Contents

Lecture 01 - Introduction

I. Course Introduction
II. Introduction to Computer Programming
III. Simple C++ Programs
IV. Topic Discussion
V. Checkpoints and Your Tasks

I. Course Introduction

- Lecturers; Course Syllabus and ILOs; Course Arrangements [Please refer to Canvas]

- Weekly Time Table:

<table>
<thead>
<tr>
<th>Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-09:50</td>
<td>CS2310-C01</td>
<td>CS2310-C01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00-10:50</td>
<td>CS2310-C01</td>
<td></td>
<td>CS2310-C01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00-11:50</td>
<td>MMW 2450</td>
<td>MMW 2450</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00-12:50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:00-13:50</td>
<td>CS2310-L01</td>
<td></td>
<td>CS2310-L01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:00-14:50</td>
<td>MMW 2450</td>
<td>MMW 2450</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:00-15:50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:00-16:50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Note: To pass the course, at least 40% of the coursework (i.e. continuous assessment) and 30% of the examination must be obtained.

- References:
  1. Some well known sites:
     - [http://www.cprogramming.com](http://www.cprogramming.com)
  2. Books:
     - Walter Savitch. Problem Solving with C++
     - H.M. Deitel & P.J. Deitel. C++ How to Program
     - Stanley B. Lippman, Josee Lajoie, Barbara E. Moo. C++ Primer
     - Dale and Weems Programming and Problem Solving with C++: Comprehensive

An Important Rule: Respect the class during lessons

It is disturbing to others if:
You go out or come in during a lesson.

If you decide to leave the classroom temporarily, please kindly stay outside and then come back during the break only.
Do assignments and exercises ON Your Own

"On your own" means
- discuss the problems with any other people.
- study materials available on the internet.
- refer to any book.

But the details and write-up must be entirely your work.
The principle is: Students should gain through practicing and developing skills in doing your work.

The PASS System – for doing exercises and assignments
** DON’T upload other people’s code with your account **

FAQ

Question 1
Is programming interesting?  
Is programming easy?

Answer:
"If you enjoy the game and be serious with the rules, you won’t find it difficult."

Question 2
CS2310 - How to study successfully?

Sincere warnings:
- To study from sample programs, don’t just read. Digest and re-do to try it or edit the code for what-if tests.
- Do not “pile-up” questions to “next-week”!
- 😊 Good progress in this lesson => good foundation for the next  
  😖 Poor progress in this lesson => get lost during next lesson

Tips:
- Lecture: 10%
- Revision: 10%
- Reading: 10%
- Programming (coding): 35%
- Debugging (handling errors): 35%

You’ll learn from ALL of the above.

First programming courses in universities usually focus on programming for console-mode applications.

Reason: Developing non-console mode applications usually involve much use of third-party tools. Thus not suitable for students to practice solid programming skills from the foundation.

You should not create any chance for other students to copy your work.

For any plagiarism case,
- The student who plagiarizes will be punished.
- Any student who allows his/her work to be copied will also be punished.
II. Introduction to Computer Programming

Who writes programs in a Typical IT Department?

The Quality of a Program
- Correctness and Performance / Efficiency
- Requirement of resources (Large memory? Powerful computer?)
- Programming style / Clarity, Simplicity, Generality: The code is Easy to read and Easy to maintain? (Upgraded by the original programmer or other colleagues?)
- Robust (Can handle "all" cases in different situations correctly?)
- Easy to migrate to other computers or database systems later? (eg. Windows ⇆ Linux)

The C++ Program Language
- C
  ➢ developed in 1970’s
  ➢ originally for writing system programs such as OS (eg. UNIX) and compilers
  ➢ “close to machine”
- C++
  ➢ developed in 1980’s
  ➢ C enhanced with object-oriented features, for more complex applications.
  ➢ “close to the problems to be solved”

E.g., Use Microsoft Word!!

When a program executes, both the program executable code and data are in the main memory.

E.g., Use Microsoft Word to open letter1.doc

(1) From program files (winword.exe etc..), the machine code is brought to the main memory, then the CPU executes the machine code of the program.

(2) When letter1.doc is opened, the content of this file is also brought to the main memory (controlled by the machine code).
Main Memory

- Bit (Binary Digit - stores either 0 or 1)
- 1 byte : 8 bits
- Each byte in the main memory is associated with an address
  - 1 K bytes = $2^{10}$ bytes = 1024 bytes
  - 64K bytes of memory = 65536 bytes

![Memory Diagram]

We start counting from 0 instead of 1.

