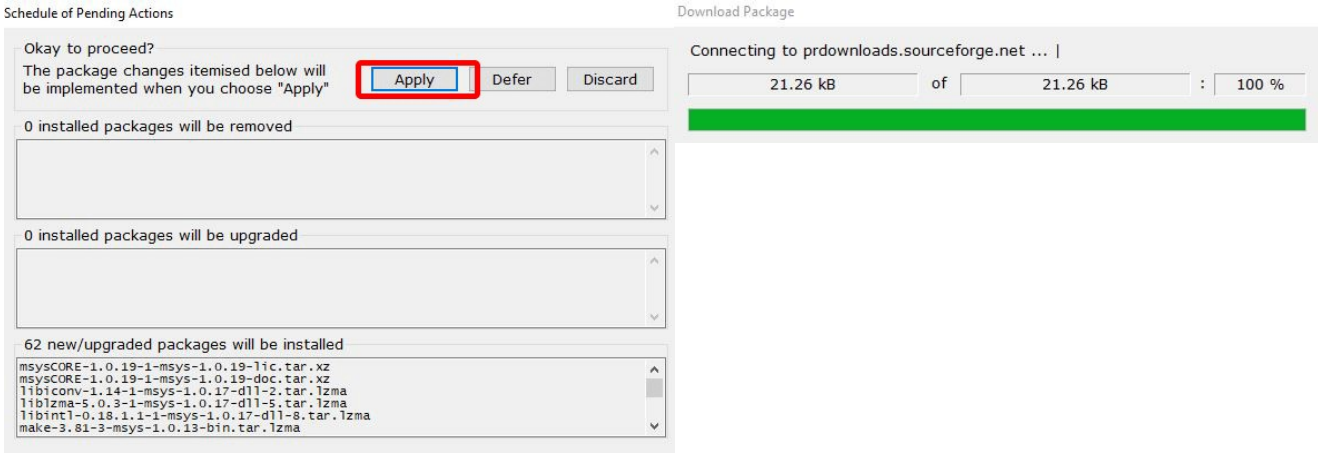


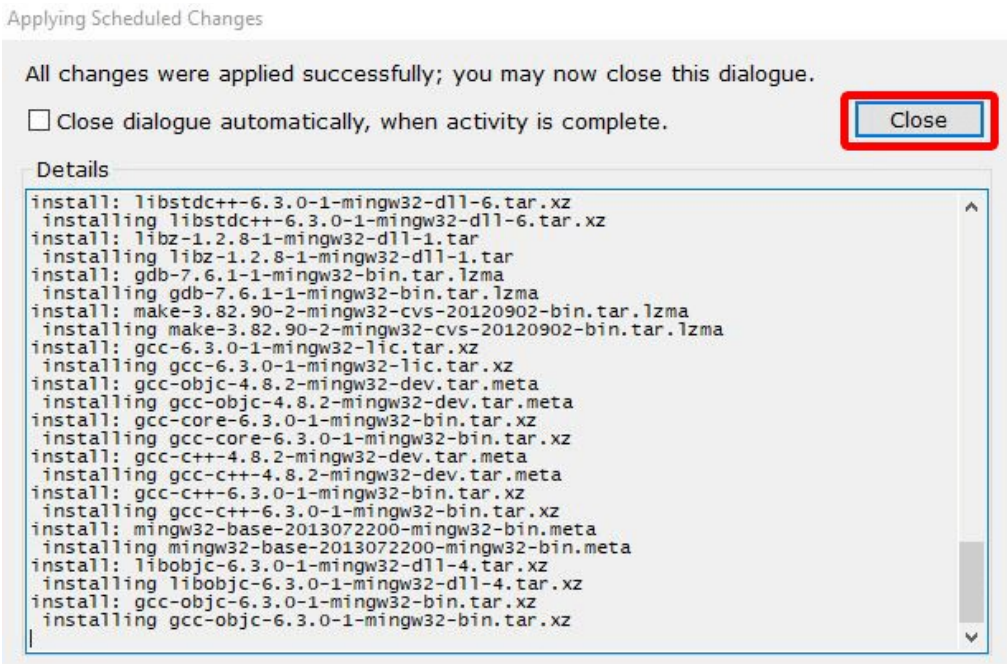
4. ဒီ Screen ကိုမြင်လျှင် mingw32-gcc-g++-bin ဆိုတာ လေးကို ဘေးနားက အစိမ်းမျဉ်းကွေးလေး ပေါ်အောင်နှိပ်ပေးပါ။ ပြီးလျှင်အပေါ်မှာရှိတဲ့ Installation ထဲက Apply Changes ဆိုတာကို နှိပ်လိုက်ပါ။




5. Apply ကိုထပ်နှိပ်ပါ။ Download ဆွဲနေတာလေးကိုခဏစောင့်ပါ။




6. Install ဆွဲနေတာကို ခဏစောင့်ပြီး Close ကိုနှိပ်လိုက်ပါ။ ဒါဆိုရင် GCC compiler ကိုအောင်မြင်စွာ Install လုပ်ပြီးပါပြီခင်ဗျာ။



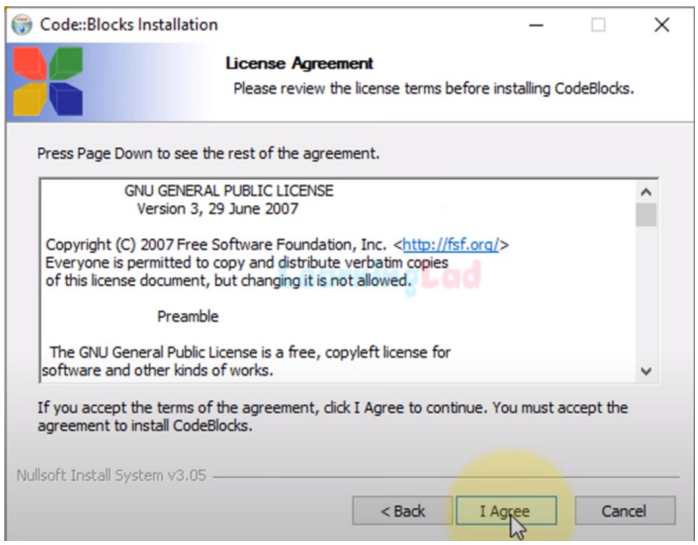
## *Code::Blocks Installation*

Code::Blocks ဆိုတာ C, C++ တို့အတွက် အထူးပြုလုပ်ထားတဲ့ IDE တစ်ခုပါ။ တခြား IDE တွေ သုံးချင်လည်း သုံးလို့ရပါတယ် အရေးကြီးတာက Code ရေးဖို့အတွက်ပါပဲ ကျွန်တော်ကတော့ ဒီစာအုပ်မှာ Code::Blocks နဲ့ပဲ သင်မှာဖြစ်တဲ့အတွက် Code::Blocks ပဲ Install လုပ်ပြပါမယ်။

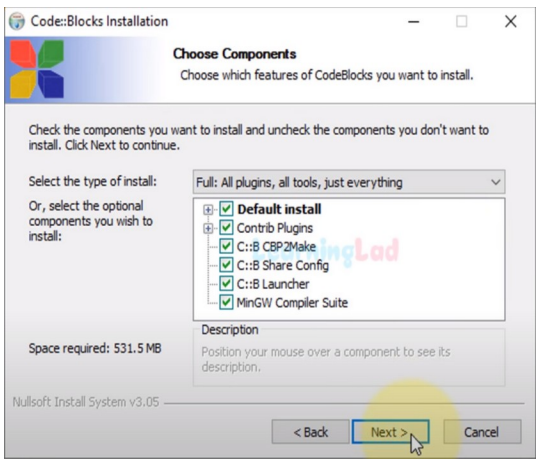
1. Code::Blocks Setup File အားပေးထားသော Link မှ Download ဆွဲပါ။

