Introduction to C++

Introduction to C++

- Structure of C++ Program
- Statements and Semicolon
- Comments
- Character and String Literal
- Output and Input Operator
- Variables and Declarations
- Operators and Precedences
- Integer and Boolean
- Real Number
- C++ String

Structure of C++ Program

- The shortest program:

```
main(){}
```

- Generally,

```
#include <iostream>
using namespace std;
// Hello World Program
int main()
{
        cout << "Hello World.\n";
        return 0;
}
```

C++ Source Code

```
#include <iostream>
using namespace std;
int main()
{
        cout << "Hello World.\n";
        return 0;
}
```

- The #include includes a header file which contains additional functions.
- Every program **must** contain main() function.
- Program statements **must** be in {...} of main()
- All statements **must** be ended by semicolon (;).
- A program **may** return an integer value to OS when exit.

C++ Source Code

- File extension = .cpp
- C++ code is **case-sensitive**.
- Single line comment written after //
- Multiline comment written in between /* and */

```
/*P1
Mr. Dekdee Tangjai
ID: 5712345678-9
*/
#include <iostream>
using namespace std;
// Hello World Program
```

The Output Operator: <<

- Use with the output stream: `cout`.
- Generally, `cout` connects to the screen.
- i.e., everything that has been sent to `cout` will be display on screen.
- Syntax:

cout << exp*1* << exp*2* << ... << exp*n*;

- The output operator will sent expressions – from left to right – orderly to the output stream.

```
cout <<"Hello world\n";
cout <<"Hello " <<"world" <<"\n";
```

```
Hello world
Hello world
```

Characters and String Literals

- A single character is an alphabet, a numeric, or a symbol enclosed within a pair of single quotes, e.g.,
- 'A', 'b', '9', '+'.
- A string literal consists of series of characters within a pair of double quote, e.g.,
- "Hello", "World", "   ".
- Non-printable Characters:

•Newline	'\n'
•Tab	'\t'
•Return	'\r'

Example: Length of String Literals

- Function: strlen() returns the length of string
- <cstring> header must be included in the source file.

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

int main()
{
        cout << strlen("Hello, World.\n") << '\n';
        cout << strlen("Hello, World.") << '\n';
        cout << strlen("Hello, ") << '\n';
        cout << strlen("H") << '\n';
        cout << strlen("") << '\n';
        return 0;
}
```

```
14
13
7
1
0
```

Variables and Declarations

- A variable is a symbol or name referred to a value stored in the memory.
- Variable name always begin with an alphabet or an underscore
- followed by alphanumeric character(s) or underscore
- In C/C++, all variables must be declared before use, e.g.,

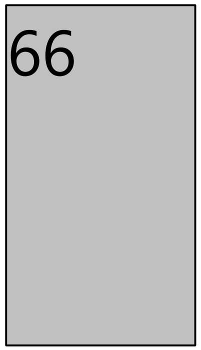
```
int x;
int a1, a2;
char ch, _digit;
string str, first_name, last_name;
```

Assignment Statements

- To assign a value to a variable
- Syntax

```
variable = exp;

// An example to illustrate assignment
int main()
{
        int n;
        n = 66;
        cout << n << endl;
        return 0;
}
```



Initialization

```
// This shows how to initialize
// variable as they are declared:

int main()
{
        int george = 44;
        int martha = 33;
        int sum = george + martha;
        cout << george << " + " << martha
                        << " = " << sum << endl;
  return 0;
}
```

```
44 + 33 = 77
```

Reserved Words / Identifiers

- Reserved words are words with special meaning in C++. **These words cannot be declared to be a variable name**, e.g., include, return, endl, if, switch, while, ...
- Identifier is a name used to declare variables, functions, data types, etc.
- *main* is a function.
- *n* and *cout* are variables.
- *int* is a data type.

Integers

- For **32-bit architecture**:

Types	Bits	Min	Max
char	8	-128	127
unsigned char	8	0	255
short	16	-32,768	32,767
unsigned short	16	0	65,535
int	32	-2,147,483,648	2,147,483,647
unsigned int	32	0	4,294,967,295
long	32	-2,147,483,648	2,147,483,647
unsigned long	32	0	4,294,967,295

Size of Data Type

- Function: sizeof() queries size of the object or type

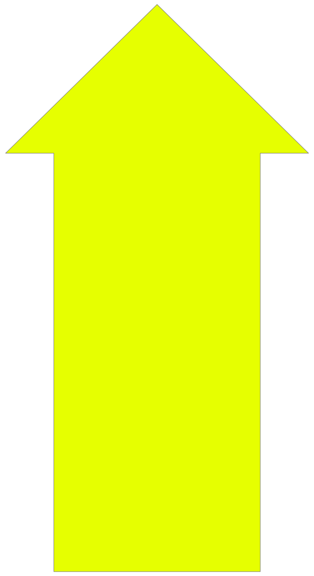
```
•int main()
•{
•  cout<< "Size of char:\t"  << sizeof(char) <<endl;
   cout<< "Size of short:\t" << sizeof(short) <<endl;
   cout<< "Size of int:\t"   << sizeof(int) <<endl;
   char a;
   int b;
   cout<< "Size of a: " <<sizeof(a) <<endl;
   cout<< "Size of b: " <<sizeof(b) <<endl;
•  return 0;
•}
```