Example:

The program below inputs the user’s name and age, and then shows the fee required (Child: $10, Adult:$20).
The following are the screen-dumps of 2 test runs:

![Screen Dumps]

An enhanced version of the program can show the memory contents where the users' data reside:

![Enhanced Memory]

The user's age

The characters in the user’s name
(eg. at address 0012FF66, the character 'l' is stored as the number 108₁₀, or 6C₁₆ or 01101100₂)

Note that the computer treats all kinds of data as numbers (actually binary numbers).
For text data, characters are encoded as numbers using the ASCII coding scheme:

<table>
<thead>
<tr>
<th>ASCII CODE</th>
<th>'a'-'z'</th>
<th>97-122</th>
</tr>
</thead>
<tbody>
<tr>
<td>'A'-'Z'</td>
<td>65-90</td>
<td></td>
</tr>
</tbody>
</table>

* American Standard Code for Information Interchange (ASCII)
Computer Programming

- The instructions in the programs must be:
  - Workable
  - Detail and clear to the computer (The computer is stupid.)

- The process of creating a C++ program:

  ![Diagram of C++ program creation process]

  **Visual Studio** is an integrated development environment (involves source code editor, building tools, and project management facilities) that supports multiple programming languages. It includes C++ compiler, C compiler, C# compiler, Visual Basic interpreter, etc.

  - First, the programmer writes the source code in the C++ files.
    E.g., Student.cpp, Course.cpp, EnrollSys.cpp

  - Then the programmer builds the program executable file using software building tools: compiler and linker
    - **Compilation**
      The compiler checks the grammatical rules (syntax) and converts the source code into object code. E.g., Student.obj, Course.obj, EnrollSys.obj
      The object code contains the binary code in **machine language** of the target computer system (e.g. Windows).
    - **Linking**
      The linker combines object code to give the executable, e.g. EnrollSys.exe.
      A necessary code routine, main(), must exist in the object code.

  - If the source code has any syntax or potential problem, the compiler will give
    1. **compilation warning**, or
    2. **compilation error** (compilation stops, i.e. you won’t get the .obj or the .exe file)

  - After successful compilation and linking, the program **executable file** is produced. We can run this executable file. However, you may still get
    1. **run-time error** (Program halts during run-time due to an invalid statement or operation, e.g., divide by zero, endless loop)

  - After the program has started to run and finished normally, we may still get
    1. **logic error** (design or programming mistake that, although not causing problem during execution, but but produces incorrect result)
### III. Simple C++ Programs

**Example 1:**

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

int main()
{
    cout << "Hello, world!" << endl;
    cout << "How are you?" << endl;
    return 0;
}
```

- **Line 1:** "#include <iostream>" is called a **preprocessing directive**.
  Here it tells the compiler that we will use some things (here std and cout) defined in the **iostream** library.

- **Line 2:** "using namespace std;" is called a **using directive**. It lets us use cout conveniently.

- Also note:
  - "int main() { .. }" is called the **main function**. The statements inside it will decide what steps the program will do.
  - Case-sensitive – don't write Main or COUT etc.. 😫
  - Use of semi-colons: ; at the ends of statements.
  - Extra whitespaces will not affect compilation result (tab, space, new line).
    - **We SHOULD** use proper whitespaces to increase readability.
      - Eg., at line 6-8, we use **tab** to apply **indentation** to the code:

```
  1 2 3 4 5 6 7 8 9 0  
  w e r t y u i o p a s d f g h j k l
  g t y : n m - . / 2 3 4 5 6
  ` ---
  The tab key
  
