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။

Author : Thura Lin Htut

5.Operators

Operators တာတွေဆိုတာ သင်္ချာလက္ခဏာ ရုပ်တွေပဲဖြစ်ပါတယ်။ 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 ပဲလိုပါတယ်။ 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 ဟာတွေ့ run ပါတယ်။

Using switch(expression)

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

Author : Thura Lin Htut

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။