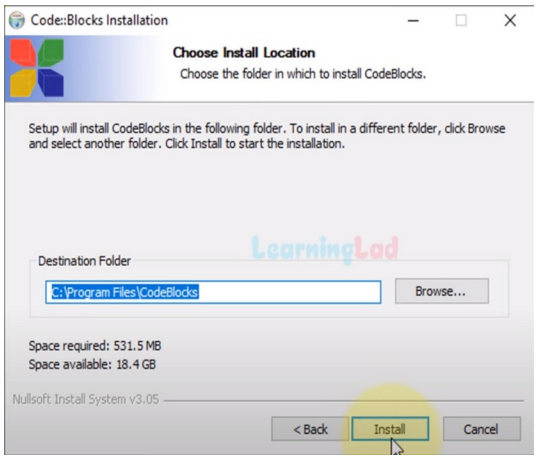
https://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/codeblocks-20.03-setup.exe/download

2. ပြီးလျှင် Code::Blocks Setup File အားဖွင့်၍ Install ဆွဲပါ။

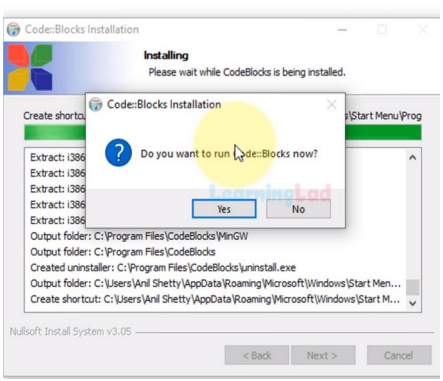


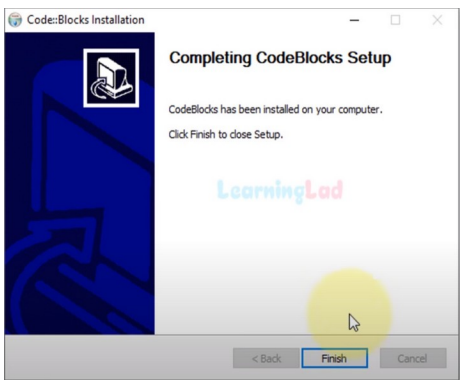

3. Code::Blocks ကို Install ဆွဲတဲ့နေရာမှာ **Next/Install** Button ကိုသာ ဆက်တိုက်နှိပ်ပါ။




4. ဒါဆိုရင် **Install** လုပ်လို့ပြီးပါပြီ။ ပြီးလျှင် **Finish** ကိုနှိပ်ပ်လိုက်ပါ။



## 1.3 Basic Structure of C++ Programs

အားလုံး Install လုပ်ပြီးသွားရင် Code::Block ထဲမှာ C++ Programs တစ်ခုကိုစရေးလို့ရပါပြီ။ File > New> Empty File တစ်ခုကိုရွေးလိုက်ပါ။ ပြီးလျှင် File ကို Save ပါ။ Save တဲ့နေရမှာ File Name ရဲ့ နောက်မှာ C++ extension *.cpp ဖြင့် save ပါ။ ဥပမာ – helloworld.cpp ပေါ့။ ဒါဆိုရင် အောက်က Code ကို စရေးလို့ရပါပြီ။

```
#include <iostream>


int main() {
   std::cout << "Hello, World!";
   return 0;
}
```

ဒါကတော့ Hello World ဆိုတဲ့ စာသားကို Display ပေါ်မှာမြင်ရအောင် print ထုတ်ပေးတဲ့ Program ပဲဖြစ်ပါတယ်။ #include <iostream> ဆိုတာ Computer ပေါ်မှာ မြင်ရတဲ့ Input/Output တွေကို မြင်ရအောင် ပြုလုပ်ပေးတဲ့ Objects တွေကို Code ထဲမှာ အသုံးပြုလို့ရအောင် Import လုပ်လိုက်တာဖြစ်ပါတယ်။ int main(){... return 0;} ဆိုတဲ့ Functions က C++ မှာ မရှိမဖြစ် ပါဝင်ရမယ့် Functions တစ်ခုဖြစ်ပါတယ်။ C++ Program တစ်ခုကို စ run လိုက်တာနဲ့ int main() ဆိုတဲ့ Function ထဲမှာရှိတဲ့ Code Block တွေကို အရင်စ Run တာဖြစ်ပါတယ်။ std::cout << "Hello, World!"; ဆိုတာကတော့ Hello, World! ဆိုတဲ့ စာသားကို Display ပေါ်မှာ မြင်ရအောင် ထုတ်ပေးတဲ့ Syntax ပဲဖြစ်ပါတယ်။

1. Code တွေရဲ့ syntax တစ်ခုဆုံးတိုင်း Semicolon တွေကို ထည့်ဖို့လိုအပ်ပါတယ်။

```
std::cout << "Hello World" ;
```

2. C++ မှာပါဝင်တဲ့ Built In Functions တွေ, Data Types တွေ, Code Syntax တွေကို မှားလို့မရပါဘူး။ မှားမယ်ဆိုရင် Complier မှာ Erros တွေပြပါတယ်။ Error တွေကို ဖတ်တက်ဖို့လဲ အရေးကြီးပါတယ်။

```
std:: cout < "Hello World";
```

```
error: no match for 'operator<' (operand types are 'std::ostream' {aka 'std::basic_ostream<char>'} and...
note: candidate: 'template<class _T1, class _T2> constexpr bool std::operator<(const std::pair<_T1, _T...
note:   template argument deduction/substitution failed:
note:   'std::ostream' {aka 'std::basic_ostream<char>'} is not derived from 'const std::pair<_T1, _T2>'
note: candidate: 'template<class _Iterator> constexpr bool std::operator<(const std::reverse_iterator<...
```

3. ရေးပြီးသား Code တွေကို မ Run ခင် Compiler မှာ Code ကို Build လုပ်ပါတယ်။ Build လုပ်တယ်ဆိုတာ Readable Code တွေကို Exectuable Code (Machine Code ) အဖြစ်ပြောင်းလဲပေးတာဖြစ်ပါတယ်။

4. C++ မှာ Code တွေကို စ Run လိုက်တာနဲ့ main () ဆိုတဲ့ function ထဲမှာရှိတဲ့ Code တွေကို Line by Line စ Run ပါတယ်။ ဒါကြောင့် ကိုယ်လုပ်ချင်တဲ့ Code အစီအစဉ်တွေကို Line by Line ရေးဖို့ အရေးကြီးပါတယ်။ ဥပမာ –

1. အိပ်ယာထမယ်
2. သွားတိုက်မယ်
3. မနက်စာစားမယ်

ကိုယ်ဖြစ်ချင်တဲ့ အကြောင်းအရာတွေကို အစဉ်လိုက်ရေးတက်ဖို့ အရေးကြီးပါတယ်။ ဒါပြီးရင် ဒါလုပ်ဆိုတာ ကို Computer က နားလည်ဖို့အတွက်ပါ။

### 1.4 C++ Comments

ကျွန်တော်တို့ Program တွေ Projects တွေကို Team တွေနဲ့ အတူတူလုပ်တဲ့ အခါမှာ ကိုယ့်ရဲ့ Code တွေက Easy to Read ဖြစ်ဖို့အရေးကြီးပါတယ်။ ကိုယ်ရေးတဲ့ Code တွေက Complex ဖြစ်နေတဲ့ အခါမှာ ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာကို ရှင်းပြတာပိုပြီးတော့ ကောင်းပါတယ်။ ဒီလိုရှင်းပြဖို့ အတွက် Comments တွေကို အသုံးပြုရပါမယ်။ Comments တွေက C++ Complier ထဲမှာ Execute မလုပ်ပါဘူး ဒါကြောင့် ကြိုက်သလိုရေးလို့ရပါတယ် Errors မတက်ပါဘူး။

### Single-line Comments

Comment တစ်ကြောင်းထဲရေးတာပါ။ Comments တစ်ကြောင်းထဲ ရေးရင် အရှေ့မှာ Slashes (//) Symbol လေးခံပေးရပါမယ်။

```
//This is Hello World Program

cout << "Hello World!";
```

