C++ & OOP

COURSE REQUIREMENT

វត្ថុបំណងនៃមេរៀន

នៅក្នុងមេរៀននេះ យើងនឹងផ្តោតសំខាន់ទៅលើការសិក្សាស្វែងយល់ពីចំណុចសំខាន់ៗចំនួន ៥ គឺ Variable, Control Structure, Data Structure, Function រួមទាំង Class & Object ផងដែរ ។

- VARIABLE
- CONTROL STRUCTURE
- DATA STRUCTURE
- FUNCTION
- CLASS & OBJECT

VARIABLE

តើអ្វីទៅជា Variable?

- ជាឈ្មោះតំណាងអោយការរក្សារទុកទិន្នន័យបណ្តោះអាសន្ននៅក្នុង memory(RAM)ដែលទិន្នន័យទាំងនោះនឹងត្រូវបានបាត់បង់ទៅវិញ នៅពេលកម្មវិធីត្រូវបានបិទ។
- Variable ជា block នៅក្នុង memory ដែលយើងអាចដាក់តម្លៃចូល និង អាចទាញតម្លៃនោះមកប្រើប្រាស់វិញបាន។

របៀបនៃការបង្កើត variable

```
Datatype varName;
```

បើសិនជា variable ច្រើនមាន Datatype ដូចៗគ្នាអាចបង្កើត

```
Datatype varName1, varName2, ...;
```

DataType ជាការកំណត់ប្រភេទទិន្នន័យទៅឱ្យ Variable អាចទទួលតម្លៃជាអ្វី(ឧ. លេខ, តួអក្សរ, ពាក្យ, boolean,...)

របៀបនៃការដាក់ឈ្មោះ variable

- អាចជាអក្សរពី a-z ឬ A-Z និង 0-9
- មិនអាចផ្តើមដោយលេខ
- មិនត្រូវមានផ្នែកសញ្ញាផ្សេងដូចជា *, -, /, +, (,), &, #,...
- មិនត្រូវមានឈ្មោះជាន់ keyword ដូចជា int, float, return,...
- មិនអាចដកឃ្លា តែអាចជំនួសដោយសញ្ញា _ បាន។

- គ្រប់ ឈ្មោះ variable មានលក្ខណៈ Case Sensitive មានន័យថាវាចែកដាច់រវាងអក្សរតូច និងអក្សរធំ។

  ឧទាហរណ៍៖

```
int level;
Level = 3; //error ព្រោះមិនមាន variable ឈ្មោះ Level ទេ
```

- មិនត្រូវមានឈ្មោះជាន់គ្នា(ក្នុង block តែមួយ)

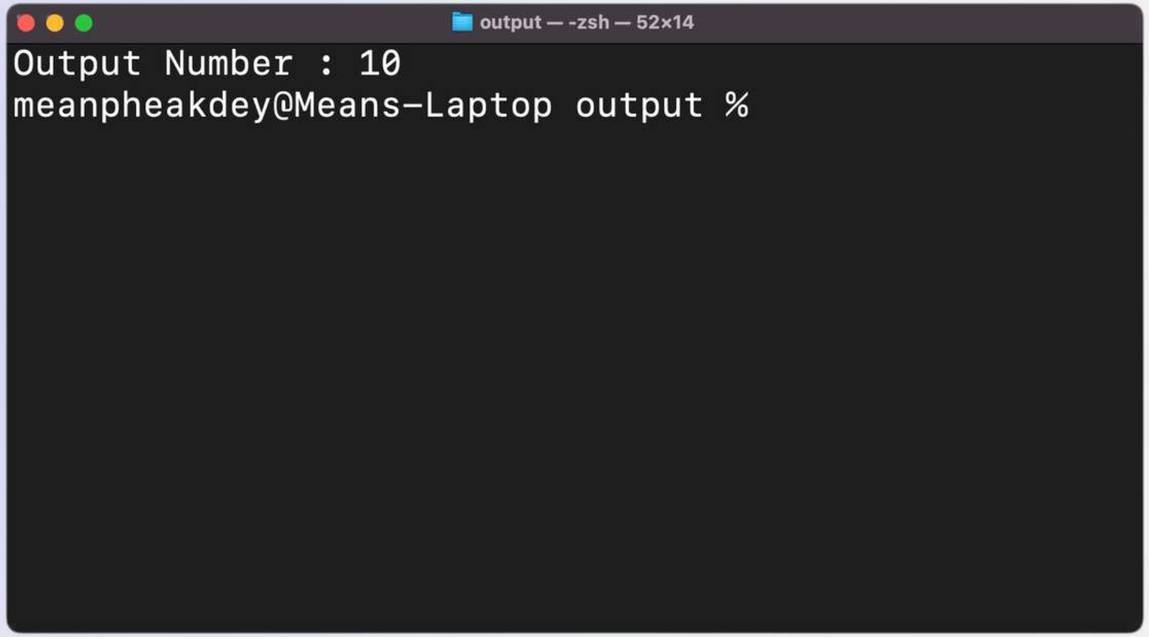
  ឧទាហរណ៍៖

```
int weight;
float weight; //error ព្រោះបានបង្កើតម្តងហើយ
```

ឧទាហរណ៍

```cpp
#include <iostream>

using namespace std;

int main(){
    int number;
    number = 10;

    cout << "Output Number : " << number << endl;

    return 0;
}
```

លទ្ធផល



DATATYPE

- នៅក្នុងភាសា C/C++ Data Type គឺជាច្បាប់ឬ ដែនកំណត់សម្រាប់បញ្ជាក់ន័យឱ្យ variable ក្នុងការទទួលយកទិន្នន័យជាប្រភេទជាក់លាក់ណាមួយសម្រាប់រក្សាទុកបណ្តោះអាសន្នក្នុង memory។

DATATYPE	មុខងារ
int	លេខគត់
float, double	លេខទសភាគ
char	តួអក្សរទោល
string	តួអក្សរច្រើន
bool	boolean

BASIC OUTPUT

តើអ្វីទៅជា Output នៅក្នុង C/C++?

- Output ជាលទ្ធផលដែលត្រូវបានបង្ហាញតាមរយៈ Screen Console ។ អ្នកចាំបាច់ត្រូវប្រើប្រាស់ Standard Output Stream (cout) ។ cout គឺយើងប្រើប្រាស់ជាមួយ insertion operator (<<) ។

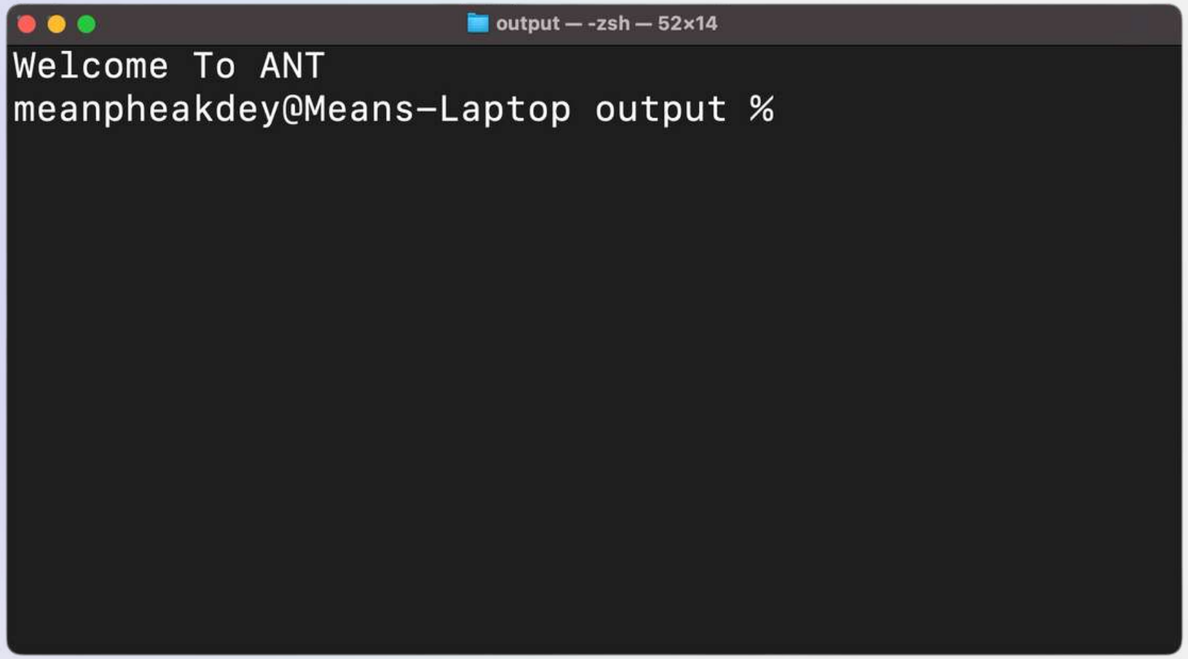
  Syntax :

```
cout << varName;
```

ឧទាហរណ៍

```cpp
#include <iostream>

using namespace std;

int main(){

    cout << "Welcome To ANT" << endl;

    return 0;
}
```

លទ្ធផល



BASIC INPUT

តើអ្វីទៅជា Input នៅក្នុង C/C++?