  - int main()
  { 
    cout << "Hello, world!" << endl;
    cout << "How are you?" << endl;
    return 0;
  }
```

- **Exit program (by convention, 0 means “exit normally”)**
  - "cout << "" : **Output statement**
  - "endl" : writes a new line.

- **Lines 6 to 7 can be combined as one statement:**
  - cout << "Hello, world!" << endl << "How are you?" << endl;
    - This statement is long. We can split it across several lines like the above.

- **If line 1 is removed, we get compilation error at lines 2, 6-7**
  - Reason: The compiler doesn’t know what are std and cout.

- **Don’t add ; after line 4:**
  - Don’t add ; after line 1: #include..

- **For indentation, “Using spacebar instead of tab” is NOT a good habit.**
  - For indentation, "Using spacebar instead of tab" is NOT a good habit.
  - Don’t add ; after line 4: int main()
There are totally ______ statements in this main function. When the program runs, these statements are executed one by one in sequence.

There is/are totally _____ output statement(s): line(s) ____.

There is/are totally _____ input statement(s): line(s) ____.

At line _____, a variable, named 'n', is defined. It is used to store the user's input.

In fact, by defining variables, memory locations are reserved when the program runs.

- We use the variable name to refer to the data value stored at the memory location.
- In "int n;", "int" means integer data type: "The variable n is used to store an integer data value".

** We will learn more about data types in next lecture.

---

**IV. Topic Discussion:**


Q2. “Compiling” vs “Building a program”:

“Compilation” is only one (a very important) part of “building”.
But programmers often say “compile a program” when they actually want to say “build a program”. Why? ☺

Q3. What’s wrong if a student says: “When I run the program, the compiler gives me a run-time error.”
Q4. We are to study **compilation warning**, **compilation error**, **run-time error**, **logic error** using the programs below. For each of them, we will first compile it. If the compilation is successful, we can run it.

<table>
<thead>
<tr>
<th>Program (a)</th>
<th>Program (b)</th>
<th>Program (c)</th>
</tr>
</thead>
</table>
| #include <iostream>  
using namespace std;  
int main() {  
    int x;  
    int y;  
    cout << "Input x: ";  
    cin >> x;  
    cout << "100 / x is: ";  
    << 100/x << endl;  
    return 0;  
} | #include <iostream>  
using namespace std;  
int main() {  
    int x;  
    cout << "Input x: ";  
    cin >> x;  
    cout << "100 / x is: ";  
    << 100/x << endl;  
    return 0;  
} | #include <iostream>  
using namespace std;  
int main() {  
    int x;  
    cout << "Input x: ";  
    cin >> x;  
    cout << "100 / x is equal to: ";  
    << 0.01*x << endl;  
    return 0;  
} |

**Compilation warning:** Program _____ causes a compilation warning: ____________________________

The executable can still be generated. If we run it and type 3 for x, The output is: _______

**Compilation error:** Program _____ causes a compilation error: ____________________________

The executable cannot be generated. We cannot run the program.

**Run-time error:** If we run program _____ and type ___ for x, we get a run-time error.

**Logic error:** Program ____ can compile successfully. When we run the program, it can always behave normally and finish normally. However, the output result is wrong.

Q5. Copying code from a pdf file may be problematic. [Refer to the teacher’s demonstration]

When we copy and paste the program code on the right from the contents in the pdf file to Visual Studio as a C++ program and compile it. We get the following message box. What’s wrong?

![Message box](image)

We get a **compilation warning** at line 8 and a **compilation error** at line 9. Why does the compiler give different results for these two lines?

**[Line 8] Compilation warning:** ____________________________

**[Line 9] Compilation error:** ____________________________
Q6. Below are 3 very common Linking Errors. Match them to the causes. [You may need to take a guess]

<table>
<thead>
<tr>
<th>Linking error</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>function int main() already has a body</td>
<td>More than one main() function is written:</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>int main()</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>int main()</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>fatal error: cannot open ...\xxx.exe for writing</td>
<td>Missing a main() function in the program</td>
</tr>
<tr>
<td>unresolved symbol _main referenced ...</td>
<td>We start running the program:</td>
</tr>
<tr>
<td></td>
<td>We suddenly want to change the code, so we go back to VS i.e. Visual Studio, edit the code. Then we re-compile the code and wait for VS to build the new version of executable file. (However, the old version is running!)</td>
</tr>
</tbody>
</table>

V. Checkpoints and Your Tasks

In this lesson, we have covered the introduction of the course and some concepts on programming.

Summary / Checkpoints of this Lecture
- Programming Languages and Compilation
- Console-mode Programs
- Program source file vs Binary executable file
- Importance of good quality of programming
- Data and Main Memory
- Simple C++ programs, I/O (i.e. input/output)

Before next lecture:
- Review this lecture sheet; attend Lab and finish the Take-home exercises

--- end ---