SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya College Of Engineering

(Module-01)

C++ Programming Basics

SMITA SANKHE
Assistant Professor
Department of Computer Engineering

What is C++?

- The C++ programming language can be thought of as a "**superset**" of the C language: a legal ANSI C program is a legal C++ program and means the same thing.

Why study C++ ?

1.Foundational Knowledge
2.System Programming
3.Performance-Critical Applications
4.Game Development
5.Embedded Systems
6.Competitive Programming
7.Cross-Platform Development
8.Performance Critical Libraries
9.Understanding Memory Management

A simple C++ program

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

//This program displays a "Hell Word" message
int main( ){
    cout<<"Hello Word"<<endl;
    return 0;
    }
```

- Save as hello.cpp

Explanation

- Files that contain C++ programs are named using the .cpp file extension (for example hello.cpp).
- The statement `#include<iostream>` is a preprocessor directive that is included in C++ program to allow the program to perform the input and output.
- `iostream` is a standard C++ header file that defines `cin`, `cout`, and the operators << and >>.
- `using namespace std` is written to avoid prefixing standard library names like `cin, cout, endl` with `std::`
- Files that contains C++ programs must have a `main()` function to indicate where the program execution begins.
- The `main( )` function usually is written to return an integer value.
- The statement `return 0` will tell C++ to exit the `main()` function.

Primitive data types, sizes, limits

Type	Size (bytes)	Min Value	Max Value
bool	1	0	1
char	1	-128	127
signed char	1	-128	127
unsigned char	1	0	255
short	2	-32768	32767
unsigned short	2	0	65535
int	4	-2147483648	2147483647
unsigned int	4	0	4294967295
long	4	-2147483648	2147483647
unsigned long	4	0	4294967295
long long	8	-9223372036854775808	9223372036854775807
unsigned long long	8	0	18446744073709551615
float	4	1.17549e-38	3.40282e+38
double	8	2.22507e-308	1.79769e+308
long double	16	3.3621e-4932	1.18973e+4932

Example of input / output in C++

- Refer iodemo.cpp

Explanation of cin and cout w.r.t. iodemo.cpp

- The expression cin >> a causes the program to read an integer into the variable a, from the standard input.
- Similarly, cin >> b reads an integer into b, but cin >> c interprets its input as a real number, and stores it in c.
- The next statement, cin >> str reads characters until a whitespace is encountered, and stores a null-terminated string in str (excluding the whitespace).
- Note that if you typed a very long string (i.e., more than 20 characters), your program could crash.
- To read a string with spaces, use getline(cin, str);

OOP Basics in C++

- `class` Keyword
  - The class keyword is used to define a new class.
  - A class is a blueprint for creating objects, which can include both data members (variables) and methods (functions)
- By default, all members of a class are private if no access specifier is declared.
- This means they can only be accessed and modified by member functions of the class and not from outside the class.
- The public keyword makes the members accessible from outside the class.

Why Use Access Specifiers

- Encapsulation
  - Encapsulation is one of the fundamental principles of object-oriented programming (OOP).
  - By making member variables private, we control how they are accessed and modified.
  - This helps in maintaining the integrity of the data.
- Interface vs. Implementation
  - Public methods serve as the interface through which the outside world interacts with the class, while private members and methods are the implementation details that are hidden from the outside world.

Public Section

- Members declared in the public section can be accessed from outside the class.
- This is where you usually put the interface of the class: methods and constructors that users need to access.

```
class MyClass {
public:
    void publicMethod() {
        // Method accessible from outside the class
    }
};
```

Private Section

- Members declared in the private section can only be accessed by other members of the class.
- This is used to encapsulate the internal state and implementation details of the class.

```
class MyClass {
private:
    int privateMember;
    void privateMethod() {
        // Method not accessible from outside the class
    }
};
```

Constructors in C++

- A constructor is a special member function of a class that is automatically called when an object of that class is created.
- Constructors are used to initialize the objects of a class.
- They have the same name as the class and do not have a return type.

Types of Constructors

- Default Constructor
  - A constructor that takes no arguments.
  - It is called when an object is created without any parameters.
  - Example:
  - class MyClass {
  - public:
  - MyClass() {
  - // Default constructor body
  - }
  - };

- Parameterized Constructor:
  - A constructor that takes one or more arguments.
  - It is used to initialize objects with specific values.
  - Example:

```
class MyClass {
public:
    MyClass(int value) {
        // Parameterized constructor body
    }
};
```

- Member Initialization List
  - A member initialization list is a more efficient way to initialize class members.
  - It is preferred over assigning values in the constructor body, especially for initializing const members and reference members.
  - Example:

```
class MyClass {
    int x;
    const int y;
    float z;
public:
    MyClass(int a, int b, float c) : x(a), y(b), z(c) {
        // Constructor body (if needed)
    }
};
```

Creating Objects in C++

- Using the Default Constructor
  - Syntax: classname objectname;
  - Example:

```
int main(){
    MyClass m1;
}
```

- Using a Parameterized Constructor
  - Syntax: className objectName(arguments);
  - Example:

```
int main(){
    MyClass m2(3,4,5.5);
}
```

- Refer classDemo.cpp

Constructor VS Destructor

Feature	Constructor	Destructor
Purpose	Initializes an object when it is created.	Cleans up resources when an object is destroyed.
Invocation	Automatically called when an object is instantiated.	Automatically called when an object is destroyed or goes out of scope.
Syntax	Has the same name as the class, no return type. Example()	Same name as the class but preceded by a tilde ~. ~Example()
Parameters	Can accept parameters to initialize the object.	Cannot accept parameters.
Execution Time	Executes when the object is created.	Executes when the object is destroyed.
Resource Management	Used for resource allocation (like memory or file handling).	Used for resource de-allocation or cleanup.

Integer and Floating-point Division

- The behavior of division operator (/) depends on the type of the operands.
- If both operands are integers, C++ performs integer division. That means any fractional part of the answer is discarded, making the result an integer.
- If one or both operands are floating-point values, the fractional part is kept, making the result floating-point.

Type Conversion in C++

- C++ operations occur on same type operands. If operands are of different types, C++ will convert one.
- Implicit conversion do not require any operator. They are automatically performed.

	→ No data loss →	
Lower Data Type	char → short → int → long → float → double → long double	Higher Data Type
	← Data loss ←	

Explicit Type Conversion (Casting)

- Syntax of the explicit type casting
- (type) expression;
- Syntax of the Static Cast
- static_cast < new_data_type> (expression);

Control Flow statements

- The syntax of if-else, switch-case, while loop, do-while loop, for loop, break, continue is exactly the same as you have learned in C.

Questions?