- Input គឺសំដៅទៅលើការ បញ្ចូលទិន្នន័យឬតម្លៃផ្សេងៗដែល User បានបញ្ចូលបោះទៅកាន់ variable សម្រាប់ផ្តល់ចេញជាលទ្ធផល ឬ សម្រាប់គណនាអ្វីមួយ។ អ្នកចាំបាច់ត្រូវប្រើប្រាស់ Standard Input (cin)។ cin ប្រើប្រាស់ជាមួយ extraction operator (>>) ។

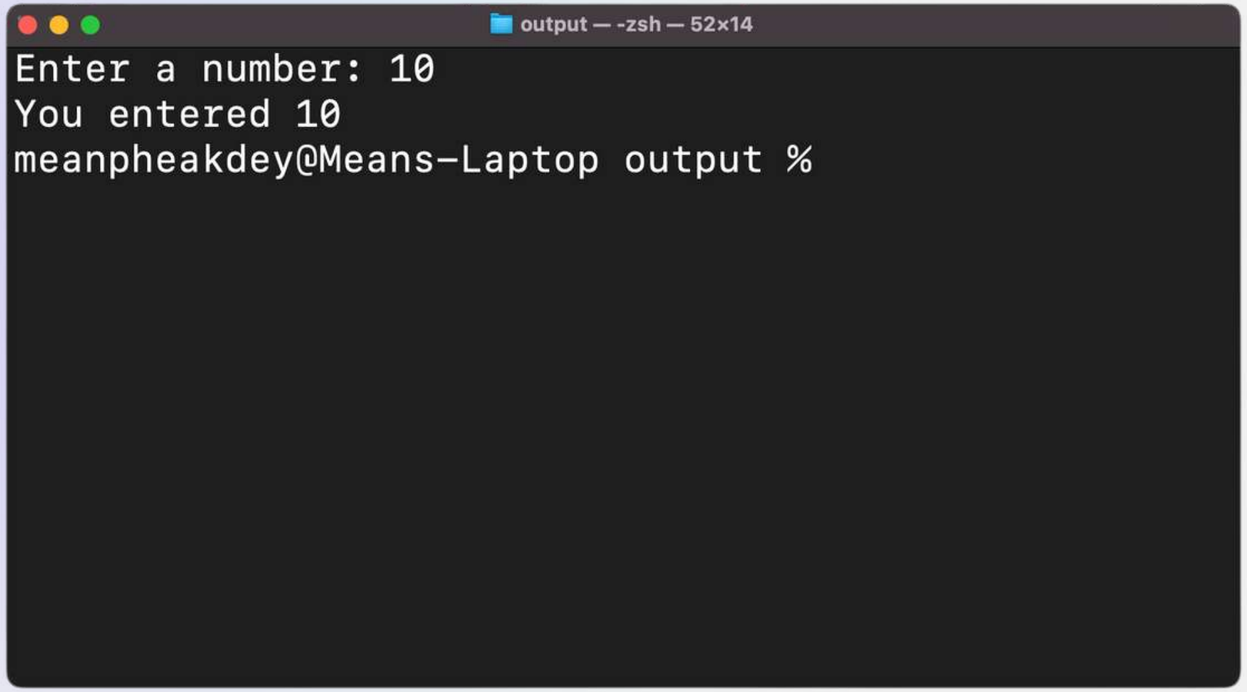
  Syntax :

```
cin >> varName;
```

ឧទាហរណ៍

```cpp
#include <iostream>

using namespace std;

int main(){
    int number;

    cout << "Enter a number: ";
    cin >> number;

    cout << "You entered " << number << endl;

    return 0;
}
```

លទ្ធផល



EXERCISES

1. ចូរបង្កើតចំនួន ៧ ដោយបញ្ចូលតម្លៃតាម Keyboard និង បង្ហាញទិន្នន័យនោះមកលើអេក្រង់វិញ :

```
int id, age;
char gender;
char name[20], address[30], email[40];
double score;
```

2. ចូរសរសេរកម្មវិធីដើម្បីគ្រប់គ្រងទិន្នន័យរបស់ Employee ដែលមាន field ដូចជា id(int), name(string), rate(float), hours(float) និង salary(float)។ បញ្ចូលទិន្នន័យទាំងអស់នោះតាម Keyboard, គណនា salary និង បង្ហាញទិន្នន័យទាំងអស់នោះមកលើអេក្រង់វិញ។

3. Write a C++ program to find the Area and Perimeter of a Rectangle.

4. Write a C++ program to convert temperature in Celsius to Fahrenheit.

5. Write a program in C++ that converts kilometers per hour to miles per hour.

6. Write a program in C++ to check whether a number is positive, negative or zero.

CONTROL STRUCTURE

តើអ្វីទៅជា Control Structure?

- Control Structure ត្រូវបានគេប្រើសម្រាប់ត្រួតពិនិត្យទៅលើដំណើរ ការរបស់កម្មវិធី នៅពេលដែលកម្មវិធីកំពុងដំណើរការ។ ជាទូទៅ ការត្រួតពិនិត្យលក្ខខណ្ឌនៅ ក្នុងកម្មវិធីមួយគឺចង់ដឹងពីសកម្មភាពដែលវាបានកើតឡើង។ នៅក្នុងការត្រួតពិនិត្យលក្ខខណ្ឌ នៅក្នុងភាសា C/C++ មានតែពីរប្រភេទគត់គឺៈ លក្ខខណ្ឌពិត (True) និង លក្ខខណ្ឌមិនពិត (False)។

IF STATEMENT

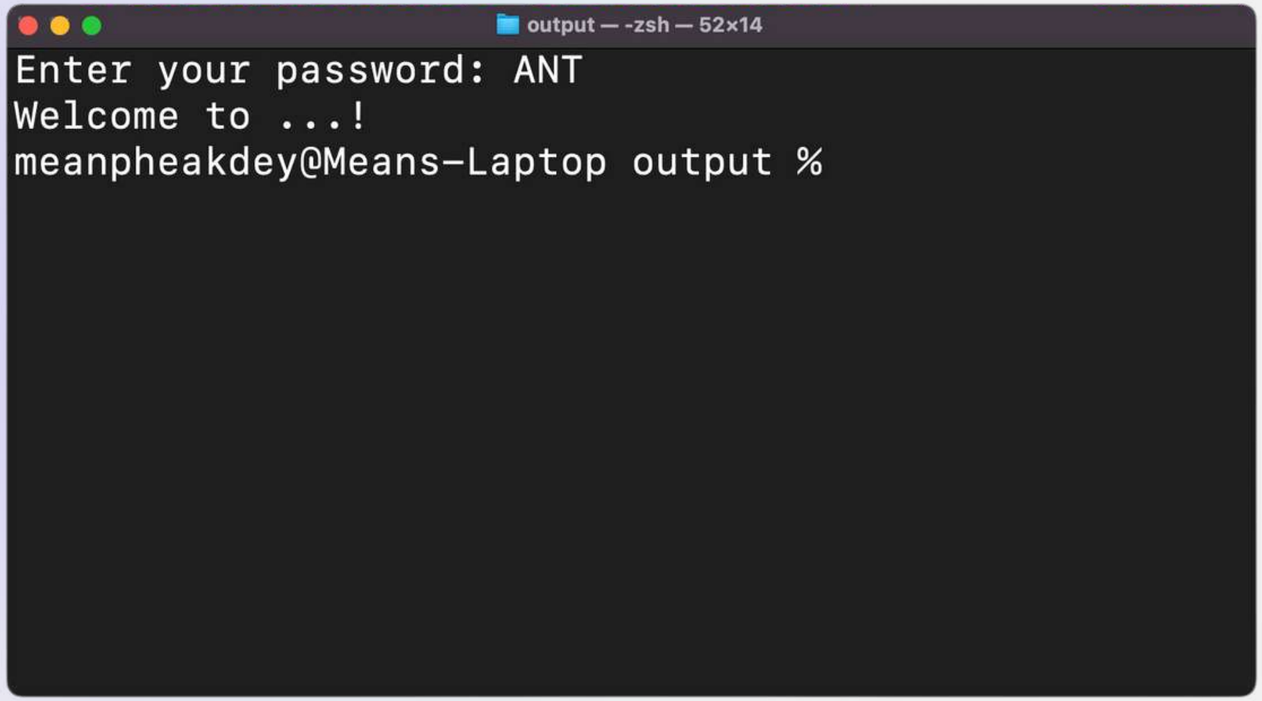
- ពាក្យគន្លឹះ if អនុញ្ញាតអោយជ្រើសរើសការងារមួយក្នុងចំណោមការងារពីរឬច្រើន មក ធ្វើប្រតិបត្តិការណ៍ដោយអាស្រ័យទៅលើលក្ខខណ្ឌ ដែលអ្នកសរសេរកម្មវិធីបានកំណត់អោយ។

