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:

Time	Mon	Tue	Wed	Thu	Fri
09:00-09:50					
10:00-10:50	CS2310-C01			CS2310-C01	
11:00-11:50	MMW 2450			MMW 2450	
12:00-12:50					
13:00-13:50	CS2310-L01			CS2310-L01	
14:00-14:50	MMW 2450			MMW 2450	
15:00-15:50					
16:00-16:50					

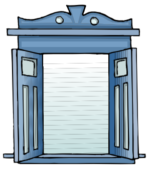
- Note: To pass the course, at least <u>40% of the coursework</u> (i.e. continuous assessment) and <u>30% of the examination</u> must be obtained.

- References:

1. Some well known sites:
    - https://msdn.microsoft.com/en-us/library/3bstk3k5.aspx
    - http://www.cplusplus.com/doc/tutorial
    - http://www.cprogramming.com
    - http://www.learncpp.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

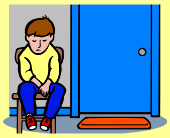
An Important Rule: Respect the class during lessons



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.

Deserved mark?

Unfair situation ✗

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.

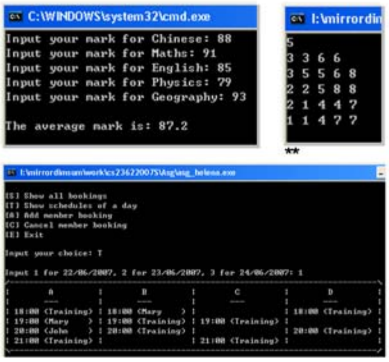
The **PASS** System – for doing exercises and assignments

** DON'T upload other people's code with your account ☠ **

What kind of programs to write?

(i) Console Mode Applications ?

"text only"



```
C:\WINDOWS\system32\cmd.exe
Input your mark for Chinese: 88
Input your mark for Maths: 91
Input your mark for English: 85
Input your mark for Physics: 79
Input your mark for Geography: 93

The average mark is: 87.2
```

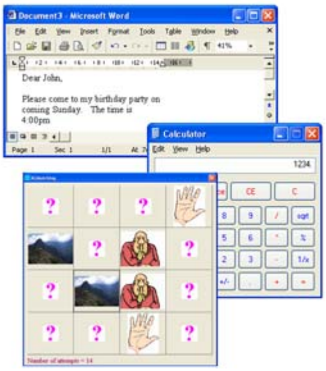
```
I:\mirrordin
5
3 3 6 6
3 5 5 6 8
2 2 5 8 8
2 1 4 4 7
1 1 4 7 7
```

**

```
I:\mirrordimsum\work\cs23622007S\Asg\asg_helena.exe
[S] Show all bookings
[T] Show schedules of a day
[A] Add member booking
[C] Cancel member booking
[E] Exit

Input your choice: T

Input 1 for 22/06/2007, 2 for 23/06/2007, 3 for 24/06/2007: 1
/-----------------------------------------------------------------------------\
|        A          |        B          |        C          |        D          |
|       ---         |       ---         |       ---         |       ---         |
| 18:00 (Training)  | 18:00 (Mary    )  |                   | 18:00 (Training)  |
| 19:00 (Mary    )  | 19:00 (Training)  | 19:00 (Training)  |                   |
| 20:00 (John    )  | 20:00 (Training)  |                   | 20:00 (Training)  |
| 21:00 (Training)  |                   | 21:00 (Training)  |                   |
\-----------------------------------------------------------------------------/
```

(ii) Windows Applications ? Or ...

"Graphics mode"



Document3 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Calculator
Edit View Help
1234
CE C
8 9 / sqrt
5 6 * %
2 3 - 1/x
+/- . + =

Dear John,

Please come to my birthday party on coming Sunday. The time is 4:00pm

Number of attempts = 14

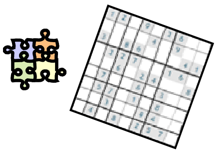
First programming courses in universities usually focus on programming for *console-mode applications*.

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.

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."

- You enjoy completing the game by yourself.
- You are curious about every error. You want to understand and solve it.
- After knowing how other people solve a problem, you also want to try it out on your own.