```
Size of char:   1
Size of short:  2
Size of int:    4
Size of a: 1
Size of b: 4
```

Arithmetic Operators

Operator	Meaning	Example
++/--	Pre	++n
- Negate	-n	
* Multiply	m * n	
/ Divide	m / n	
%	Remainder	m % n
+ Add	m + n	
– Subtract	m – n	Lower
++/--	Post	n++

Higher



- If precedences are equal, then compute from left to right
- Use parentheses to change precedence

Example: Precedence of Operator

- int x = 1 + 2 - 3 * 4 % 5; // x = 1

```
= 1 + 2 - 3 * 4 % 5
= 1 + 2 - (3 * 4) % 5
= 1 + 2 - (12 % 5)
= 1 + 2 - 2
= (1 + 2) - 2
= 3 - 2
= 1
```

Example: Arithmetic Operator

```
int main()
{
  int m = 38, n = 5;
  cout  << m << " + " << n << " = " << (m + n)   << endl;
  cout  << m << " - " << n << " = " << (m – n)   << endl;
  cout  <<    "    -" << n << " = " << (-n)      << endl;
  cout  << m << " * " << n << " = " << (m * n)   << endl;
  cout  << m << " / " << n << " = " << (m / n)   << endl;
  cout  << m << " % " << n << " = " << (m % n)  << endl;
  return 0;
}
```

```
38 + 5 = 43
38 - 5 = 33
    -5 = -5
38 * 5 = 190
38 / 5 = 7
38 % 5 = 3
```

The result of integer divide integer was converted to integer

Increment and Decrement

- **++** operator increases an integer by one
- Literally the same as `m = m + 1;`
- Pre-increment: `++m`
- Post-increment: `m++`
- **--** operator decreases an integer by one
- Literally the same as `m = m - 1;`
- Pre-decrement: `--m`
- Post-decrement: `m--`

Increment and Decrement – Example

```
int main()
{
    int a=10, b=10, c=10, d=10;

    cout<<"a++ = "<< a++ <<endl;
    cout<<"++b = "<< ++b <<endl;
    cout<<"a = "<< a <<endl;
    cout<<"b = "<< b <<endl;

    int x = c++;
    int y = ++d;
    cout<<"x = "<< x <<endl;
    cout<<"y = "<< y <<endl;

    return 0;
}
```

```
a++ = 10
++b = 11
a = 11
b = 11
x = 10
y = 11
```

Composite Assignment Operators

- Syntax

variable op= expression

variable = variable op expression

- e.g., `n += 8;` is the same as `n = n + 8;`

Expression + Assignment – Example

```
int main()
{
        int n = 44;
        n += 9;
        cout << n << endl;
        n -= 5;
        cout << n << endl;
        n *= 2;
        cout << n << endl;
        return 0;
}
```

```
53
48
96
```

Overflow / Underflow

- Overflow: value > max
- Underflow: value < min
- Divide by zero

Overflow occurs when an arithmetic operation attempts to create a numeric value that is too large to be represented within the available storage space.

```
#include <climits>
int main()
{
        short n = SHRT_MAX - 1;
        cout << n++ << endl;
        cout << n++ << endl;
        cout << n++ << endl;
        cout << n++ << endl;
        return 0;
}
```

```
32766
32767
-32768
-32767
```

Characters

- char can also be considered as an integer

```
char c = 54;    // c = '6'
char d = 2*c–7; // d = 2 * 54 - 7 = 101 = 'e'
c += d % 3;                     // c = c+(101%3) = 54+2 = '8'
```

- char = 8 bits = 1 byte
- Display a character according to the ASCII.

Real Numbers

- Three types
- float – 32 bits
- double – 64 bits
- long double – 64, 80, 96, or 128 bits
- Two components: Mantissa + Exponent
- float: 23-bit mantissa + 8-bit exponent + 1-bit sign
- double: 52-bit mantissa + 11-bit exponent + 1-bit sign
- Floating-point Arithmetic
- Addition, Subtraction, Multiplication, Division.
- Floating-point division does **not** truncate the result.

Type Casting

- All numbers can be casted from one type to another.
- char, short, integer, long, float, double, long double
- Casting can be automatic, or explicit.