  ទម្រង់ទូទៅរបស់ if ៖

```
if (expression) {
      statement;
}
```

ឧទាហរណ៍ លទ្ធផល

```cpp
int main(){
    string password;

    cout << "Enter your password: ";
    getline(cin, password);

    if (password == "ANT"){
        cout << "Welcome to ...!" << endl;
    }

    return 0;
}
```



IF..ELSE STATEMENT

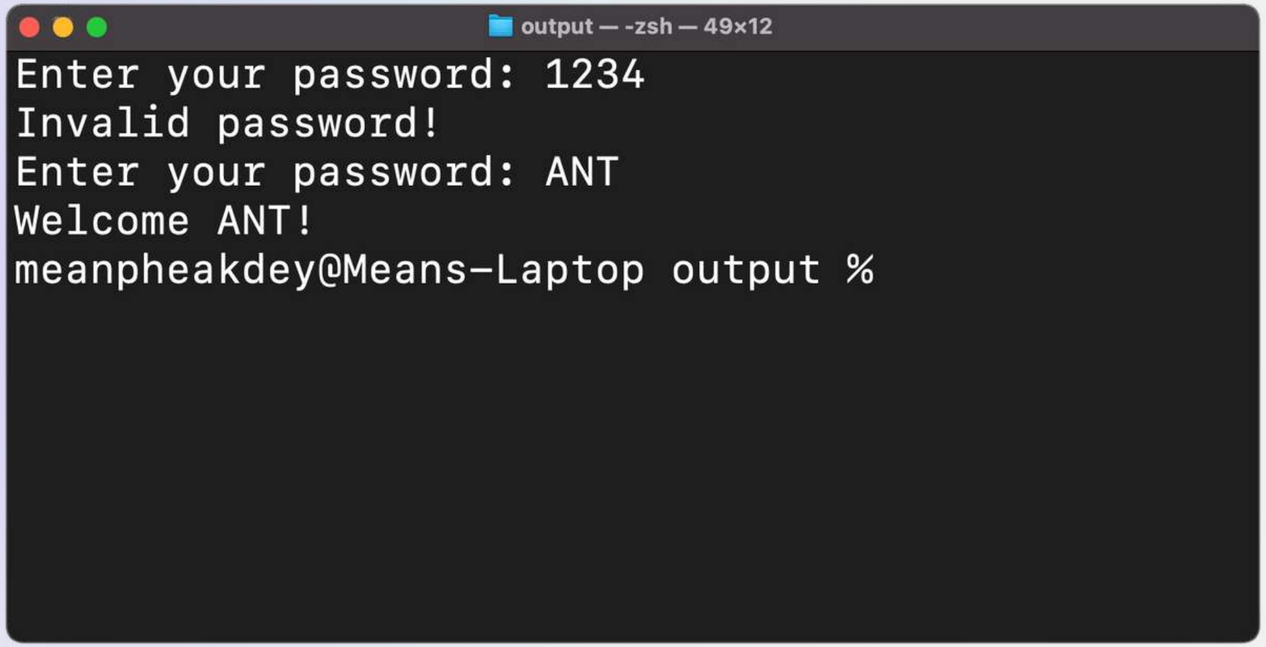
- if នឹងត្រួតពិនិត្យលក្ខខណ្ឌរបស់ expression បើ expression ពិត នោះវានឹងអនុវត្តន៍ statement(s) ដែលនៅក្រោមបញ្ជារបស់ if ក៏ប៉ុន្តែបើលក្ខខណ្ឌរបស់ if មិនពិតទេនោះវានឹង អនុវត្តន៍ statement(s) ដែលនៅក្រោមបញ្ជារបស់ else។

  ទម្រង់ទូទៅរបស់ if..else ៖

```
if (expression) {
        statement1;
} else {
        statement2;
}
```

ឧទាហរណ៍ លទ្ធផល

```cpp
int main(){;
    string password;

    cout << "Enter your password: ";
    getline(cin, password);

    if (password == "ANT"){
        cout << "Welcome ANT!" << endl;
    }else{
        cout << "Invalid password!" << endl;
    }

    return 0;
}
```



IF..ELSE-IF STATEMENT

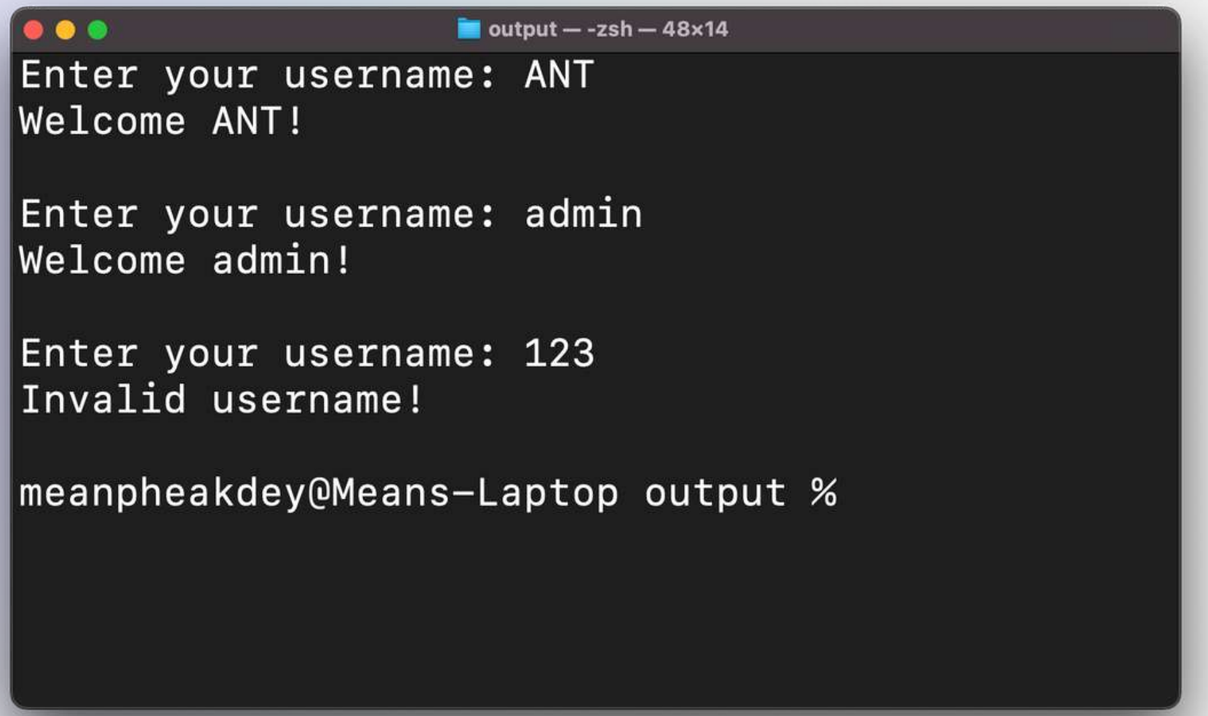
- នៅក្នុងជីវិតជាក់ស្តែង យើងត្រូវការធ្វើការវិនិច្ឆ័យទៅលើរឿងជាច្រើន ដែលមានជាបន្ត បន្ទាប់គ្នា។ ហើយ if...else-if statement គឺវាមានលទ្ធភាពអាចដោះស្រាយបញ្ហាទាំងអស់នោះបាន ដោយងាយស្រួល។

  ទម្រង់ទូទៅរបស់ if ៖

```
if (expression1) {
        statement1;
} else if (expression2){
        statement2;
} else {
        statement3;
}
```

ឧទាហរណ៍ លទ្ធផល

```cpp
int main(){;
    string username;

    cout << "Enter your username: ";
    getline(cin, username);

    if (username == "ANT"){
        cout << "Welcome ANT!" << endl;
    }else if(username == "admin"){
        cout << "Welcome admin!" << endl;
    }else{
        cout << "Invalid username!" << endl;
    }

    return 0;
}
```



LOOP STATEMENTS

- Loop statement ត្រូវបានប្រើប្រាស់យ៉ាងចាំបាច់ក្នុងការសរសេរកម្មវិធី។ តើ អ្វីជា Loop?
- Loop គឺជាដំណើរការនៃការអនុវត្តការងារណាមួយដដែលៗជាច្រើនដង រហូតដល់ជួប លក្ខខណ្ឌដែលអ្នកសរសេរកម្មវិធីបានកំណត់អោយទើបបញ្ឈប់ បើពុំដូច្នោះទេវានឹងដំណើរការ រហូតគ្មានទីបញ្ចប់។ រង្វិលជុំនៅក្នុងភាសា C មាន៣ប្រភេទគឺ: រង្វិលជុំទំរង់ while រង្វិលជុំទម្រង់ do...while និង រង្វិលជុំទំរង់ for។
  - ឧបមាថា បើយើងចង់ធ្វើប្រមាណវិធផលបូកស្វ៊ីត(sequence number)ពី 1 ដល់ 100 ឬ ពី 1 ដល់ 10000 ឬលើសពីនេះ តើយើងត្រូវតែសរសេរតម្លៃលេខពី 1+2+...+10000 ឬយ៉ាងណា?