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

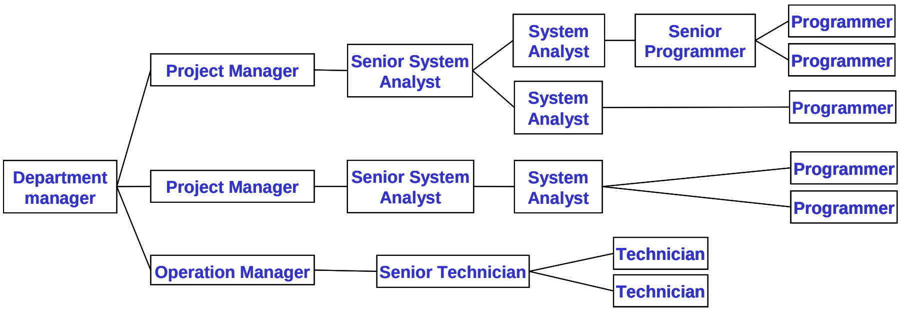
Tips:

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

You'll learn from ALL of the above.

II. Introduction to Computer Programming

Who writes programs in a Typical IT Department?



"What is the importance of programming knowledge and practical skills of the staff?"

The Quality of a Program

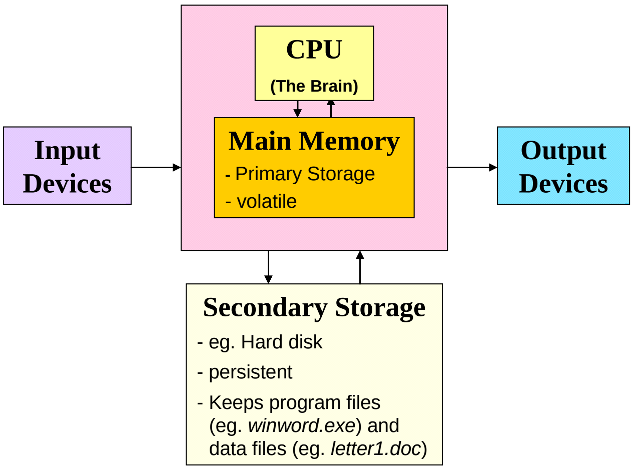
- Correctness and Performance / Efficiency
- Requirement of resources (Large memory? Powerful computer?)
- Programming style / Clarity, Simplicity, Generality: The code is **Easy to read** and **Easy to maintain**? (Upgraded by the original programmer or other colleagues?)
- **Robust** (Can handle "all" 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"

← *Eg. Microsoft Word!!*

Stored-Program Computer (Also called von Neumann machines)



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

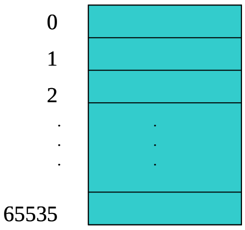
Eg., Use *Microsoft Word* to open *letter1.doc*

(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.

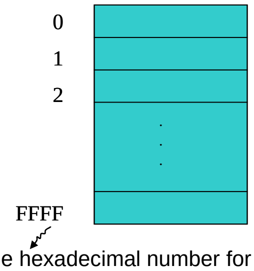
(2) When *letter1.doc* is opened, the content of this file is also brought to the main memory (controlled by the machine code).

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



or

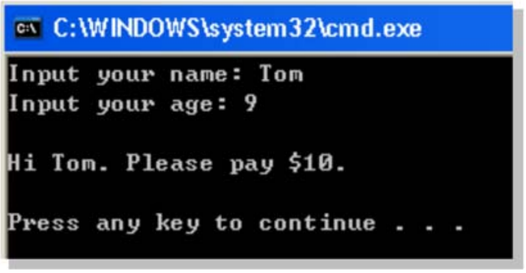


We start counting from 0 instead of 1.

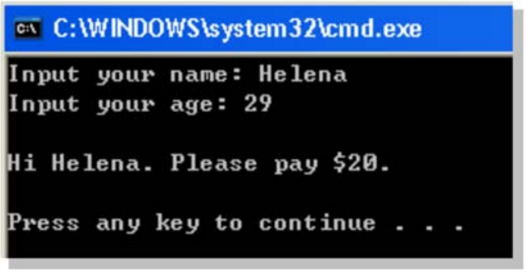
FFFF is the hexadecimal number for 65535. (Hexadecimal digits are: 0,1,2,3,4,5,6,7,8,9,A, B,C,D,E,F)

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:



```
Input your name: Tom
Input your age: 9

Hi Tom. Please pay $10.

Press any key to continue . . .
```