```
int main() {
  float f = 2 * 3.14159 * 10;
  int n = 2 * 3.14159 * 10;
  float x = float(1)/3;

  cout << "f = " << f << endl;
  cout << "n = " << n << endl;
  cout << "x = " << x << endl;
}
```

```
f = 62.8318
n = 62
x = 0.333333
```

Round-off Error

- Might occur when computer do arithmetic on rational number
-

```
int main() {
        double x = 1/3.0;
        double y = (x * 3.0) - 1.0;
        cout << "y = " << y << endl;
}
```

```
y = -5.55112e-017
```

should be zero

y = (1/3) * 3 -1 = 0

Precision Display

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
        double x = 1/3.0;
        double y = (x * 3.0) – 1.0;
        cout << "y = " << y << endl;
        cout << std::fixed;
        cout << "y = " <<std::setprecision(5) << y << endl;
        cout << "y = " <<std::setprecision(25)<< y << endl;
        float a = 1/3.0;
        float b = (a * 3.0) – 1.0;
        cout << "b = " <<std::setprecision(25)<< b << endl;
}
```

```
y = -5.55112e-017
y = -0.00000
y = -0.0000000000000000555111512
b = 0.0000000298023223876953125
```

Constants

- An object whose value **cannot** be changed.
- Constants are declared by preceding its type specifier with the keyword const.

```
int main() {
        const float pi = 3.1415927;
// pi = 3.14159; // uncomment will get error
        cout   << "2 x Pi = " << 2 * pi << endl;
}
```

```
2 x Pi = 6.28319
```

Input Operator: >>

- Use with the input stream: cin
- Generally, cin connects to the keyboard.
- i.e., everything that has been typed on the keyboard will be sent to cin.
- Syntax:

cin >> var*1* >> var*2* >> ... >> var*n*;

- The input operator will assign value – from left to right – orderly to the variables.

```
int m, n;
cin >> m >> n;
```

Input Operator – Example

```
int main() {
  int m, n;
  cin >> m >> n;
  cout << "m + n = " << m + n << endl;
  return 0;
}
```

```
5 2
m + n = 7
```

Formatted Input

Input passes through an istream.

Defines behavior of cin.

The most common behavior is the use of the ***extraction*** or ***input operator*** >>.

It has two operands:

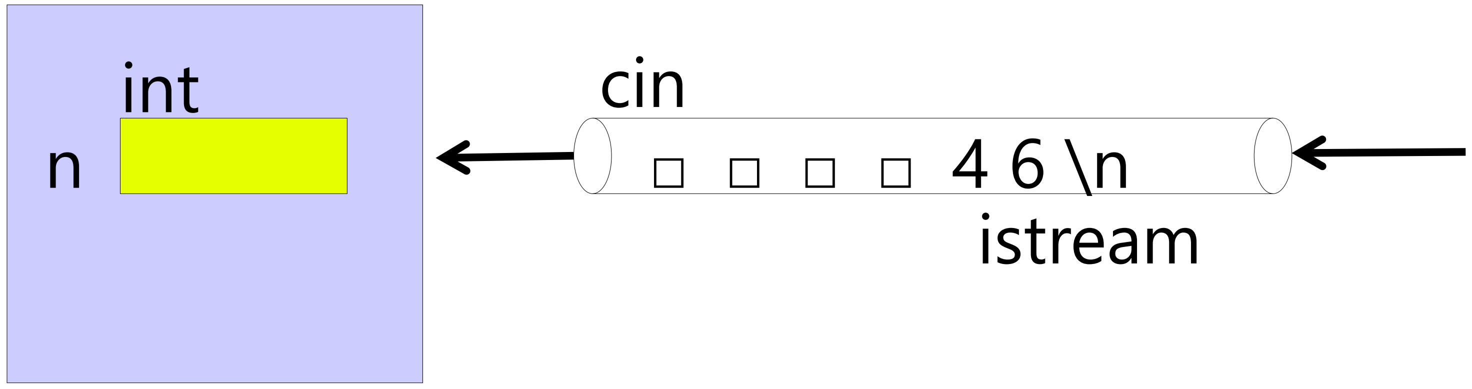
the istream, which is extracting characters

the object to which it copies from those characters.