ចម្លើយគឺពិតជាមិនអាចទៅរួចទេ បើសិនជាយើងជាអ្នកសរសេរកម្មវិធីដែលយល់ដឹងពីការប្រើ Loop Statement

FOR LOOP STATEMENT

តើអ្វីទៅជា for Loop?

- for Loop ជាប្រភេទ Loop មួយប្រភេទដែលយើងដឹងពីចំនួនជុំជាក់លាក់ណាមួយត្រូវកំណត់ទៅកាន់ Statements ឱ្យធ្វើម្តងហើយម្តងទៀត។

  វង្វិលជុំ for loop មានទម្រង់ទូទៅដូចខាងក្រោម ៖

```
for(start ; condition; go to end){
        statement(s);
}
```

ពាក្យបង្ហែកទេសក្នុង for loop

ទម្រង់ទូទៅរបស់ for loop ៖

```
for(start ; condition; go to end){
        statement(s);
}
```

- Start: ជាកន្លែងសម្រាប់កំណត់តម្លៃចាប់ផ្តើមរបស់ for ជាទូទៅគឺការតាង int variable ដោយផ្តល់តម្លៃជាស្រេច។
- Condition: ជាកន្លែងកំណត់លក្ខខណ្ឌទៅកាន់ for ដោយធៀបទៅនឹងតម្លៃបញ្ចប់ (បើពិតអនុវត្ត Statements)។
- Go to end: មានន័យថាវានឹងផ្លាស់ប្តូរតម្លៃ start ឱ្យខិតទៅរកតម្លៃបញ្ចប់(អាច កើន ឬថយ)បើសិនជាជួបលក្ខខណ្ឌពិត។

ឧទាហរណ៍ លទ្ធផល

```cpp
int main()
{
    int n;
    int sn = 0;

    cout << "Enter n: ";
    cin >> n;

    // Loop to calculate the sum of the series: 1 + 2 + 3 + ... + n
    for (int i = 1; i <= n; i++)
    {
        sn += i; // Accumulate the current value of i to the sum (sn)
    }
    cout << "Sn = " << sn << endl;

    return 0;
}
```

```
output — -zsh — 48×12
Enter n: 5
Sn = 15
meanpheakdey@Means-Laptop output %
```

WHILE LOOP STATEMENT

តើអ្វីទៅជា while Loop?

- while Loop ជាប្រភេទ Loop មួយប្រភេទដែលយើងមិនត្រូវការតម្លៃចាប់ផ្តើម។ មានន័យថា វាត្រូវការតែលទ្ធផល Boolean មួយប៉ុណ្ណោះ។

  រង្វិលជុំ for loop មានទម្រង់ទូទៅដូចខាងក្រោម ៖

```
while(Boolean){
    statement(s);
}
```

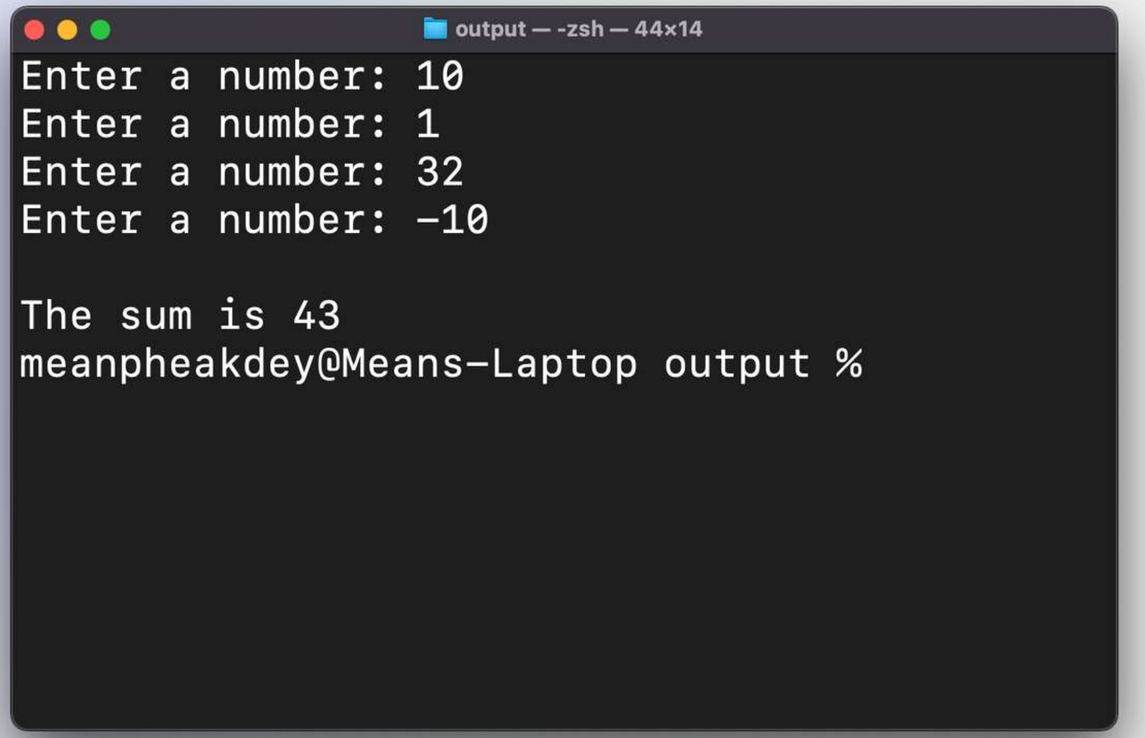
Statements នឹងអនុវត្តម្តងហើយម្តងទៀត

បើសិនជាលទ្ធផល Boolean ជាតម្លៃពិត

ឧទាហរណ៍

```cpp
int main() {
    int number;
    int sum = 0;

    cout << "Enter a number: ";
    cin >> number;

    while (number >= 0) { //the negative number entered is not added to the sum
        // add all positive numbers
        sum += number;
        // take input again if the number is positive
        cout << "Enter a number: ";
        cin >> number;
    }
    cout << "\nThe sum is " << sum << endl;

    return 0;
}
```

លទ្ធផល



DO...WHILE LOOP STATEMENT

តើអ្វីទៅជា Do...While loop?

- ជាប្រភេទ Loop ដែលមានភាពស្រដៀងគ្នាទៅនឹង while loop ដែរ។ ចំណុចខុសគ្នាគឺ while មុននឹងអនុវត្ត Statement វាត្រូវពិនិត្យលក្ខខណ្ឌជាមុន ផ្ទុយមកវិញ do... while អនុវត្ត Statement បានម្តងសិនទើបពិនិត្យលក្ខខណ្ឌតាមក្រោយ។
  រង្វិលជុំ do...while loop មានទម្រង់ទូទៅដូចខាងក្រោម ៖

```
do{
    statement(s);
}while(Boolean);
```

Statements នឹងអនុវត្តម្តងហើយម្តងទៀត

បើសិនជាលទ្ធផល Boolean ជាតម្លៃពិត

ឧទាហរណ៍

```cpp
int main() {
    int number = 0;
    int sum = 0;

    do {
        sum += number;
        // take input from the user
        cout << "Enter a number: ";
        cin >> number;
    }
    while (number >= 0); //If the user enters a negative number, the loop ends

    cout << "\nThe sum is " << sum << endl;

    return 0;
}
```

លទ្ធផល

```
output — -zsh — 44×14
Enter a number: 10
Enter a number: 1
Enter a number: 32
Enter a number: -10

The sum is 43
meanpheakdey@Means-Laptop output %
```

EXERCISES

1. Make a simple 5-function calculator program that allowed the user to select from a menu of operations:
    - Add two numbers
    - Subtract two numbers
    - Multiply two numbers
    - Divide two numbers
    - Modulus two numbers
2. ចូរសរសេរកម្មវិធីដើម្បីរកចំនួនធំជាងគេរវាងអថេរពីរចំនួន។ រួច ហើយបង្ហាញតម្លៃដែលធំជាងគេមកលើអេក្រង់។ សន្មត់ថា យើងមានអថេរ A, B, C ដែលមានប្រភេទទិន្នន័យជាចំនួនគត់ ហើយតម្លៃ របស់អថេរ A, B, C ត្រូវបញ្ចូលតាមរយៈ Keyboard នៅពេលដែលកម្មវិធីកំពុងតែ ដំណើរការ។
3. ហាងលក់ទំនិញក្នុងផ្សារទំនើបមួយកន្លែងបានកំណត់ការចុះថ្លៃជូនអតិថិជនរបស់ គេដូចខាងក្រោម:
    - បើសិនជាអ្នកទិញទំនិញក្រោម 200$ នោះគេនឹងចុះថ្លៃជូន 1%
    - បើសិនជាអ្នកទិញទំនិញចាប់ពី 200$ ដល់ 499.99$ នោះគេនឹងចុះថ្លៃជូន 2%
    - បើសិនជាអ្នកទិញទំនិញចាប់ 500$ ដល់ 999.99$ នោះគេនឹងចុះថ្លៃជូន 3%
    - បើសិនជាអ្នកទិញទំនិញចាប់ 1000$ ដល់ 1499.99$ នោះគេនឹងចុះថ្លៃជូន 4%
    - បើសិនជាអ្នកទិញទំនិញចាប់ 1500$ ឡើងទៅ នោះគេនឹងចុះថ្លៃជូន 5%

      ចូរសរសេរកម្មវិធីសំរាប់ហាងលក់ទំនិញនេះ។
4. ចូរសរសេរកម្មវិធីដើម្បីធ្វើការគណនាផលបូកចំនួនគត់នៃ N ចំនួន ដែលយើងវាយបញ្ចូលតាមរយៈ keyboard នៅពេលដែលកម្មវិធីកំពុងដំណើរការ ដោយប្រើរង្វិលជុំនឹងរកមធ្យមភាគ រួចបង្ហាញលទ្ធផលមកលើផ្ទៃនៃលទ្ធផល។

DATA STRUCTURE

តើអ្វីទៅជា Data Structure នៅក្នុង C/C++?