### Multi-line Comments

Comments တွေအများကြီးရေးချင်တယ်ဆိုရင် /* နဲ့စပြီး စာကြောင်းအဆုံးမှာ */ အဆုံးသတ်ပေးရပါတယ်။

```
/*Multil line comments will multi line

Hello World Program

*/

cout << "Hello World!";
```

## 2. Using Input and Output

Computer Program တွေ Software တွေရဲ့ ရည်ရွယ်ချက်က User Input တွေကို လက်ခံမယ် ပြီးတော့ User Input တွေကနေ Data တွေကို Processing လုပ်ပြီးရင်User ရှိကို Output လုပ်ပေးမယ် ဒါကြောင့် Program တစ်ခုမှာ ပါသင့်တဲ့ အချက်တစ်ခုကတော့ Input နဲ့ Ouput ပါပဲ။ ဒါကို ကျွန်တော် တို့က C++ Language နဲ့ User တွေရှိကနေ Input ယူမယ်၊ ပြီးရင် Output ပြန်ထုတ်ပေးလို့ရပါတယ်။ အဲ့တာဆိုရင် Input/Output Library ကို ကျွန်တော်တို့ ထည့်ရပါမယ်။ C++ ရဲ့ Input/Output Library က <iostream> ပါ။ ဒါကို C++ program ထဲကို ထည့်မည်ဆိုလျှင် #include <iostream> ဆိုပြီး ထည့်ရ ပါမယ်။ အဲ့ဒီ Input/Output Library ကိုထည့်မှာ program မှာ သူနဲ့ သက်ဆိုင်တဲ့ cin , cout ဆိုတဲ့ syntax တွေကို အသုံးပြုလို့ရမှာဖြစ်ပါတယ်။

### 2.1 Using cout

cout ဆိုတဲ့ syntax ကို ကျွန်တော်တို့ Hello World Program မှာ အသုံးပြုထားပါသေးတယ်။ အဲ့ တော့ cout ဆိုတာက User ရှိကို Ouput ထုတ်ပြပေးတာဖြစ်ပါတယ်။ cout ဆိုတဲ့ syntax က <iostream> ဆိုတဲ့ Standard Libarary ထဲမှာပါပါတယ်။ အဲ့တာကြောင့် cout ကို အသုံးပြုဖို့ဆိုရင် #include <iostream> ဆိုပြီး Code ရဲ့ အပေါ်ပိုင်းမှာ ရေးရပါမယ်။ ဒါဆိုရင် ကျွန်တော်တို့ Simple Program တစ်ခုစရေးကြည့်ပါမယ်။

1. File > New > Empty File ပြီးရင် filename တစ်ခုပေးပြီး save လိုက်ပါ။ ဥပမာ – output.cpp, File Save ပြီဆိုရင် Filename Extensions ကို .cpp ပါအောင် save ရပါမယ်။

2. ဒါဆိုရင် အောက်က Code ကို စရေးလို့ရပါပြီ။

```
#include <iostream>
int main(){
	std::cout << "I am output" << std::endl;
	std::cout << "I am output too" << std::endl;
	return 0;
}
```

3. #include <iostream>

<iostream> ဆိုတဲ့ Library ကို Import လုပ်တယ်။

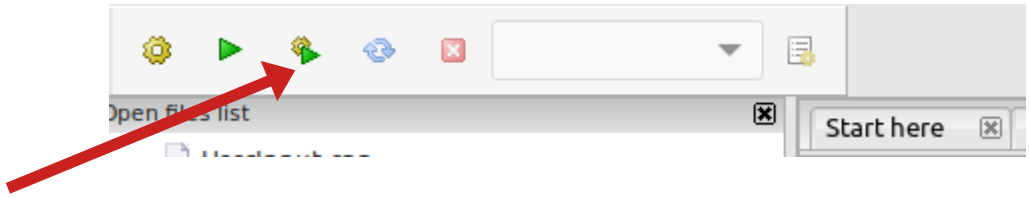
4. int main(){ .... return 0; } ဆိုတာက Code တွေကို စ Run မဲ့ main function ဖြစ်ပါတယ်။ Code တွေကို main function ရဲ့ { } ထဲမှာ စရေးရပါမယ်။

5. std::cout << "I am output" << endl; ဆိုတာက တော့ "I am output" ကို ထုတ်ပေးတာဖြစ်ပါ တယ်။ std:: ဆိုတာကတော့ Standard Library တွေကို အသုံးပြုတဲ့ အခါမှာ ရေးပေးရပါတယ်။ cout

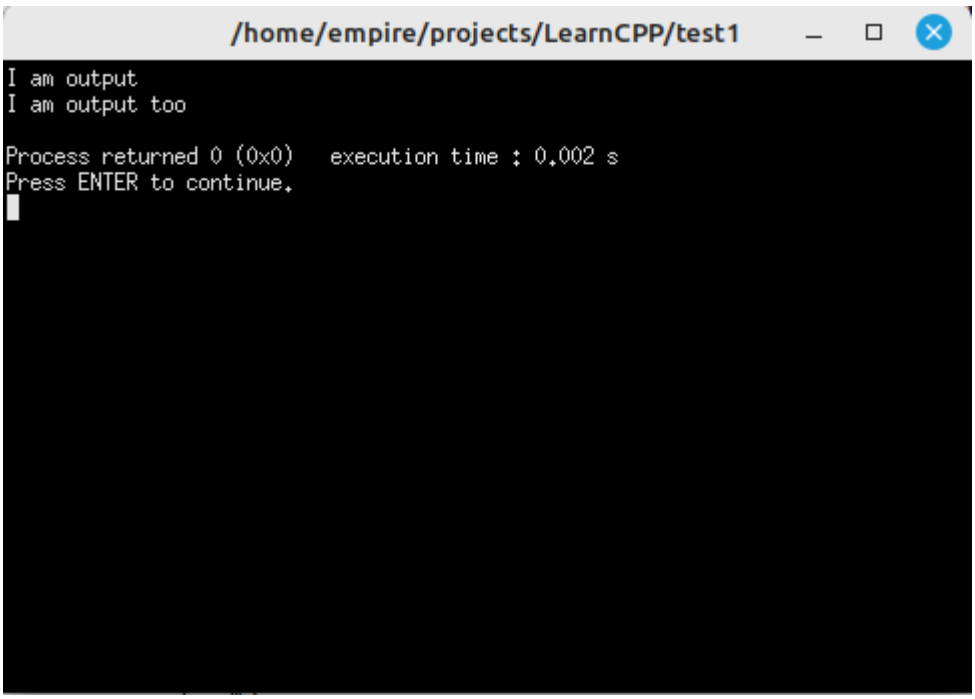
ဆိုတာက Standard Library ဖြစ်တဲ့အတွက်ကြောင့် cout ကို အသုံးပြုဖို့အတွက် std:: ဆိုတာကို cout ရဲ့ အရှေ့မှာ ရေးပေးဖို့လိုပါတယ်။

cout ဆိုတဲ့ syntax က << နောက်မှာရှိတဲ့ စာသားတွေကို output လုပ်ပေးပါတယ်။ ဒါကြောင့် "I am output" ဆိုတာကို Ouput မှာ ထုတ်ပေးပါတယ်။ std::endl; ဆိုတာကတော့ စာကြောင်း ရဲ့နောက်မှာ Enter ခေါက်တာ ဖြစ်ပါတယ်။

6. ဒါဆိုရင် Code ကို စ Build ပြီးတော့ Run လို့ရပါပြီ။ Code::Blocks မှာဆိုလျှင် F9 ကိုနှိပ်ပြီးတော့ Build and Run လုပ်လို့ရပါတယ်။




