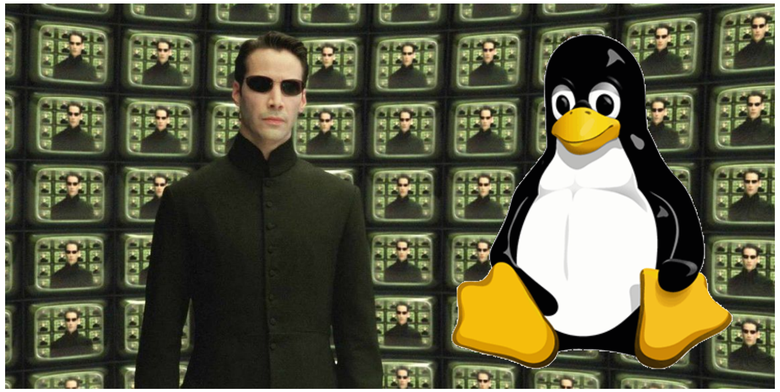
01.06.20 19:03

CSCI 125
Introduction to Computer Science and Programming II
Lecture 1: Your First C/C++ Programme II



Jetic Gū

2020 Summer Semester (S2)

Overview

- Focus: Basic C/C++ Syntax
- Architecture: Linux/Unix OS
- Core Ideas:
    1. Systematic Introduction of C++: basic components
    2. Simple data types
    3. iostream

P0
Review

Your First C++ Programme

- Hello World: `include`; `cstdio`; `int main()`
- A plus B: `for` loops; conditions

Review

Basic Components

Systematic Introduction

Summary

P1
Components

The Basics of a C++ Program

- **Functions**
  Collections of instructions, each designed to accomplish certain things

- **Syntax**
  Rules that define legal instructions

- **Semantic rule**
  Meaning of the instruction

- **Programming language**
  A set of rules, symbols, and special words

Concept

Comments

- Comments are for the reader, not the compiler
- Two types:
  - Single line

```
// This is a C++ program. It prints the sentence:
// Welcome to C++ Programming.
```

  - Multiple line

```
/*
   You can include comments that can
   occupy several lines.
*/
```

Concept

P1
Components

Special Symbols and Keywords

- Special symbols

```
+        ?
-        ,
*        <=
/        !=
.        ==
;        >=
:
```

- Reserved words, keywords
  - `int`
  - `if, else`
  - `return`
  - `true, false, bool`

Concept

Identifiers

- Consist of letters, digits, and the underscore character (_)
- Must begin with a letter or underscore
- C++ is case sensitive
  - `NUMBER` is not the same as number
- Two predefined identifiers are `cout` and `cin`
- Unlike reserved words, predefined identifiers may be redefined, but it is not a good idea

Concept

P1
Components

Identifiers

Illegal Identifier	Description
Cheese Burger	Whitespace is not allowed
Hello!	Cannot contain special symbol (underscore is ok though)
A+B	Cannot contain special symbol (underscore is ok though)
2nd	Cannot begin with a digit

- The following are legal identifiers in C/C++:
  - `cheese`
  - `Burger`
  - `cheese_burger`
  - `cheeseBurger`

Concept

Whitespaces

- Every C++ program contains whitespaces
  - Blanks `' '`, tabs `'\t'`, and newline `'\n'` characters
- Used to separate special symbols, reserved words, and identifiers
- Readability through indentation
  - Not "strictly required" like in Python though

Concept

Preprocessor Directives

- C++ has a small number of operations
- Many functions and symbols needed to run a C++ program are provided as collection of libraries
- Every library has a name and is referred to by a header file
- Preprocessor directives are commands supplied to the preprocessor
- All preprocessor commands begin with #
- No semicolon at the end of these commands

Concept

Preprocessor Directives

- Syntax to include a header file:

```
#include <headerFileName>
```

- For example:

```
#include <iostream>
```

  - Causes the preprocessor to include the header file `iostream` in the program

Concept

Data Types

Primary Stuff

Summary

Data Types

- Primary
  - Integers, Characters, Boolean
  - Floating point
  - Void
- Derived
  - Function, Array, Pointer, Reference
- User Defined
  - Struct, Class, Enumerate, Typedef

Concept

Primary Data Types

- Integers, Characters, and Boolean
  - `char`, `unsigned char`
  - `short`, `unsigned short`
  - `int`, `unsigned int`
  - `long`, `unsigned long`
  - `bool`

Concept

Primary Data Types

Type	Storage	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int (long)	4	-2,147,483,648 to 2,147,483,647
unsigned long int (unsigned long)	4	0 to 4,294,967,295
long long int (long long)	8	-(2^63) to (2^63)-1
unsigned long long int (unsigned long long)	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255