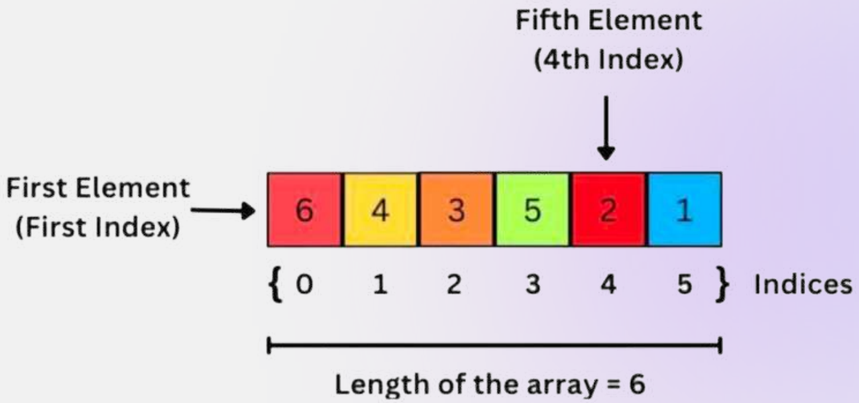
- Data Structure នៅក្នុង C++ មានសារៈសំខាន់សម្រាប់ការរៀបចំ និងគ្រប់គ្រងទិន្នន័យនៅក្នុងកម្មវិធីមួយ។ Data Structure ទូទៅមួយចំនួននៅក្នុង C++ រួមមាន arrays, linked lists, stacks, queues, និង trees ជាដើម ។ ប៉ុន្តែនៅក្នុងមេរៀននេះយើងនឹងលើកយកតែចំណុច arrays តែប៉ុណ្ណោះមកសិក្សា ។

ARRAYS

តើអ្វីទៅជា Array?

- Array ជាសំណុំនៃ Variables សម្រាប់តំណាងឲ្យទិន្នន័យណាមួយ ដែលVariables ទាំងនោះមាន DataType ដូចៗគ្នា និងអាចផ្ទុកតម្លៃរៀងៗខ្លួនតាមរយៈ index របស់វា។
- Array អាចជា Array មួយវិមាត្រ ឬ ច្រើនវិមាត្រ ។ ប៉ុន្តែយើងនឹងលើកយកតែ Array មួយវិមាត្រតែប៉ុណ្ណោះមកសិក្សា ។

ARRAY 1D



- នៅក្នុង C/C++ គេប្រើ array មួយវិមាត្រដើម្បីរក្សាទុកបណ្តុំនៃប្រភេទទិន្នន័យដូចគ្នាដែលតម្រៀបគ្នាជា List នៅក្នុងលំដាប់ លីនេអ៊ែ (ទិន្នន័យតាមលំដាប់លំដោយ)។ ដើម្បីប្រកាស Array មួយវិមាត្រនោះ យើងត្រូវអនុវត្តន៍តាមទម្រង់ដូចខាងក្រោម៖

```
Datatype arrName[element];
```

ការបង្កើត Array 1D

- ឧបមាថាថ្នាក់មួយមានសិស្ស 27 នាក់ ហើយយើងត្រូវរក្សាទុកចំណាត់ថ្នាក់នៃពួកគេទាំងអស់។ ជំនួសឱ្យការបង្កើតអថេរ 27 ដាច់ដោយឡែក យើងអាចបង្កើត array បានយ៉ាងសាមញ្ញ៖

```
double grade[27];
```

- នៅក្នុង C++ ធាតុនីមួយៗនៅក្នុង array មួយត្រូវបានភ្ជាប់ជាមួយនឹងលេខ។ លេខនោះត្រូវបានគេស្គាល់ថាជា index ។ យើងអាចចូលប្រើធាតុនៃ array មួយដោយប្រើ index ទាំងនោះ។

```
array[index]; //syntax to access array
```

C++ Array Initialization

- នៅក្នុង C++ យើងអាចចាប់ផ្តើម បញ្ចូលតម្លៃទៅឲ្យ array មួយកំឡុងពេលប្រកាសបាន។
  ឧទាហរណ៍ ៖

```
int x[6] = {19, 10, 8, 17, 9, 15}; // declare and initialize and array
int x[ ] = {19, 10, 8, 17, 9, 15};
```

ការប្រើប្រាស់ Array 1D

- ឧបមាថាថ្នាក់មួយមានសិស្ស 27 នាក់ ហើយយើងត្រូវរក្សាទុកចំណាត់ថ្នាក់នៃពួកគេទាំងអស់។ ជំនួសឱ្យការបង្កើតអថេរ 27 ដាច់ដោយឡែក យើងអាចបង្កើត array បានយ៉ាងសាមញ្ញ៖

```
double grade[27];
```

- នៅក្នុង C++ ធាតុនីមួយៗនៅក្នុងអារេមួយត្រូវបានភ្ជាប់ជាមួយនឹងលេខ។ លេខនោះត្រូវបានគេស្គាល់ថាជា index ។ យើងអាចចូលប្រើធាតុនៃ array មួយដោយប្រើ index ទាំងនោះ។

```
array[index]; //syntax to access array
```

ការបញ្ចូលតម្លៃទៅឲ្យ Arrays

- ការបញ្ចូលទិន្នន័យទៅឲ្យ Arrays គឺយើងអាចបញ្ចូលដោយផ្ទាល់ក៏បាន ឬ បញ្ចូលតាមរយៈ Keyboard ក៏បាន ។ ខាងក្រោមជាឧទាហរណ៍នៃការបញ្ចូល Array ដោយផ្ទាល់ ៖

```
// Declare an array of integer
int myArray[5];

// initialize individual elements

myArray[0] = 10;
myArray[1] = 2;
myArray[2] = 44;
myArray[3] = 66;
myArray[4] = 88;
```

ការអានតម្លៃចេញពី Array

- ដើម្បីហៅទិន្នន័យចេញពីក្នុង Array មួយវិមាត្រឬក៏ពីរវិមាត្រ យើងអាចប្រើ ប្រាស់ loop ណាមួយក្នុងចំណោម loops ទាំងបីដែលយើងបានសិក្សារួចមក ហើយ។

  ឧទាហរណ៍ ៖

```
for(int i=0; i<5; i++)
{
    //Access and manipulate each element
    cout << "Value Of myArray is " << myArray[i] << endl;
}
```

EXERCISES

1. ចូរសរសេរកម្មវិធីមួយដែលអាចបញ្ចូលពត៌មានសិស្ស ដែលមានដូចជា ID, Name, Sex, Phone, address ឲ្យបាន 3 នាក់ដោយចាប់តម្លៃតាមរយៈ keyboard។ បន្ទាប់មកធ្វើការបង្ហាញពត៌មានទាំងអស់នោះមកលើ screen។ បង្កើតជាជម្រើសមួយដែលអាចស្វែងរកពត៌មានតាម ID បាន ដោយប្រើប្រាស់ array មួយវិមាត្រ។
2. ចូរសរសេរកម្មវិធីមួយដើម្បីបញ្ចូលតម្លៃទៅក្នុង Array មួយដែលមាន N ធាតុ រួចហើយបង្ហាញតម្លៃទាំងអស់មកលើអេក្រង់។ បន្ទាប់មកទៀតតំរៀបតម្លៃរបស់ Array តាមលំដាប់កើន រួចហើយបង្ហាញតម្លៃដែលបានតំរៀបរួចមកលើអេក្រង់។
3. ចូរសរសេរកម្មវិធីមួយដើម្បីបញ្ចូលតម្លៃទៅក្នុង Array មួយដែលមាន N ធាតុ រួចហើយបង្ហាញតម្លៃទាំងអស់មកលើអេក្រង់។ បន្ទាប់មកទៀត search តម្លៃរបស់ Array រួចហើយបង្ហាញតម្លៃដែលបាន search រួចមកលើអេក្រង់ បើតម្លៃនោះមិនមានក្នុង Array បង្ហាញ Error Message ចេញមកក្រៅរួចមានលក្ខខណ្ឌសម្រាប់ឲ្យ User ធ្វើការ Search ម្តងទៀត។
4. ចូរសរសេរកម្មវិធីមួយដើម្បីបញ្ចូលឈ្មោះនិស្សិត N នាក់
    - បង្ហាញឈ្មោះនិស្សិតទាំងអស់មកវិញ
    - តំរៀបឈ្មោះតាមលំដាប់ពីតូចទៅធំ (A → Z)
    - បង្ហាញឈ្មោះបន្ទាប់ពីតំរៀបរួច