7. Ouput ကို ပိတ်မယ်ဆိုလျှင် Enter ကိုခေါက်ပါ။




### 2.2 Using cin

ဒါဆိုရင် User Input တွေကို ယူမယ်ဆိုရင် cin ဆိုတာ ကို အသုံးပြုရမှာဖြစ်ပါတယ်။ cin ဆိုတာ ကလည်း Standard Library ဖြစ်တာကြောင့် cin ရဲ့ ရှေ့မှာ std:: ဆိုတာကို ထည့်ပေးရပါမယ်။ cin ကို သုံးမယ် ဆိုရင် >> ဆိုတဲ့ Operator ကို အသုံးပြုရပါမယ်။ << ဆိုတဲ့ Operator က cin ထဲက User Input Data တွေကို export လုပ်လိုက်တာဖြစ်ပါတယ်။ cin ဆိုတာက User Input လုပ်လိုက်တဲ့ Data ထဲက One Word ကိုသာ export လုပ်ပေးပါတယ်။ ဥပမာ - User က "Hello World" လို့ရိုက်လျှင် "Hello" ဆိုတာကို သာ export လုပ်ပါတယ်။

1. ဒါဆိုရင် File တစ်ခုကို ဆောက်ပြီး Code စရေးကြည့်ကြပါမယ်။

```
#include <iostream>
int main()
{
    std::string text;
    std::cout << "Type Something : ";
    std::cin >> text;
    std::cout << text << std::endl;
    return 0;
}
```

2. ဒါဆိုရင် Code တွေကို ရေးနေကြ Code Structure တိုင်းရေးပါမယ်။ #include , main () တွေရေးမယ်ပေါ့။

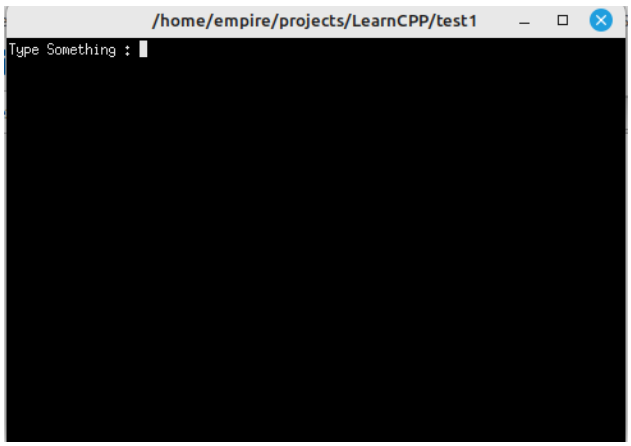
3. std::string text; ဆိုတာက String Variable တစ်ခုကို Declare လုပ်လိုက်တာပါ။ Variable အကြောင်းတွေကို နောက်အခန်းတွေမှာရှင်းပြပေးထားပါတယ်။ လောလောဆယ်တော့ Data Store လုပ်ဖို့အတွက်သာမှတ်ထားလိုက်ပါ။

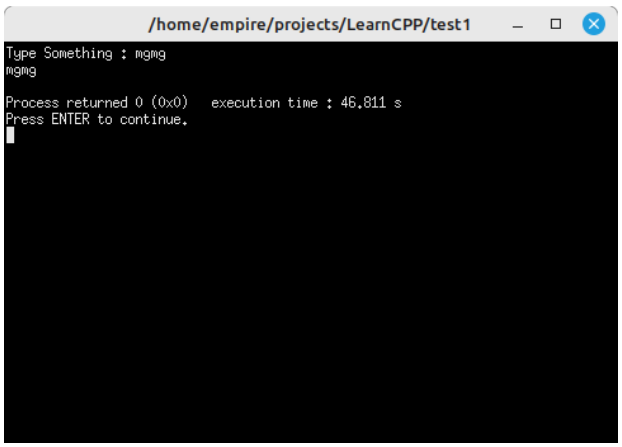
4. std::cout << "Type Something : " , User ကို ဘာလုပ်ရမလဲ ညွှန်ပြဖို့အတွက် ထည့်လိုက်တာပါ။ မထည့်လည်းရပါတယ်။

5. std::cin >> text; User က စာလုံး တစ်ခုခုရိုက်ပြီး Enter ခေါက်လိုက်မယ်ဆိုရင် text ဆိုတဲ့ variable ထဲကို cin ကနေ export လုပ်လိုက်တာပါ။ အဲ့တော့ User Input လုပ်လိုက်တဲ့ စာလုံးတွေ အားလုံးက text ဆိုတဲ့ variable ထဲမှာ Store လုပ်သွားပါပြီ။

6. std::cout<< text ; store လုပ်တဲ့ data တွေကို ပြန်ပြီးတော့ Output ထုတ်လိုက်ပါမယ်။

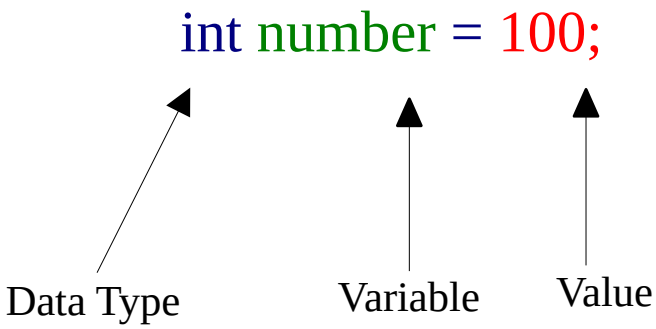
7. အဲ့တော့ Cli မှာ စ Run လို့ရပါပြီ။



## 3. Data Types and Variables

C++ အပါဝင် အခြား Programming Language တိုင်းမှာ အရေးကြီးတဲ့ အစိတ်အပိုင်းတစ်ခုကတော့ Data Types တွေနဲ့ Variable တွေပဲ ဖြစ်ပါတယ်။ Data Types, Variables တွေက Computer ရဲ့ Memory ထဲမှာ Data တွေကို Store လုပ်တယ် ပြီးတော့ Variables တွေက Manipulation လုပ်ဖို့ အတွက် သုံးပါတယ်။




### 3.1 Data Types

Computer ရဲ့ Memory ပေါ်မှာ Data တွေ ဘယ်လိုပုံစံနဲ့ Store လုပ်မယ်ဆိုတာကို Data Types တွေနဲ့ ခွဲခြားနိုင်ပါတယ်။ ဥပမာ - စာသားတွေနဲ့ Memory ပေါ်မှာ Store လုပ်မှာလား၊ ဂဏန်းတွေနဲ့ Store လုပ်မှာလားဆိုတာကို ခွဲခြားနိုင်ပါတယ်။.

**Example**

```
int number = 123;
double DoubleNum = 6.58;
float floatNum = 7.66;
char letter = 'C';
bool boolean = true;
string text = "Hello";
```

အောက်ကဇယားထဲမှာ C++ မှာ built in ပါတဲ့ Data Type တွေပြထားပေးပါတယ်။

| Data Type | Size | Description |
|---|---|---|
| `int` | 2 or 4 bytes | Stores whole numbers, without decimal |
| `double` | 8 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits |
| `float` | 4 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 6 – 7 decimal digits |
| `char` | 1 byte | Stores a single character/letter/number, or ASCII values |
| `boolean` | 1 byte | Stores true or false values |

အပေါ် Example မှာ ပြထားတဲ့ string Variable က C++ ရဲ့ Built In Data Type မဟုတ်ပါဘူး။ အဲ့တာကြောင့်သူ့ကို Library အနေနဲ့ ထည့်ရပါမယ်။ String ကို Library အနေနဲ့ ထည့်မယ်ဆိုရင် header ဖိုင်းမှာ #include <string> ဆိုပြီးရေးရပါမယ်။

| Data Type | Size | Description |
|---|---|---|
| string | - | Store a sequence of characters, such as letters, digits, and symbols. |

### 3.2 Variables

ကျွန်တော်တို့ Data Types တွေကို Declare လုပ်ယုံနဲ့ Memory ပေါ်မှာ Store လုပ်လို့မရပါဘူး။ Memory ပေါ်မှာ Variables တွေကို အမည်ပေးပြီး Store လုပ်ရပါမယ်။ ဒါမှ Variable Name တွေကို ခေါ်ပြီး Processing ပြန်လုပ်လို့ရမှာဖြစ်ပါတယ်။ **Variables ဆိုတာ Data Types တွေကို အမည်ပေးပြီး Memory ပေါ်မှာ Store လုပ်တာဖြစ်ပါတယ်။** Variables တွေက နာမည်ပေးတဲ့ နေရာမှာ -

1 . ပေးပြီးသား Name တွေပြန်ပေးလို့မရပါဘူး။ (Name တစ်ခုနဲ့ တစ်ခုတူလို့မရပါဘူး)

2. Value နဲ့သက်ဆိုင်တဲ့ နာမည်တွေကိုပေးရပါမယ်။ (Code တွေကို ဖတ်ရလွယ်ရမယ်)

3. C++ ရဲ့ keywords တွေနဲ့ ညှိလို့မရပါဘူး။

4. ဂဏန်းတွေနဲ့ စလို့မရပါဘူး။ စာသားတွေနဲ့ စရေးရပါမယ်။

```
string 1name = “mgmg”;
string name1 = "mgmg";
```

5. white spaces တွေပါလို့မရပါဘူး။ Special Characters တွေပါလို့မရပါဘူး။ !@#$%&...