Primary Data Types

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

int main()
{
    cout << "Size of char : "             << sizeof(char)             << " byte" << endl;
    cout << "Size of int : "              << sizeof(int)              << " bytes" << endl;
    cout << "Size of short int : "        << sizeof(short int)        << " bytes" << endl;
    cout << "Size of long int : "         << sizeof(long int)         << " bytes" << endl;
    cout << "Size of signed long int : "  << sizeof(signed long int)  << " bytes" << endl;
    cout << "Size of unsigned long int : " << sizeof(unsigned long int) << " bytes" << endl;
    cout << "Size of float : "            << sizeof(float)            << " bytes" <<endl;
    cout << "Size of double : "           << sizeof(double)           << " bytes" << endl;
   return 0;
}
```

Boolean

- **`bool`** type
  - Two values: **`true`** and **`false`**
  - Manipulate logical (Boolean) expressions
  - 1 byte

Concept

`char`

- Used for ASCII code characters: letters, digits, and special symbols
- Each character is enclosed in single quotes
  - `'A'`, `'a'`, `'0'`, `'*'`, `'+'`, `'$'`, `'&'`
- A blank space is a character and is written `' '`, with a space left between the single quotes
- Can be used as **lesser version of `int`** as well

```
'0' + 1 == '1', 'B' + 1 == 'C', 'A' == 65
```

Concept

Floating-Point Data Types

- C++ uses scientific notation to represent real numbers (floating-point notation)
  - 314.159 = 3.14159×10^2 = `3.141590E2`
- `float`: single precision
  - Range: -3.4E+38 to 3.4E+38 (4 bytes)
- `double` (`long double`): double precision
  - Range: -1.7E+308 to 1.7E+308 (8 bytes)

Concept

Simple Arithmetics

It’s just math

Summary

P3
Arithmetics

Arithmetic Operators and Operator Precedence

- C++ arithmetic operators are function defined for specific data types
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % modulus operator

- +, -, *, and / can be used with integral and floating-point data types

Concept

Order of Precedence

- Same as in Math
  1. All operations inside of ()
  2. *, /, and %
  3. +, –
  4. When operators are on the same level, left to right (associativity)
- `3 * 7 - 6 + 2 * 5 / 4 + 6` means

```
(((3 * 7) – 6) + ((2 * 5) / 4 )) + 6
```

Concept

Type Conversion (Casting)

- **Implicit**
  value of one type is automatically changed to another type (during value assignment and arithmetic calculation)

- **Explicit**
  e.g. `int(6.7) == 6    // No rounding!`

  - How to implement rounding?

Concept

Increment & Decrement Operators

- Increment operator: increment variable by 1
  - Pre-increment: `++variable`
  - Post-increment: `variable++`

- Decrement operator: decrement variable by 1
  - Pre-decrement: `--variable`
  - Post-decrement: `variable--`

- What is the difference between the following?

```
x = 5;
y = ++x;
```

```
x = 5;
y = x++;
```

Concept

More on Assignment Statements

- C++ has special assignment statements called compound assignments `+=`, `-=`, `*=`, `/=`, and `%=`

- Example:

```
x *= y;
```

Concept

iostream

C++'s Standard Way of Implementing IO

Summary

Basic Intro

- The Standard C runtime library uses stdio.
  In C, it's `<stdio.h>` it's `<cstdio>` in C++

- The Standard C++ runtime library uses `<iostream>`

- Why?

  - `<iostream>` is more powerful and easier to use, provides better support for complex data types such as **`string`**

Concept

namespace

- In Python, you have to specify the library when using its component

```
import sys
sys.call(...)
```

- In C++, when you include a library, all its functions are available for calling

```
#include <cstdio>
...
printf(...);
```

- What if you have two libraries containing the same identifier? --> confusion
  - Use `namespace` to differentiate!

Concept

std::cin

- uses `std` namespace
- `std::cin` is used with `>>` to gather input

```
cin >> variable >> variable ...;
```

- The stream extraction operator is >>
- For example, if miles is a double variable
  `cin >> miles;`
  - Automatically causes computer to get a value of type `double`

Concept

std::cin

- Using more than one variable in `cin` allows more than one value to be read at a time
- For example, if `feet` and `inches` are variables of type `int`, a statement such as

```
cin >> feet >> inches;
```

  - Inputs two integers from the keyboard
  - Places them in variables `feet` and `inches` respectively

Concept

std::cout

- uses `std` namespace
- `std::cout` is used with << to output

```
cout << expression or manipulator << expression or manipulator...;
```

- The stream extraction operator is <<
- Automatic recognition of variable type during output

Concept

Namespace

- If you don’t want to type `std::` all the time
  - Declare default namespace using
    `using namespace std;`
  - Usually after Preprocessor Directives

Concept

Hello World (again)

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!" << endl;  // endl stands for '\n'
    return 0;
}
```