FUNCTION

- Function គឺជាបណ្តុំនៃ statements ប្រើវាសម្រាប់ដោះស្រាយបញ្ហាណាមួយ ។ អ្នកក៏ធ្លាប់បានប្រើប្រាស់ Build-in function មួយចំនួនរួចមកហើយមានដូចជា: getch(), sizeof(), getchar(),..។ Function ចែកចេញជាពីរប្រភេទគឺ Return type និង Non-return type

  ទម្រង់ទូទៅនៃការបង្កើត Function ៖

```
DataType function_name( parameter(s))
{
        statement(s);
        return expression;
}
```

FUNCTION

- Datatype គឺជាប្រភេទទិន្នន័យណាមួយដែលមានដូចជា: int, void, char, float, double។ល។
- Function_Name គឺជាឈ្មោះរបស់អនុគមន៍ដែលអ្នកសរសេរកម្មវិធីជាអ្នក កំណត់ឲ្យ។
- ហើយ Parameter គឺជាបញ្ជីនៃប៉ារ៉ាម៉ែត្រទាំងឡាយដែលកំណត់ឡើងមាន ទម្រង់ដូចគ្នាទៅនឹងការប្រកាសអថេរឬចំនួនថេរដែរ ហើយប៉ារ៉ាម៉ែត្រទាំងនោះ ត្រូវមានប្រភេទទិន្នន័យរៀងៗខ្លួនរបស់វា ។
- ហើយ statement(s) គឺជាការបង្គាប់បញ្ជា ដែលនឹងដំណើរការការងាររបស់ អនុគមន៍ ដែលបណ្តាបង្គាប់បញ្ជាទាំងអស់នោះត្រូវបានគេហៅថា ដងខ្លួនរបស់ អនុគមន៍ (body of function)។
- ហើយ return expression គឺជាការបញ្ជូនលទ្ធផលរបស់អនុគមន៍ទៅឲ្យ អនុគមន៍ខ្លួនឯង។ ប្រសិនបើអ្នកសរសេរកម្មវិធីប្រើប្រាស់ពាក្យគន្លឹះ void នៅពី មុខឈ្មោះអនុគមន៍ នោះអនុគមន៍នឹងមិនមាន return expression ទេ។

Parameter និង Argument

- Parameter គឺជាបណ្តាឈ្មោះនៃអថេរអ្វីមួយដែលប្រើសម្រាប់យកទិន្នន័យចូលទៅក្នុងតួ ខ្លួនរបស់អនុគមន៍ ក្នុងខណៈដែលគេកំពុងប្រកាសអនុគមន៍មួយ។ ឧទាហរណ៍ដូចជាអនុគមន៍ void Sum(int a, int b); ក្នុងនោះមាន a និង b ជាប៉ារ៉ាម៉ែត្ររបស់អនុគមន៍ Sum។
- ចំណែកឯ Argument វិញគឺជាតម្លៃណាមួយដែលកំពុងរក្សាទុកក្នុងអថេរឬចំនួនថេរ ហើយដែលត្រូវបញ្ជូនទៅអោយប៉ារ៉ាម៉ែត្ររបស់អនុគមន៍ នៅពេលដែលឈ្មោះរបស់អនុគមន៍ នោះត្រូវបានគេហៅមកប្រើ។ ឧទាហរណ៍ដើម្បីហៅអនុគមន៍ Sum ខាងលើមកប្រើ នោះត្រូវយើងត្រូវសរសេរថា Sum (10, 20); ដែលពេលនោះតម្លៃ 10 ត្រូវបញ្ជូនទៅឲ្យ ប៉ារ៉ាម៉ែត្រ a ចំណែកតម្លៃ 20 ត្រូវបញ្ជូនទៅអោយប៉ារ៉ាម៉ែត្រ b។

TYPE OF FUNCTION

- ជាទូទៅ User-Defined Function ត្រូវបានបែងចែកជា ២ គឺ
  - Return Type Function
  - Non-Return Type Function
- អ្វីទៅជា Return Type Function?
  - Return Type Function គឺជាអនុគមន៍ទាំងអស់ដែលមានប្រភេទទិន្នន័យ លើកលែងតែអនុគមន៍ដែលមិន return តម្លៃទេដែលជំនួសទៅដោយពាក្យគន្លឹះ void។
- អ្វីទៅជា Non-Return Type Function?
  - Non-Return Type Function គឺជាអនុគមន៍ទាំងអស់ដែលមានប្រភេទទិន្នន័យជា Void និងមិនមាន return តម្លៃចេញមកក្រៅវិញឡើយ។

Example 01

- បង្កើត Return-Type Function មួយឈ្មោះ SUM ដែលមានមុខងារក្នុងការ បូកចំនួន ២ ដែលបញ្ចូលតាម Keyboard ។
- ខាងស្តាំដៃគឺជា ឧទាហរណ៍ ដែលបង្ហាញពីការ ប្រើប្រាស់ Return-Type Function ដែលបាន Parameter ចំនួនពីរសម្រាប់ទទួលតម្លៃពីខាងក្រៅ ។

```cpp
#include <iostream>

using namespace std;

int Sum(int a, int b){
    return a + b;
}

int main(){
    int a, b;
    cout << "Enter A : ";
    cin >> a;

    cout << "Enter B : ";
    cin >> b;

    cout << "Sum of A and B is : " << Sum(a, b) << endl;
    return 0;
}
```

Example 02

- បង្កើត Non-Return Type Function មួយឈ្មោះ SUM ដែលមានមុខងារក្នុងការ បូកចំនួន ២ ដែលបញ្ចូលតាម Keyboard ។
- ខាងស្តាំដៃគឺជា ឧទាហរណ៍ ដែលបង្ហាញពីការ ប្រើប្រាស់ Non-ReturnType Function ដែលបាន Parameter ចំនួនពីរសម្រាប់ទទួលតម្លៃពីខាងក្រៅ ។

```cpp
#include <iostream>

using namespace std;

void Sum(int a, int b){
     cout << "Sum of A and B is : " << a + b << endl;
}

int main(){
    int a, b;
    cout << "Enter A : ";
    cin >> a;

    cout << "Enter B : ";
    cin >> b;

    Sum(a, b);

    return 0;
}
```

EXERCISES

1. ចូរសរសេរកម្មវិធីរកចំនួនធំបំផុត និង ចំនួនតូចបំផុត នៃបីចំនួនដោយប្រើអនុគមន៍ (Function) ។
2. រដ្ឋបានកំណត់យកពន្ធប្រាក់ខែពីមន្ត្រីរាជការ និង បុគ្គលិកក្រុមហ៊ុនដូចខាងក្រោម:
   - ចំពោះមន្ត្រីរាជការ ឬក៏បុគ្គលិកក្រុមហ៊ុន ដែលទទួលបានប្រាក់ចំណូលក្រោម 100$ នោះរដ្ឋនឹងដកយកពន្ធ 0.1%
   - ចំពោះមន្ត្រីរាជការ ឬក៏បុគ្គលិកក្រុមហ៊ុន ដែលទទួលបានប្រាក់ចំណូលចាប់ពី 100$ ដល់ 499$ នោះរដ្ឋនឹងដកយកពន្ធ 1.5% បន្ថែមពីលើទៀត
   - ចំពោះមន្ត្រីរាជការ ឬក៏បុគ្គលិកក្រុមហ៊ុន ដែលទទួលបានប្រាក់ចំណូលចាប់ពី 500$ ដល់ 999$ នោះរដ្ឋនឹងដកយកពន្ធ 2.5% បន្ថែមពីលើទៀត
   - ចំពោះមន្ត្រីរាជការ ឬក៏បុគ្គលិកក្រុមហ៊ុន ដែលទទួលបានប្រាក់ចំណូលចាប់ពី 1000$ ដល់ 1999$ នោះរដ្ឋនឹងដកយកពន្ធ 3.5% បន្ថែមពីលើទៀត
   - ចំពោះមន្ត្រីរាជការ ឬក៏បុគ្គលិកក្រុមហ៊ុន ដែលទទួលបានប្រាក់ចំណូលចាប់ពី 2000$ ឡើងទៅ នោះរដ្ឋនឹងដកយកពន្ធ 4.5% បន្ថែមពីលើទៀត
     - ចូរសរសេរកម្មវិធីដើម្បីរកប្រាក់ពន្ធដែលរដ្ឋទទួលបានពីមន្ត្រីរាជការនឹងបុគ្គលិកក្រុម ហ៊ុននៅទូទាំងប្រទេស។
     - បញ្ជាក់: ដោយប្រើប្រាស់រង្វិលជុំ do...while ហើយកម្មវិធីនេះនឹងបញ្ចប់នៅពេលអ្នក ប្រើប្រាស់វាយអក្សរ ‘N’ or ‘n’ និង ប្រើប្រាស់អនុគមន៍
3. ចូរសរសេរកម្មវិធីមួយដើម្បីបញ្ចូលព័ត៌មានរបស់និស្សិតចំនួន N នាក់។ ដែលព័ត៌មានទាំងនោះ មានដូចជា៖ ID, Name, Sex, CPL, MAT, ENG, FUN, AVG បន្ទាប់ពីបញ្ចូលព័ត៌មានខាងលើរួចហើយ ចូរសរសេរកម្មវិធីដើម្បីដោះស្រាយបញ្ហាមួយចំនួន ដូចខាងក្រោម៖
   - បង្ហាញព័ត៌មានបន្ទាប់ពីបញ្ចូលរួចមកលើអេក្រង់
   - តំរៀបមធ្យមភាគនៃព័ត៌មានខាងលើតាមលំដាប់ពីធំមកតូច
   - បង្ហាញព័ត៌មានដែលបានធ្វើការតំរៀបរួចហើយមកលើអេក្រង់