6. ကျန်တာကတော့ name တွေကို အဆင်ပြေသလို ပေးလို့ရပါတယ်။

#### *3.2.1 Declare Multiple Variables*

တူညီတဲ့ Data Types တွေသုံးတဲ့နေရာမှာ Variables တွေကို ကော်မာခံပြီးတော့ ကြောငြာ လို့ရပါတယ်။

```
int x = 10, y = 15, z = 20;
cout << x + y + z ;  // output : 45
```

#### *3.2.2 One Value to Multiple Variables*

```
 int x, y, z;
x = y = z = 25;
cout << x + y + z; //output : 50
```

### 3.3 Declaring Constant Variables

Variables Value တွေကို ပြန်မချိန်းချင်ဘူး ပုံသေ သတ်မှတ်ထားတဲ့ Variables အတိုင်းပဲသုံးချင်တယ်ဆိုရင် **const** keyword ကို variables ရဲ့ ရှေ့မှတ်ထည့်ပေးရပါမယ်။ const keyword ကို variables ရဲ့ ရှေ့မှာ သုံးလိုက်မယ်ဆိုရင် ၎င်း Variable က Read Only ဖြစ်သွားပြီ၊ Value တွေကို ပြန် change လို့မရတော့ပါဘူး။

```
const int myNum = 22;
cout << MyNum // Output : 22
MyNum = 10; // error
```

**Before Const**

```
int myNum = 22;
MyNum = 10;
```

**After Const (Error)**

```
const int myNum = 22;
MyNum = 10; // error
```

# 4.Strings

Program တစ်ခုကိုရေးတဲ့အခါ string သည် မသုံးမဖြစ်သုံးရမယ့် Data Type အမျိုးအစားတစ်ခု ဖြစ်ပါတယ် User ရဲ့ Input/Output တွေကို Store လုပ်တဲ့အချိန်မှာ String Data Type ကို အသုံးပြုရမှာဖြစ်ပါတယ်။ ဒီအခန်းမှာတော့ String ကို ဘယ်လိုနေရာတွေမှာ ဘယ်လိုအသုံးချသင့်လဲဆိုတာကို ရှင်းပြပေးသွားမှာဖြစ်ပါတယ်။ String ဆိုတာ အရှေ့ဘက်မှာပြောခဲ့တဲ့အတိုင်း Standard Library ထဲမှာရှိတဲ့ အစိတ်အပိုင်းတစ်ခုဖြစ်ပါတယ်။ အဲ့တာကြောင့် Library ထဲက ထဲ့ဖို့ #include <string> ကို အရင် ရေးဖို့လိုပါတယ်။ String တွေကို "" ထဲမှပဲ ရေးပါတယ်။ "" ရေးထားတာမှန်သမျှကို string လို့ သတ်မှတ်ပါတယ်။

```
#include <iostream>

#include <string>

int main{

        std::string myText = "I am Text" ;

        std::cout << myText;

        return 0;

}
```

## 4.1 Omitting Name Space

String တွေအကြောင်းကို မပြောခင် std:: အကြောင်းကို အရင်ပြောချင်ပါတယ်။ ကျွန်တော်တို့ Standard Library တွေအသုံးပြုတဲ့အခါမှာ std:: ဆိုတဲ့ keyword လေးကို အရှေ့မှာ အရင်ရေးပြီးတော့ အသုံးပြုရပါတယ်။ ဒါကြီးကို အရှေ့မှာ မခံဘဲ Standard Library တွေကို အသုံးပြုလို့ရပါတယ်။ အဲ့ဒါကတော့ namespace ကိုအသုံးပြုရမှာပါ။

```
        #include <iostream>

        #include <string>

        using namespace std;

        int main {

                cout << “Hello World”;

                return 0;

        }
```

**Before using namespace**

```
std::cout << “Hello World”;
```

**After using namespace**

```
using namespace std;
...
cout << “Hello World”;
```

## 4.2 Concatenation

Concatentaion ဆိုတာ string တစ်ခုနဲ့ တစ်ခုကို ဆက်တာဖြစ်ပါတယ်။ တစ်ခုထက်ပိုပြီးတော့လည်း combine လုပ်လို့ရပါတယ်။ String တွေကို combine လုပ်တဲ့နေရာမှာ နည်းနှစ်မျိုးရှိပါတယ် + Operator ကို အသုံးပြုပြီးလုပ်တဲ့နည်းနဲ့ string ရဲ့ append() function ကို အသုံးပြုတာပဲဖြစ်ပါတယ်။

***Using + Operator***

```
string firstName = "Mg";
string lastName = "Hla";
string fullName = firstName + lastName;
cout << fullName; //Mg Hla
```

***Using append() function***

```
string fullName = firstName.append(lastName);
cout << fullName; //Mg Hla
```

## 4.3 User Input String

ကျွန်တော်တို့ Input/Ouput အခန်းမှာ User ရိုက နေ Input ယူတဲ့အခါမှာ cin ကို အသုံးပြုခဲ့ဖူးပါတယ်။ ဒါပေမဲ့ cin က User က Input လုပ်လိုက်တဲ့ one word ကိုပဲ extract လုပ်ပေးတာဖြစ်ပါတယ်။ ဒါကြောင့် User Input လုပ်လိုက်တဲ့ စာတွေအကုန်လုံးကို လိုချင်ရင် cin ကို အသုံးပြုလို့ရမှာမဟုတ်ပါဘူး။ အဲ့ဒီအစား getline() ဆိုတဲ့ function ကို အသုံးပြုရမှာဖြစ်ပါတယ်။

```
string fullName;
cout << "Type your full name : ";
getline(cin,fullName);
cout << "Your name is : " << fullName;
```

ဒီမှာဆိုရင် getline() ဆိုတဲ့ function က cin ထဲက string တွေကို တစ်ခုချင်းစီယူပြီးတော့ fullName ဆိုတဲ့ variable ထဲမှာ store ပြန်လုပ်ပါတယ်။

## 5.Operators

Operators တာတွေဆိုတာ သင်္ချာလက္ခဏာ ရပ်တွေပဲဖြစ်ပါတယ်။ သင်္ချာလက္ခဏာတွေရဲ့ Rules တွေအတိုင်း Variables တွေ Values တွေနဲ့ အလုပ်လုပ်ပါတယ်။ Operators အချို့တွေက ကိန်းဂဏန်းတွေကို Perform လုပ်တဲ့နေရာမှာပို အသုံးဝင်ပါတယ်။ Operators တွေကလည်း Programming Languages တွေတိုင်းမှာပါဝင်ပါတယ်။

```
int x = 300 + 200;
```

**C++ Operators**

Operators တွေကို အသုံးပြုပုံပေါ်မူတည်ပြီးတော့ အမျိုးအစားတွေ ခွဲခြားထားနိုင်ပါတယ်။