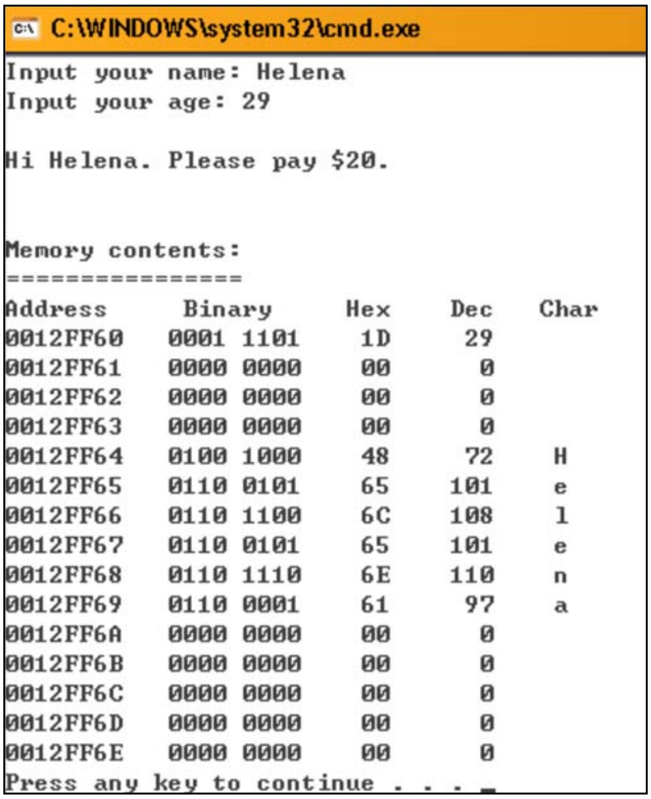


```
Input your name: Helena
Input your age: 29

Hi Helena. Please pay $20.

Press any key to continue . . .
```

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

The user's age

The characters in the user's name (eg. at address 0012FF66, the character 'l' is stored as the number 108_{10}, or $6C_{16}$ or 01101100_2)



```
Input your name: Helena
Input your age: 29

Hi Helena. Please pay $20.


Memory contents:
================
Address       Binary      Hex    Dec    Char
0012FF60    0001 1101     1D     29
0012FF61    0000 0000     00      0
0012FF62    0000 0000     00      0
0012FF63    0000 0000     00      0
0012FF64    0100 1000     48     72      H
0012FF65    0110 0101     65    101      e
0012FF66    0110 1100     6C    108      l
0012FF67    0110 0101     65    101      e
0012FF68    0110 1110     6E    110      n
0012FF69    0110 0001     61     97      a
0012FF6A    0000 0000     00      0
0012FF6B    0000 0000     00      0
0012FF6C    0000 0000     00      0
0012FF6D    0000 0000     00      0
0012FF6E    0000 0000     00      0
Press any key to continue . . . _
```

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:

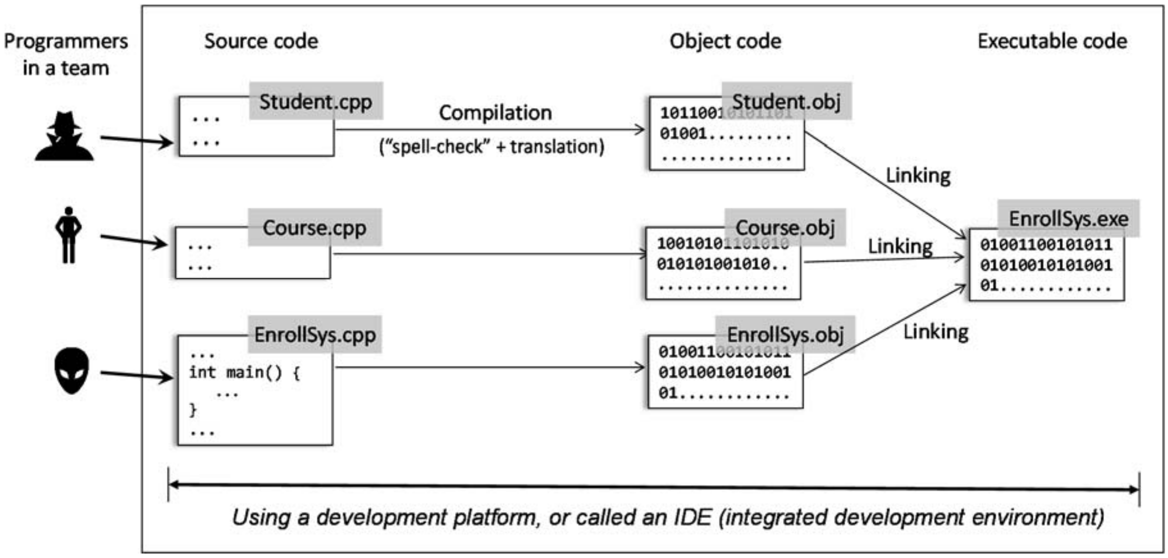
ASCII* CODE 'a'-'z' : 97-122
'A'-'Z' : 65-90

* 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:



***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 (eg. 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, ie. you won't get the `.obj` or the `.exe` file)

There are also **linking errors**. One common linking error is like this: "unresolved symbol _main referenced in .."