This process of forming a typed value from raw input character is called ***formatting***.

The Extraction Operator

```
int n;
cin >> n;
```



This input has 7 characters

4 space characters, '4', '6', and '\n'

If the first character is a whitespace, it extracts and ignore. It continues to extract and ignore until it encounters a non-whitespace character.

Since `cin >> n` has type `int`, the `cin` is looking for digits to form an integer.

As soon as it sees a non-digit, it stops.

Boolean Data Type

Use bool data type

It only stores either value 1 or 0,

which represents either YES or NO

Use 0 or false to represent NO

Use non-zero or true to represent YES

Boolean Example

```
int main() {
    bool a=true, b=false;
    bool c = 3 < 4;
    bool d = 3 > 4;
    bool e =0.4, f=0;
    cout << "a = " <<a<<endl;
    cout << "b = " <<b<<endl;
    cout << "c = " <<c<<endl;
    cout << "d = " <<d<<endl;
    cout << "e = " <<e<<endl;
    cout << "f = " <<f<<endl;
  return 0;
}
```

```
a = 1
b = 0
c = 1
d = 0
e = 1
f = 0
```

Alphanumerical bool Values

```
int main () {
  bool b = true;
  cout << std::boolalpha << b << '\n';
  cout << std::noboolalpha << b << '\n';
  return 0;
}
```

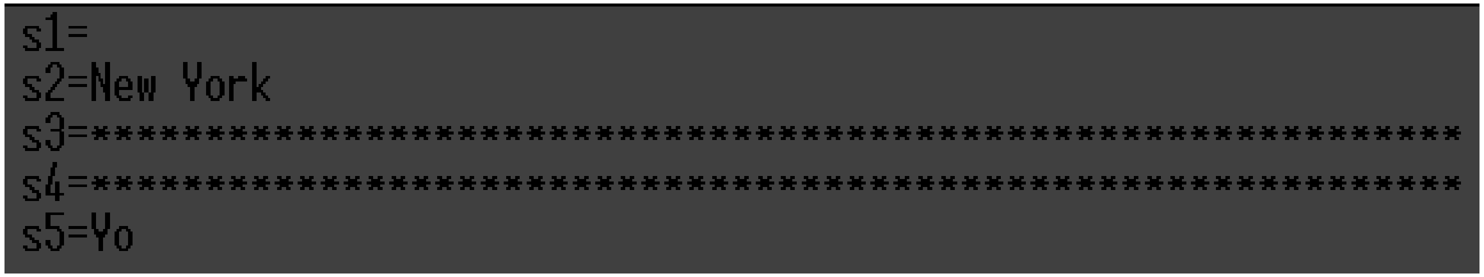
```
true
1
```

The Standard C++ String Type

Standard C++ defines its `string` type in the `<string>` header.

Objects of type `string` can be declared and initialized in several ways:

```
string s1;
string s2 = "New York";
string s3(60, '*');
string s4 = s3;
string s5(s2, 4, 2);
```



```
s1=
s2=New York
s3=************************************************************
s4=************************************************************
s5=Yo
```

If the string is not initialized, it represents the empty string.

Basic String Usage

Input one word to each string

+ operator used for concatenating string

```
#include <string>
 int main () {
string firstname, lastname;
cout <<"What is your name?\n";
cin  >> firstname >> lastname;
cout <<"Hello " << firstname <<".\n";
string fullname = firstname + " " + lastname;
cout <<"Your full name is "<< fullname <<".\n";
return 0;
 }
```

```
What is your name?
Tom Jerry
Hello Tom.
Your full name is Tom Jerry.
```

Return and Exit

exit(int *status*) used to terminate application

must include <cstdlib>

return used to escape from function

If **return in main()** it cause application termination

When application return 0 to OS meaning normal termination

If return **non-zero** there may be something wrong

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main () {
    cout << "Hello world.\n";
   exit(1); //terminate here
   cout << "How are you?";
   return 0;
}
```

```
Hello world.
```

Return Many Times

Not require additional library <cstdlib>

```
 #include <iostream>
// #include <cstdlib>
 using namespace std;
 int main () {
     cout << "Hello world.\n";
    return 1; //terminate here
    cout << "How are you?";
    return 0;
}
```

```
Hello world.
```