- **Arithmetic Operators**
- **Unary Operators**
- **Assignment Operators**
- **Comparison Operators**
- **Logical Operators**

### 5.1 Arithmetic Operators

Arithmetic Operators တွေကို အောက်ကဇယားမှာ ပြထားပေးပါတယ်။

| Symbols | Name | Description | Usage |
|---|---|---|---|
| + | Addition | | x + y |
| – | Subtraction | | x – y |
| * | Multiplication | | x * y |
| / | Division | စားလဒ်ကို ထုတ်ပေးပါတယ်။ | x / y |
| % | Modulus | စာကြွင်းကိုထုတ်ပေးပါတယ်။ | x % y |

ကျွန်တော်တို့တွေ Arithmetic Operators တွေကို variable တွေနဲ့ ပေါင်းပြီးရေးလိုက်လျှင် သင်္ချာမှာလို Expression တစ်ခုထွက်လာပါတယ်။ ဥပမာ –

| Math Expression | Expression in C++ |
|---|---|
| 2x + 3y - 6 | `(2*x) + (3*y) - 6` |

## 5.2 Unary Operators

C++ မှာပါဝင်တဲ့ Unary Operators တွေကို အောက်က ဇယားမှာ ဖော်ပြပေးထားပါတယ်။ ဒီ Operator တွေကို အသုံးပြုပြီး variable တွေရဲ့ တန်ဖိုးတွေကို အတိုးအလျော့လုပ်ပေးလို့ရပါတယ်။

| Symbols | Name | Description | Usage |
|---|---|---|---|
| ++ | Increment | လက်ရှိတန်ဖိုးကိုတိုးပေးသည်။ | ++x |
| -- | Decrement | လက်ရှိတန်ဖိုးကိုလျော့ပေးသည်။ | --x |

**Using Increment Operators**

```
#include <iostream>
using namespace std;

int main() {
  int x = 10;
  ++x;
  cout << x;
  return 0;
}
```

**Using Decrement Operators**

```
#include <iostream>
using namespace std;

int main() {
  int x = 10;
  --x;
  cout << x;
  return 0;
}
```

## 5.3 Assignment Operators

Assignment Operators တွေဆိုတာ **values တွေကနေ variables တွေကို Assign လုပ်ပေးပါတယ်**။ အဲ့ဒါတွေအပြင် Expression တွေထဲက ထွက်လာတဲ့ return values တွေကိုလည်း assign လုပ်ပေးပါတယ်။ ကျွန်တော်တို့ `=` assignment operator တစ်ခုကို ခဏခဏ အသုံးပြုပြီးဖြစ်ပါသည်။ အဲ့ဒါအပြင် တခြား Assignment Operator တစ်ချို့ရှိပါသေးတယ်။

| Operator | Example | Same As |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| *= | x *= 3 | x = x * 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| &= | x &= 3 | x = x & 3 |
| \|= | x \|= 3 | x = x \| 3 |
| ^= | x ^= 3 | x = x ^ 3 |

| >>= | x >>= 3 | x = x >> 3 |
|---|---|---|
| <<= | x <<= 3 | x = x << 3 |

## 5.4 Comparison

Comparison ဆိုတာ Value တစ်ခုနဲ့ တစ်ခုကို compare လုပ်ပြီး အဖြေရှာမယ် ပြီးတော့ Decisions တွေလုပ်တာဖြစ်ပါတယ်။ Programming တွေမှာ Decisions လုပ်ဖို့က true နဲ့ false ပဲလို ပါတယ်။ Decisions အကြောင်းကို Chapter 6 မှာရှင်းပြပေးထာပါတယ်။ Comparison လုပ်တာကတော့ true နဲ့ false နဲ့ကိုအဖြေရှာဖို့အတွက်ပါ။ true နဲ့ false နဲ့က Bol Data Type ပါ အဲ့တာကြောင့် Comparison လုပ်ထားတဲ့ values တွေကို Bol Data Type နဲ့ store လုပ်လို့ရပါတယ်။

**Comparison လုပ်တဲ့နေရာမှာ return values တွေက 0 နဲ့ 1 ပါ။ 0 ကို false နဲ့ 1 ကို true လို့သတ်မှတ်လို့ရပါတယ်။**

**Example**

```
int x = 10;
int y = 8;
cout << (x > y);// returns 1 (true) because 10 is greater than 8
```

Syntax တစ်ကြောင်းထဲမှာ Operator တွေ အများကြီးသုံးထားရင် () ခံပြီးသုံးပေးပါ။ Expression အရှည်ကြီးတွေမှာလည်းသုံးရပါမယ်။

အောက်က ဇယားမှာတော့ Comparison Formula နဲ့ Example တွေပြပေးထားပါတယ်ခင်ဗျာ။

| Operator | Name | Example |
|---|---|---|
| == | Equal to | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

```
int x = 3;
int y = 5;
cout << (x==y); // returns 0 (false) because 3 and 5 is not equal
cout << (x!=y); // returns 1 (true) because 3 and is  not equal
cout << (x>y); // return 0 (false) because 3 is not greater than 5
```

```
cout << (x<y); // return 1 (true) because 3 is less than 5
cout << (x>=y); // return 0 (false) because 3 is less than 5;
cout << (x<=y); // return 1 (false) because 3 is less than 5;
```

## 5.5 Logical Operators

Logical Operators တွေဆိုတာ Comparison Operators တွေနဲ့ return values (true or false) တူပါတယ်။ ဒါပေမဲ့ သူက Comparison နှစ်ခု (သို့) နှစ်ခုထက်ပိုပြီးတော့ကို နှိုင်းယှဉ်လို့ရပါတယ်။ Expression တွေကိုလည်းနှိုင်းယှဉ်လို့ရပါတယ်။ Logical Operators တွေက လက်တွေ့ Projects တွေရေးတဲ့နေရမှာ အလွန်အသုံးဝင်ပါတယ်။

| Operator | Name | Description | Example |
|---|---|---|---|
| && | Logical and | Returns true if both statements are true | x < 5 && x < 10 |
| \|\| | Logical or | Returns true if one of the statements is true | x < 5 \|\| x < 4 |
| ! | Logical not | Reverse the result, returns false if the result is true | !(x < 5 && x < 10) |

Example

```
int x = 4;
cout << (x < 5 && x < 10); // return 1 (true) because 4 is less than 5 and 10;
cout << (x < 5 && x > 10); // return 0 (false) because 4 is not greater than 10;
cout << !(x < 5 && x > 10); // return 1 (true) because using logical not while expression is false
```

## 6.Control Statements

Control Statment တွေဆိုတာ Programming တွေရဲ့ အခြေခံတွေပဲဖြစ်ပါတယ်။ ပြီးတော့ Algothrims တွေရဲ့ အခြေခံတည်ဆောက်ပုံလည်းဖြစ်ပါတယ်။ Control Statment တွေမှာ **Condition Statement** နဲ့ **Looping Statement** ဆိုပြီးတော့ နှစ်မျိုးရှိပါတယ်။

### 6.1 Condition Statement

Condition Statement တွေ ဆိုတာ ဘာဖြစ်လျှင် ဘာလုပ်ရမယ်၊ ဘာမလုပ်ရဘူးဆိုတာကို Computer Programming နဲ့ရေးသားခြင်းပဲဖြစ်ပါတယ်။ C++ ရဲ့ Condition Statement တွေကို အောက်မှာ ဖော်ပြပေးထားပါတယ်။

| | |
|---|---|
| if | Condition true ဖြစ်လျှင် သတ်မှတ်ထားတဲ့ Code တွေကို run ပါတယ်။ |
| else | Condition false ဖြစ်လျှင် သတ်မှတ်ထားတဲ့ Code တွေကို run ပါတယ်။ |
| else if | ပထမ Condition က false ဖြစ်နေလျှင် သတ်မှတ်ထားတဲ့ Code တွေကို run ပါတယ် |
| switch | if, else, else if, နဲ့အတူတူပါပဲ Condition မှန်တဲ့အခါမှ သတ်မှတ်ထားတဲ့ Code တွကို Run ပါတယ်။ |

