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

Function တွေမှာ Parameters တွေခံပြီးတော့ Data တွေ၊ Value တွေကို Pass လုပ်လို့ရပါတယ်။ Parameter ဆိုတာ () ထဲမှာ Variable Name တွေထည့်ထားတာကိုဆိုလိုတာပါ။ Parameters တွေကို Arguments လို့လဲခေါ်ကြပါတယ်။

```
void myFunction(int a, int b)
```

Example 7.3

```
void SumCalculate(int a, int b){
cout << (a + b);
}
int main(){
 SumCalculate(25,40);  //65
 return 0;
}
```

7.4 Returning values from functions

Function တစ်ခုမှ value တစ်ခုကို return လုပ်လို့ရပါတယ်။ Value ဆိုတော့ data type တစ်ခုခုသုံးရမှာပေါ့။ Return value ပါတဲ့ function တွေမှာ void အစား return value ရဲ့ data type ကိုပဲသုံးရပါမယ်။ ဥပမာ - Value က ကိန်းဂဏန်းဖြစ်တယ်ဆိုရင် Number data types တွေကို သုံးရမယ်၊ စာသားဖြစ်မယ်ဆိုရင် String data type ကိုသုံးရပါမယ်။ ပြီးရင်တော့ return ဆိုတဲ့ keyword ကို နောက်ဆုံး code line မှာ ထည့်ပေးဖို့လိုပါမယ်။

```
int Compute(int num1, int num2) {
    int result =  num1 + num2;
    return result;
}
.....
Compute(5,3) // 8
.....
```

Example 7.4

```
#include <iostream>
#include <string>
using namespace std;
string FullName(string firstname,string lastname){
    return firstname + lastname;
}
int main(){
  cout << FullName("Taylor", "Swift"); // Taylor Swift
  return 0;
}
```

8.Structures

Structures ဆိုတာကတော့ တစ်ခုနဲ့တစ်ခုသက်ဆိုင်တဲ့ Variable တွေကို Group တစ်ခုအနေနဲ့ စုစည်းထားခြင်းဖြစ်ပါတယ်။ Structure ထဲမှာရှိတဲ့ Variable တွေကို Member လို့သတ်မှတ်နိုင်ပါတယ်။ Structure တစ်ခုကို ဖန်တီးဖို့ struct ဆိုတဲ့ Keyword ကို အသုံးပြုရပါမယ်။

8.1 Creating Structures

Structure တစ်ခုကို ဖန်တီးရန် struct ဆိုတဲ့ keyword ကို ဦးစွာ ပထမဆုံးရေးရပါမယ်။ သူ့ရဲ့ Member တွေကိုတော့ curly braces { } အထဲမှာရေးရပါမယ်။ Member တွေကိုကြေငြာတဲ့ပုံစံက variable တွေနဲ့ပုံစံအတူတူပါပဲ။ ပြီးရင်တော့ struct အတွက် variable name တစ်ခုပေးရပါမယ်။ variable name ကိုတော့ နောက်ဆုံးမှာရေးရပါမယ်။ ပြီးရင်တော့ ; ပေါ့။

```
    struct{
          string name;
          int age;
          double height;
    } person;
```

8.2 Accessing Structures Member

Structures တွေရဲ့ Member တွေကို Access လုပ်ဖို့ဆိုရင် Structures variable name ရဲ့ နောက်မှာ (.) dot syntax ကို သုံးပြီးတော့ member ရဲ့ variable name ကို ခေါ်သုံးယုံပါပဲ။

```
struct {
 string name;
 int age;
 double height;
} person;

//Assing values to members of person structure
person.name = 'Mg Mg'
person.age = 18;
person.height = 5.7;

// Print members of person structure
cout << "Name : " << person.name << "\n";
cout << "Age : " << person.age << "\n";
cout << "Height (ft) :" << person.height << "\n";
```

8.3 One Structure in Multiple Variables

Variable Name ရဲ့ နောက်မှာ (,) comma ကို အသုံးပြုပြီး Variable တွေများစွာကို Define လုပ်လို့ရပါတယ်။

```
struct {
 string name;
 int age;
 double height;
} person1, person2, person3, person4;


// Assign value to member;
person1.name = "Mg Mg";
person2.name = "Hla Hla";
person3.name = "U Ba";

person1.age = 13;
person2.age = 15;
person3.age = 50;

person1.height = 4;
person2.height = 5.5;
person3.height = 5.2;

// Print the structure members
cout << person1.name << endl;
cout << person2.name << endl;
cout << person3.name << endl;
```