     ***បញ្ជាក់: ចូរបង្កើតអនុគមន៍នៅក្នុងការដោះស្រាយបញ្ហានីមួយៗខាងលើ

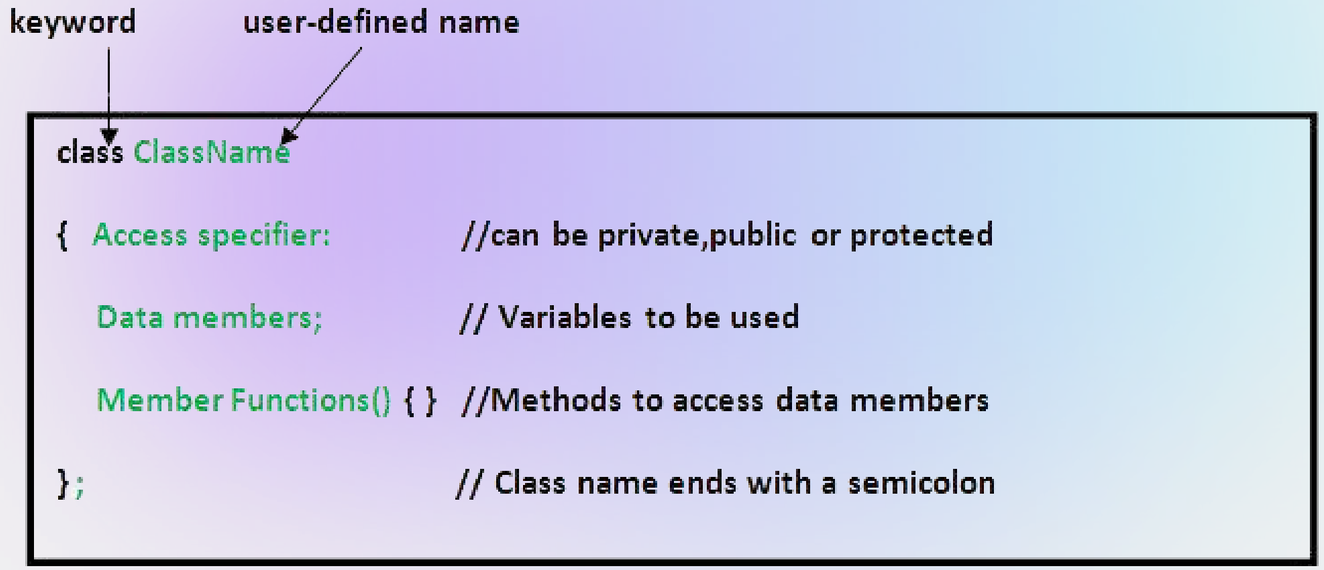
CLASS & OBJECT

តើអ្វីទៅជា Class & Object?

- Class គឺជាពុម្ព ឬក៏ ជាការតំណាងឲ្យទិន្នន័យអ្វីមួយ ដែលអាចហៅថា Datatype ឬ User-Defined Datatype ដែលអ្នក Programmers បង្កើតឡើងសម្រាប់តំណាងទិន្នន័យអ្វីមួយនៅក្នុង Program របស់ពួកគេ។
- ចំណែកឯ Object វិញគឺជាអ្វីៗដែលមានប្រភេទច្បាស់លាស់ណាមួយ។ គ្រប់ Object ត្រូវមានរូបរាង (Attributes) និង សកម្មភាព(Behavior)។ ជាក់ស្តែង Object គឺកើតចេញពី Class ដែលបានផ្សំឡើងពី Data និង Function។

ទម្រង់ទូទៅនៃការបង្កើត Class

- ខាងក្រោមនេះគឺជាទម្រង់ទូទៅនៃការបង្កើត Class នៅក្នុង C++ ៖



ពាក្យបច្ចេកទេសក្នុង Class

- អ្វីទៅជា Class Data Members?
  - Class Data Members គឺសំដៅទៅលើអថេរទាំងឡាយណាដែលបានប្រកាសនៅក្នុង Class
- អ្វីទៅជា Class Member Functions?
  - Class Member Functions គឺសំដៅទៅលើ function ឬ method ទាំងឡាយណាដែលបានប្រកាសនៅក្នុង Class ណាមួយ។ ជាក់ស្តែង យើងប្រើប្រាស់វាសម្រាប់ឲ្យធ្វើការលើ Data members និង ទិន្នន័យ Object ផ្សេងទៀត។
- អ្វីទៅជា Access Specifier?
  - Access Specifier សំដៅទៅលើដែនកំណត់មួយក្នុងការកំណត់ពីទំហំការងារដែលអាចធ្វើទៅបាននៃ Members របស់ Class។ Access Specifier នៅក្នុង Class សំខាន់ៗមាន ៣ គឺ Private, Public និង Protected ។

Class Access Specifier

- ខាងក្រោមនេះគឺជាតួនាទីរបស់ Class Access Specifier ៖
  - The public Members គឺជា Access Specifier ដែលអាចទាញយក (accessible) ទៅប្រើបានពីគ្រប់ទីកន្លែងដែលនៅខាងក្រៅ Class បាន។
  - The private Members គឺជា Access Specifier ដែលមិនអាចទាញយក (inaccessible) ទៅប្រើប្រាស់នៅខាងក្រៅ Class បានឡើយ។
  - The protected Members គឺជា Access Specifier ដែលមិនអាចទាញយក (inaccessible) ទៅប្រើប្រាស់នៅខាងក្រៅ Class បានដូច private ដែរប៉ុន្តែលើកលែងក្នុងលក្ខខណ្ឌ inheritance។

```
class Student{
    public: //access specifier
        //class data members
        int age;
        int score;

        //class member functions
        void AddStudent(){}
        void Display(){}.
};//class ended with semicolem
```

```
class Employee{
    public: //access specifier
        //class data members
        string name;
        float hours, rate;

        //class member functions
        void AddStudent(){}
        float CalSalary(){}
        void Display(){}
};//class ended with semicolem
```

Example

- ខាងឆ្វេងដៃគឺជា ឧទាហរណ៍ ដែលបង្ហាញពីការបង្កើត Class ដែលមាន Data members និង Member function សម្រាប់គ្រប់គ្រង Student និង Employee

ទម្រង់ទូទៅនៃការបង្កើត Object

- ខាងក្រោមនេះគឺជាទម្រង់ទូទៅនៃការបង្កើត Object នៅក្នុង C++ ៖

```
className objectName;
```

- ប្រសិនបើ Object ច្រើនមាន Abstract Datatype ដូចៗគ្នាយើងអាចបង្កើតវាតាមទម្រង់ទូទៅដូចខាងក្រោម ៖

```
className objectName1, objectName2 ...;
```

ការ Access Members របស់ Class

តើយើងអាចចូលទៅ Access Class Member របស់ Class ណាមួយបានដោយរបៀបណា?