**Using if(condition) & else**

```
bool isStudent = true;
if(isStudent){
    cout << "Go To School";
}
cout << "Life Finish";
```

ကျွန်တော်တို့အရှေ့အခန်းတွေမှာ Comparison Operators တွေကို အသုံးပြုပြီးတော့ true/false value တွေကို ဘယ်လိုထုတ်ရလဲပြောပြထားပါတယ်။ အဲ့ဒီ Comparison Operators တွေကို အသုံးပြုပြီးတော့လည်း Condition Statement တွေကို အသုံးပြုလို့ရပါတယ်။

**Example 6.1**

```
int age = 15;
if(age > = 18){
   cout << “You are adult”;
}else{
   cout << “You are boy”;
}
```

1. ကျွန်တော်က ဒီ Program မှာ အသက် 18 (သို့) အသက် 18 ထက်ကြီးတဲ့သူတွေကို "You are adult" ဆိုပြီးတော့ output ထုတ်ချင်ပါတယ်။ အသက် 18 အောက် ငယ်တဲ့သူတွေကို "You are boy" ဆိုပြီးတော့ output ထုတ်ချင်ပါတယ်။

2. ဒါဆိုရင် age ဆိုတဲ့ variable ကို 15 ဆိုပြီး Declare လုပ်လိုက်ပါတယ်။ အသက် 15 ပေါ့နော်။

3. ဒါဆိုရင် ကျွန်တော်တို့က 18 ထက် ငယ်လား၊ ကြီးလား၊ ညီနေလားကို ဆုံးဖြတ်ဖို့အတွက် if statement ကို သုံးရပါမယ်။

4. if statement ကိုသုံးဖို့အတွက် () ထဲမှာ Condition ကိုထည့်ရပါမယ်။ age >= 18 , age က 18 နဲ့ညီပြီးတော့ 18 ထက်ကြီးရင်ဆိုတဲ့ Condition တစ်ခုကိုရေးလိုက်ပါမယ်။ ပြီးရင် { } ထဲမှာ cout << "You are adult"; ဆိုပြီးတော့ output ထုတ်ပေးအောင်ရေးလိုက်ပါတယ်။

5. if မှာရှိတဲ့ Condition က false ဖြစ်နေလျှင် else ဆိုတဲ့ code block ထဲက ဟာတွေကို run ပါတယ်။ age >= 18 ဆိုပြီးရေးထားတဲ့ အတွက်ကြောင့် else မှာ age 18 အောက်ငယ်သွားတာအခါ run မှာဖြစ်ပါတယ်။ ဒါကြောင့် else { } ထဲမှာ cout << "You are boy"; ဆိုပြီး Output ထုတ်လိုက်ပါတယ်။

6. Program ကို Run ပြီးတော့ Output တွေကို ကြည့်နိုင်ပါတယ်။

7. age variable ရဲ့ values တွေကို 18, 19, .. စသည်ဖြင့် ချိန်းကြည့်ပြီး if, else တွေရဲ့ အလုပ်လုပ်ပုံကို နားလည်နိုင်ပါတယ်။

**Using else if(condition)**

**Example 6.2**

```
 int num1 = 50;
 int num2 = 50;
if (num1 > num2) {
   cout << num1 << " is greater than " << num2 << endl;
} else if (num1 < num2) {
   cout << num2 << " is greater than " << num1 << endl;
} else {
   cout << num1 << " and " << num2 << " are equal" << endl;
}
```

else if ဆိုတဲ့ statement အရှေ့မှာရှိတဲ့ condition က false ဖြစ်နေလျှင်.... ဒါကော ဖြစ်နိုင်သေးလား ဆိုတာ ထပ်ပြီး စစ်တာဖြစ်ပါတယ်။ Example Code မှာဆိုရင် num1 > num2 က false ဖြစ်နေလျှင် num1 < num2 ကောဖြစ်နိုင်လားလို့ else if() ကို သုံးပြီး ထပ်စစ်တာဖြစ်ပါတယ်။ နောက်ဆုံး else statement ကတော့ အပေါ်က statement နှစ်ခုလုံးမမှန်ရင် else {} ထဲက ဟာတွေ run ပါတယ်။

Using switch(expression)

switch statement ကတော့ expression တစ်ခု ကို case: ထဲမှာရှိတဲ့ value တွေနဲ့ Compare လုပ်ပါတယ်။ case: ထဲက value တစ်ခုခုနဲ့ညီလျှင် သူ့အောက်မှာရှိတဲ့ Code တွေ ကို run ပါတယ်။ break; ဆိုတာကတော့ switch statement ကို ထပ်မစစ်တော့အောင် ရပ်လိုက်တာဖြစ်ပါတယ်။ case: ထဲက value တစ်ခုနဲ့မှ မညီလျှင် default: ဆိုတဲ့ code တွေကို run ပါတယ်။

## Syntax

```
    switch (expression){
         case x:
          //code block
         break;
         case y:
          //code block
         break;
         default:
          //code block
    }
```

## Example 6.3

```
 int choice;
cout << "Select an option (1-3): ";
cin >> choice;
 switch (choice) {
    case 1:
       cout << "You chose option 1.\n";
       break;
    case 2:
       cout << "You chose option 2.\n";
       break;
    case 3:
       cout << "You chose option 3.\n";
       break;
    default:
       cout << "Invalid choice.\n";
       break;
}
```

ဒီ Code မှာဆိုရင် User Input ကို အသုံးပြုပြီးတော့ ဂဏန်း 1 to 3 အထိ ရွေးခိုင်းပါမယ်။ ပြီးလျှင် switch case syntax ကိုအသုံးပြုပြီးတော့ User က ဘယ်ဂဏန်းကိုရွေးလည်းဆိုတာကို Output

ထုတ်ပေးမှာဖြစ်ပါတယ်။ User က 1 to 3 ထဲမှာ မပါတဲ့ ဂဏန်းကိုရွေးလိုက်လျှင် "Inavalid Choice" ဆိုပြီးတော့ Output ထုတ်ပေးပါတယ်။

## 6.2 Looping Statement

ကျွန်တော်တို့က Program တွေကို ထပ်တလဲလဲလုပ်ဆောင်လုပ်ရမယ်ဆိုလျှင် Looping Statement တွေကို အသုံးပြုရမှာပဲဖြစ်ပါတယ်။ Loops တွေကို အသုံးပြုခြင်းသည် ထပ်ထပ်ခါ လုပ်ရမည့်ကိစ္စများကို အချိန်ကုန်သက်သာစေပြီးတော့ Errors တွေကိုလည်းလျှော့ချနိုင်မှာဖြစ်ပါတယ်။ C++ Language မှာဆိုရင် အဓိကအားဖြင့် Looping Statement သုံးမျိုးရှိပါတယ်။

| Looping Statement | Usage |
|---|---|
| for | Used to execute a block of code a fixed number of times, often used for iterating over a sequence of elements such as an array or list |
| while | Used to execute a block of code repeatedly as long as a specific condition is met, often used for situations where the number of iterations is unknown |
| do-while | Similar to a while loop, but guaranteed to execute the block of code at least once before checking the condition, often used for situations where the block of code must be executed at least once |

### Using for loop

#### Syntax

```
for(statement 1;statement 2; statement 3) {
 // code block to be executed
}
```

for loop ကိုသုံးဖို့အတွက် () ထဲမှာ  Statement သုံးခုရေးရပါမယ်။

Statement 1 : for loop မှာ တစ်ကြိမ်ထဲ run မဲ့ Code ကိုရေးရပါမယ်။

Statement 2: loop ရဲ့ Condition ကိုရေးရပါမယ်။ ဘာကြောင့် Loop ပတ်ရမလဲ ဆိုတာကိုပေါ့။

