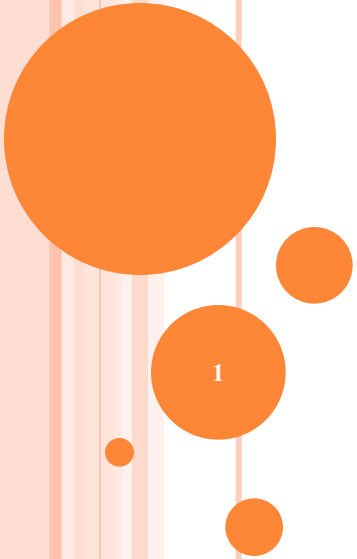
Introduction to C++



Contents

1. Introduction

- C++ improves on many of C's features.
- C++ provides *object-oriented programming (OOP).*
- C++ is a superset to C.
- No ANSI standard exists yet (in 1994).

2. C++ Single-Line Comments

- In C,

/* This is a single-line comment. */

- In C++,

// This is a single-line comment.

3. C++ Stream Input/Output

- In C,

```
printf("Enter new tag: ");
scanf("%d", &tag);
printf("The new tag is: %d\n", tag);
```

- In C++,

```
cout << "Enter new tag: ";
cin >> tag;
cout << "The new tag is : " << tag << '\n';
```

3.1 An Example

```
// Simple stream input/output
#include <iostream.h>

main()
{
    cout << "Enter your age: ";
    int myAge;
    cin >> myAge;

    cout << "Enter your friend's age: ";
    int friendsAge;
    cin >> friendsAge;
```

```
    if (myAge > friendsAge)
        cout << "You are older.\n";
    else
        if (myAge < friendsAge)
            cout << "You are younger.\n";
        else
            cout << "You and your friend are the
  same age.\n";

    return 0;
}
```

4. Declarations in C++

- In C++, declarations can be placed anywhere (except in the condition of a **`while`**, **`do/while`**, **`for`** or **`if`** structure.)
- An example

```
cout << "Enter two integers: ";
int x, y;
cin >> x >> y;
cout << "The sum of " << x << " and " << y
     << " is " << x + y << '\n';
```

- Another example

```
for (int i = 0; i <= 5; i++)
    cout << i << '\n';
```

5. Creating New Data Types in C++

```
struct Name {
        char first[10];
        char last[10];
};
```

- In C,

```
struct Name stdname;
```

- In C++,

```
Name stdname;
```

- The same is true for **enum**s and **union**s

6. Reference Parameters

- In C, all function calls are call by value.
  - Call be reference is simulated using pointers

- *Reference parameters* allows function arguments to be changed without using return or pointers.

6.1 Comparing Call by Value, Call by Reference with Pointers and Call by Reference with References

```
#include <iostream.h>

int sqrByValue(int);
void sqrByPointer(int *);
void sqrByRef(int &);

main()
{
   int x = 2, y = 3, z = 4;

   cout << "x = " << x << " before sqrByVal\n"
        << "Value returned by sqrByVal: "
        << sqrByVal(x)
        << "\nx = " << x << " after sqrByVal\n\n";
```

```
    cout << "y = " << y << " before sqrByPointer\n";
    sqrByPointer(&y);
    cout << "y = " << y << " after sqrByPointer\n\n";


 cout << "z = " << z << " before sqrByRef\n";
    sqrByRef(z);
    cout << "z = " << z << " after sqrByRef\n";

    return 0;
}
```

```
int sqrByValue(int a)
 {
    return a *= a;
     // caller's argument not modified
 }

 void sqrByPointer(int *bPtr)
 {
    *bPtr *= *bPtr;
     // caller's argument modified
 }

 void sqrByRef(int &cRef)
 {
    cRef *= cRef;
     // caller's argument modified
 }
```

Output

```
$ g++ -Wall -o square square.cc

$ square
x = 2 before sqrByValue
Value returned by sqrByValue: 4
x = 2 after sqrByValue

y = 3 before sqrByPointer
y = 9 after sqrByPointer

z = 4 before sqrByRef
z = 16 after sqrByRef
```

7. The Const Qualifier

- Used to declare "*constant variables*" (instead of #define)

```
const float PI = 3.14156;
```

- The const variables must be initialized when declared.

8. Default Arguments

- When a default argument is omitted in a function call, the default value of that argument is automatically passed in the call.
- Default arguments must be the rightmost (trailing) arguments.

8.1 An Example

```
// Using default arguments
#include <iostream.h>

// Calculate the volume of a box
int boxVolume(int length = 1, int width = 1,
              int height = 1)
   { return length * width * height; }
```

```
main()
{
    cout << "The default box volume is: "
         << boxVolume()
         << "\n\nThe volume of a box with length 10,\n"
         << "width 1 and height 1 is: "
         << boxVolume(10)
         << "\n\nThe volume of a box with length 10,\n"
         << "width 5 and height 1 is: "
         << boxVolume(10, 5)
         << "\n\nThe volume of a box with length 10,\n"
         << "width 5 and height 2 is: "
         << boxVolume(10, 5, 2)
         << '\n';

    return 0;
}
```

Output

```
$ g++ -Wall -o volume volume.cc

$ volume
The default box volume is: 1

The volume of a box with length 10,
width 1 and height 1 is: 10

The volume of a box with length 10,
width 5 and height 1 is: 50

The volume of a box with length 10,
width 5 and height 2 is: 100
```

9. Function Overloading

- In C++, several functions of the same name can be defined as long as these function name different sets of parameters (different types or different number of parameters).

9.1 An Example

```
// Using overloaded functions
#include <iostream.h>

int square(int x) { return x * x; }

double square(double y) { return y * y; }

main()
{
    cout << "The square of integer 7 is "
         << square(7)
         << "\nThe square of double 7.5 is "
         << square(7.5) << '\n';
    return 0;
}
```

Output

```
$ g++ -Wall -o overload overload.cc

$ overload
The square of integer 7 is 49
The square of double 7.5 is 56.25
```