- ដើម្បីអាចចូលទៅ access class member បាន គឺយើងប្រើ ជាមួយនឹងសញ្ញា dot "." ។

```
objectName.ClassMember;
```

- ប៉ុន្តែនៅពេលដែលយើងចង់ access member ណាមួយនោះ យើងក៏ត្រូវមើលទៅលើ access specifier របស់ class នោះ ផងដែរ។

Example

```
class Student{
    public: //access specifier
        //class data members
        int age;
        int score;

        //class member functions
        void AddStudent(){
            cout << "Enter age : "; cin >> age;
            cout << "Enter score : "; cin >> score;
        }
        void Display(){
            cout << "Age : " << age << ", Score : " << score << endl;
        }
};//class ended with semicolem
```

```
int main(){
    Student s1, s2;

    cout << "Adding details of first student" << endl;
    s1.AddStudent();
    cout << "Adding details of second student" << endl;
    s2.AddStudent();

    cout << "\nDisplaying the information of both students\n";
    s1.Display();
    s2.Display();

    return 0;
}
```

ការ Access Private Members

តើយើងអាច Access Private Member ពី ខាងក្រៅ Class បានដោយវិធីណា?

- យើងអាច Access Private Member ពី ខាងក្រៅ Class បានដោយប្រើ Method (function) ឈ្មោះថា getter និង setter នោះ យើងអាចទាញយក និង កំណត់តម្លៃទៅឲ្យ private member របស់ Class នោះបាន។
  - Setter គឺជា Method សម្រាប់ផ្តល់តម្លៃទៅឲ្យ Data member តាមវិធីណាមួយ។
  - Getter គឺជា Method ដែលអាចទាញយកតម្លៃចេញពី Data member មកប្រើប្រាស់បាន។

Example

```
class Student{
    private: //access specifier
        //class data members
        int age;
        int score;

    public:
        //setter
        void SetStudent(int age, int score){
            this->age = age;
            this->score = score;
        }
        //getter
        int GetAge(){ return age; }
        int GetScore(){ return score; }
};
```

```
int main(){
    Student s1, s2;

    cout << "Setting Students info!" << endl;
    s1.SetStudent(20, 100);
    s2.SetStudent(21, 90);

    cout << "Age of the first student is : "<<s1.GetAge()
         << " and score is : "<<s1.GetScore()<<endl;

    cout << "Age of the second student is : "<<s2.GetAge()
         << " and score is : "<<s2.GetScore()<<endl;

    return 0;
}
```

Inheritance

អ្វីទៅជា Inheritance?

- Inheritance គឺជាផ្ទេរ class member ឬអាចនិយាយម៉្យាង ទៀតថា ជាការផ្ទេរមរតកពី Parent/Base Class ទៅឲ្យ Child/Derived Class

```
class ChildClass : access specifier BaseClass
```

- child class គឺជា Class ថ្មីដែលនឹងទទួលមរតកពី Base Class
- base class គឺជា Class មេរបស់ child class
- access specifier ត្រង់នេះ គឺប្រៀបបានជាច្រកចូលមក កាន់ Child Class អីចឹង គឺវាអាច ជាប្រភេទ public, private ឬ protected ។

Example

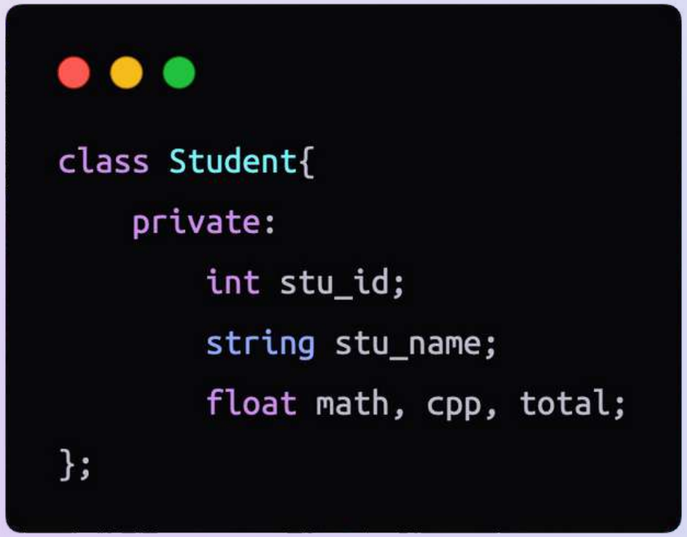
```cpp
// Base class
class Person {
    protected:
        string name;
        int age;

    public:
        // Setter method for all attributes
        void setAll(string n, int a) {
            name = n;
            age = a;
        }

        void displayInfo() {
            cout << "Name: " << name << ", Age: " << age << endl;
        }
};
```

```cpp
// Derived class (inherits from Person)
class Student : public Person {
    private:
        int studentID;

    public:
        // Setter method for all attributes
        void setAll(string n, int a, int id) {
            Person::setAll(n, a); // Call base class setAll
            studentID = id;
        }

        void displayStudentInfo() {
            displayInfo(); // Call the base class method
            cout << "Student ID: " << studentID << endl;
        }
};
```

```cpp
int main() {
    // Create an object of the derived class
    Student student;

    // Set all information using the setAll method
    student.setAll("John Doe", 20, 12345);

    // Call methods from the base class
    student.displayInfo();

    // Call method from the derived class
    student.displayStudentInfo();

    return 0;
}
```

EXERCISES

1. ចូរប្រកាស Class មួយសម្រាប់ផ្ទុកព័ត៌មានរបស់និស្សិតម្នាក់ដែលមាន អត្តលេខ (stu_id), ឈ្មោះ (stu_name), ពិន្ទុតាមមុខវិជ្ជា (math, cpp) និង ពិន្ទុសរុប (total) ដូចខាងក្រោម

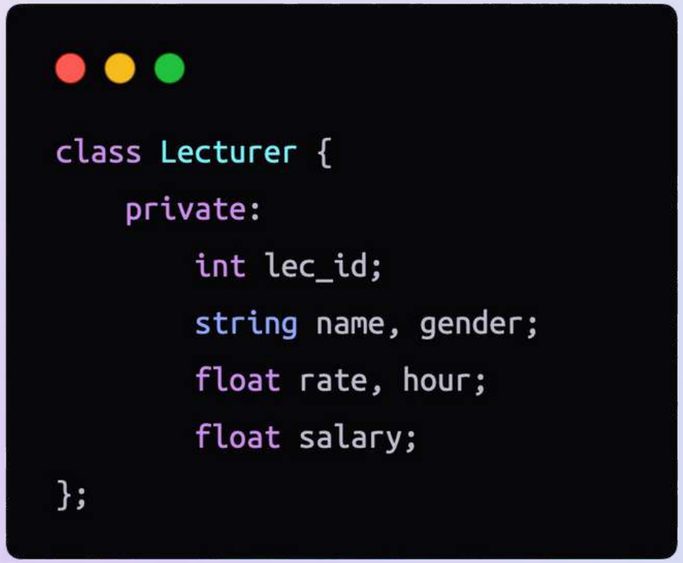


```
class Student{
    private:
        int stu_id;
        string stu_name;
        float math, cpp, total;
};
```

- បង្កើត Setter និង Getter ដើម្បីគ្រប់គ្រងទិន្នន័យរបស់ សិស្ស
- បញ្ចូលទិន្នន័យទៅឲ្យ Student តាមរយៈ keyboard។
- បន្ទាប់មកបង្ហាញព័ត៌មានទាំងអស់នោះមកលើអេក្រង់វិញ។

EXERCISES

2. ចូរសរសេរកម្មវិធីមួយដើម្បីបញ្ចូលព័ត៌មានរបស់សាស្ត្រាចារ្យចំនួន N នាក់ ដែល ព័ត៌មានទាំងនោះមានដូចខាងក្រោម:



```
class Lecturer {
    private:
        int lec_id;
        string name, gender;
        float rate, hour;
        float salary;
};
```

- បង្កើត Setter និង Getter ដើម្បីគ្រប់គ្រងទិន្នន័យរបស់ សាស្ត្រាចារ្យ
- បញ្ចូលទិន្នន័យទៅឲ្យ Lecturer តាមរយៈ keyboard។
- បន្ទាប់មកបង្ហាញព័ត៌មានទាំងអស់នោះមកលើអេក្រង់វិញ។
- បង្កើត Method សម្រាប់ Search ទិន្នន័យរបស់ Lecturer តាម lec_id។

EXERCISES

2. ចូរសរសេរកម្មវិធីមួយដើម្បីគ្រប់គ្រងទិន្នន័យរបស់ Product ដូចខាងក្រោម:

```
class Product {
    protected:
        int barcode, qty;
        string proName;
        float price;
        float totalPrice;

    public:
        //setProduct
        //displayProduct
};
```

```
class ProductManager : public Product{
    private:
        int sellPro;

    public:
        //sellProduct
        //displayProduct after sell
};
```

- បង្កើត Setter និង Display Method ដើម្បីគ្រប់គ្រងទិន្នន័យរបស់ Product
- បញ្ចូលទិន្នន័យទៅឲ្យ Product តាមរយៈ keyboard។
- អាចលក់ Product នោះបានតាមរយៈ Object
- បន្ទាប់មកបង្ហាញព័ត៌មានទាំងអស់នោះមកលើអេក្រង់វិញ។

Thank You

ANT TECHNOLOGY TRAINING CENTER