Statement 3: loop တစ်ကြိမ်ပတ်ပြီးတိုင်းမှာ Statement 3 ကို run ပါတယ်။

#### Example 6.4

```
for(int i = 1; i <= 10; i++) {
  cout << i << "\n";
}
```

for loop ကိုသုံးတဲ့အခါမှာ ဘယ်နကြိမ် loop ပတ်မလဲဆိုတာကိုသိဖို့လိုပါတယ်။ for loop ရဲ့ Statement တွေက Integer တွေနဲ့ပဲအလုပ်လုပ်ပါတယ်။ Example 6.4 မှာဆိုရင် –

statement 1 က loop ရဲ့ initial value အဖြစ် (int i = 1) ဆိုပြီးတော့ သတ်မှတ်လိုက်ပါတယ်။

statement 2 က တော့ i ရဲ့ value က 10 မရောက်ခင်ထိ loop ပတ်မယ်ဆိုတဲ့ condition ကို ဆိုလိုပါတယ်။ (i <= 10;)

statement 3 ကတော့ loop တစ်ခါပတ်ပြီးတိုင်း i ရဲ့ value ကို 1 ပေါင်းပေးပါတယ်။ i++;

## Using while Loop

while loop က တော့ () ရှိတဲ့ condition က true ဖြစ်နေသည့်တိုင်အောင် loop ပတ်နေမှာပဲဖြစ်ပါတယ်။ ဒါကြောင့် while loop ကို အသုံးပြုဖို့အတွက် true/false boolean တွေကို အသုံးပြုရမှာပဲဖြစ်ပါတယ်။

## Syntax

```
while (condition) {
    // code block to be executed
}
```

## Example 6.5

```
int i = 0;
while (i < 10) {
  cout << i << "\n";
  i++;
}
```

1. int i = 0; ဆိုပြီးတော့ variable တစ်ခုကိုကြေငြာလိုက်ပါတယ်။

2. ပြီးတော့ i ရဲ့ value က 10 အောက်ငယ်မယ်ဆိုလျှင် while အောက်က Code တွေကို Loop ပတ်ပြီး run ပါတယ်။ while(i < 10 )

3. cout << i << “\n”, i ရဲ့ value ကို output ထုတ်လိုက်ပါတယ်။

4. i++ ကတော့ i ရဲ့ value ကို 1 ပေါင်းပေးတာဖြစ်ပါတယ်။ code block တစ်ခုလုံးပြီးသွားရင် loop ပြန်ပတ်ပါတယ်။ ဒီတစ်ချိန်မှာတော့ i ရဲ့ value က 1 တိုးသွားပါပြီး။ ဒီအတိုင်းပဲ ထပ်ထပ်ခါ loop ပတ်ပြီးတော့ i ရဲ့ value က 10 ရောက်သွားတဲ့အချိန်မှာ loop ပတ်တာရပ်သွားပါလိမ့်မယ်။

### Using do while Loop

do while loop က while loop ရဲ့အမျိုးအစားတစ်ခုပါပဲ၊ while loop မှာတော့ condition ကို အရင်စစ်ပြီးတော့မှာ code block တွေကို run ပါတယ်။ do while loop မှာတော့ သူနဲ့ပြောင်းပြန်ပါ code block တွေကို အရင် run ပြီးတော့မှ condition တွေကို စစ်ပါတယ်။

### Syntax

```
do{
        //code block
}while(condition);
```

**ဒါကြောင့် do while loop မှာတော့ သူရဲ့ conditon က false ဖြစ်နေရင်တောင် do ဆိုတဲ့ code block ထဲမှာရှိတဲ့ code တွေကို တစ်ကြိမ်တော့ run ပါတယ်။**

### Example 6.6

```
int i = 0;
do {
  cout << i << "\n";
  i++;
}while (i < 10);
```

**Example 6.6** ကလည်း while loop ရဲ့ **Example 6.5** နဲ့အတူတူပါပဲ။ ဒါပေမဲ့ do while loop ရဲ့ ပုံစံအတိုင်း code blocks တွေကို အရင် run ပြီးမှ condition ကို စစ်ပါတယ်။ loop ပတ်တဲ့နေရမှာ condition ကို false ဖြစ်အောင် လုပ်ရပါမယ်။ မဟုတ်ရင် loop က infinity ဖြစ်နေပါလိမ့်မယ်။

# 7.Functions

Functions တွေဆိုတာ Program တစ်ခုတည်ဆောက်ရာမှာ အရေးပါပြီး အခြေခံကျတဲ့ subprogram တွေပဲဖြစ်ပါတယ်။ Program တစ်ခုမှာ အနည်းဆုံး function တစ်ခုတော့ ပါရပါမယ်။ C++ program မှာဆိုရင် အဓိက ဝင်ပေါက်၊ ထွက်ပေါက်ဖြစ်တဲ့ main() function ပါပဲ။ function တွေက call ခေါ်မှ အဲ့ဒီ function ထဲမှာရှိတဲ့ code block ကို အလုပ်စလုပ်ပါတယ်။ ချွင်းချက်တစ်ခုကတော့ main() function ဆိုတဲ့ ဟာကို Call ခေါ်လို့ရမှမဟုတ်ပါဘူး။

ထပ်ခါတစ်လဲလဲရေးရမယ့် Code တွေကို function ထဲ ထည့်ရေးခြင်းဖြင့် Code တွေ Complex ဖြစ်တာတွေကို လျော့ချနိုင်ပါတယ်။ ဒါကြောင့် main() function ထဲမှာ code တွေ ရှုပ်လာပြီဆိုရင် function တွေ အသစ်ခေါ်ပြီး ရေးလို့ရပါတယ်။

```
#include <iostream>
 using namespace std;
 void callMe(){
    cout << “Hey, I am function \n”;
 }
  int main(){
     callMe();
       return 0;
  }
```

function တွေကို ခဏခဏ ခေါ်သုံးလို့လဲရပါတယ်။

```
int main(){
 callMe();
 callMe();
}
 //output
 //Hey, I am function
 //Hey, I am function
```

## 7.1 Defining  Functions

Function တစ်ခုကို Declare လုပ်တော့မယ်ဆိုလျှင် အရှေ့မှာ void ဆိုတာထဲ့ရပါမယ်။ void ဆိုတာက return value မရှိဘူးဆိုတာကိုပြောတာပါ။ ပြီးရင် Function Name ပေးရပါမယ်။ function name ပေးတဲ့ နေရာမှာ variable name တွေပေးတဲ့အခါ သုံးတဲ့ rule တွေနဲ့ အတူတူပါပဲ C++ ရဲ့ Keyword တွေနဲ့ ညှိလို့မရပါဘူး၊ Name တွေမှာ Space ခြားလို့မရဘူး။ ဂဏန်းတွေ၊ Special Character တွေနဲ့ စလို့မရပါဘူး။ Function Name ရဲ့ နောက်မှာ () ရေးရပါမယ်။ Code တွေကိုတော့ { } ထဲမှာရေးရပါမယ်။ function တွေ Declare လုပ်တဲ့နေရာမှာ main () function ရဲ့အပေါ်ပိုင်းမှာပဲရေးရပါမယ်။

### Example 7.1

```
void myFunction(){

  cout << " Hello World From Functions";
}
```

## 7.2 Calling Functions

Function တွေကို သတ်မှတ်ယုံနဲ့ Compiler ထဲမှာ Execute မလုပ်ပါဘူး Function တွေကို Calling Function တွေမှာခေါ်မှ အလုပ်လုပ်ပါမယ်။ Function တွေကို ကိုယ်တိုင်ရေးပြီး Call လုပ်တာရှိသလို၊ Library တွေထဲကနေ Call လုပ်တာလဲရှိပါတယ်။ ဘယ်ကနေပဲ Call လုပ်လုပ် Function တွေ Call လုပ်တဲ့ ပုံစံက အတူတူပါပဲ။ Function Name ကိုရေးပြီး (); လို့ရေးပေးရပါမယ်။ ဒါမှ Function ထဲက Code တွေကို execute လုပ်မှာဖြစ်ပါတယ်။

### Example 7.2

```
int main(){

   myFunction();

return 0;

}
```

## 7.3 Passing Parameters to Functions

Passing