It happens when the linker cannot find a main() routine in any of the object code.

- After successful compilation and linking, the program **executable file** is produced. We can run this executable file. However, you may still get

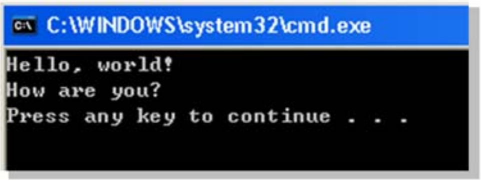
  (3) **run-time error** (Program halts during run-time due to an invalid statement or operation, e.g., divide by zero, endless loop)

- After the program has started to run and finished normally, we may still get

  (4) **logic error** (design or programming mistake that, althought not causing problem during execution, but but produces incorrect result)

III. Simple C++ Programs

Example 1:



```
Hello, world!
How are you?
Press any key to continue . . .
```

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hello, world!" << endl;
7      cout << "How are you?" << endl;
8      return 0;
9  }
```

- include library <iostream>
- use the "std" namespace

Exit program (by convention, 0 means “exit normally”)

- "cout << …" : ***Output statement***
- "endl" : writes a new line.

- 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:

Tab
CAPS
Return
Ctrl
Win
Alt
Alt Gr
Menu

The **tab** key

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

- This program executes the following steps:

(1) output "Hello, world!", then a new line (<enter>)

(2) output "How are you?", then a new line (<enter>)

"Press any key to continue…" is generated by Visual Studio.

- 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**.

- Line 2 can be removed. But we will need to write "**std**::**cout**" instead of "**cout**" at lines 6-7.

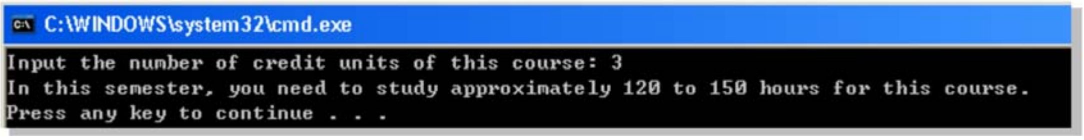
- Don't add ; after line 1:
#include..

Don't add ; after line 4:
int main()

- For indentation, "*Using **spacebar** instead of **tab***" is **NOT** a good habit.

Start practicing to use the tab key right now!!

Example 2:



```
Input the number of credit units of this course: 3
In this semester, you need to study approximately 120 to 150 hours for this course.
Press any key to continue . . .
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int n;
7       cout << "Input the number of credit units of this course: ";
8       cin >> n;
9       cout << "In this semester, you need to study approximately ";
10      cout << 40 * n;
11      cout << " to ";
12      cout << 50 * n;
13      cout << " hours for this course." << endl;
14      return 0;
15  }
```

- Define a ***variable*** (**n**) to store user's input.
- "cin >> …" : ***Input statement***
  - Wait for the user to input a value and use n to store it.
  - The user needs to type an integer and press <Enter>.

- 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:

Q1. What does “cout” stand for? Hint: You can guess what “c” means by reading page 2.

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.

Program (a)

```
#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;
}
```

Program (b)

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

int main() {
    int x;
    cout << "Input x: ";
    cout << "100 / x is: "
         << 100/x << endl;
    return 0;
}
```

Program (c)

```
#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.

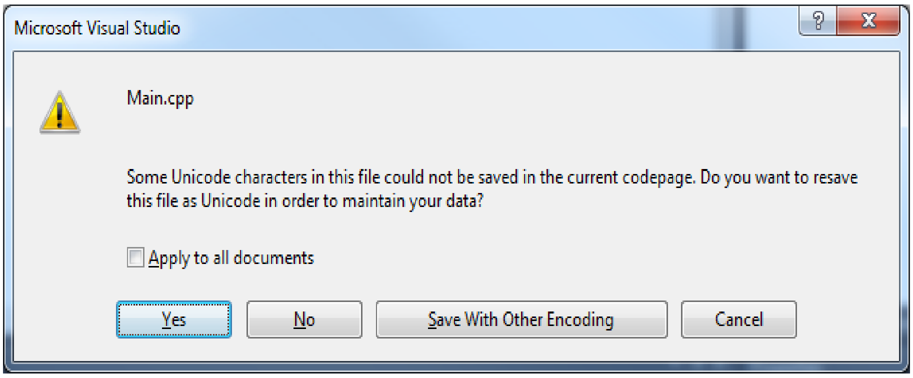
Run-time error: If we run program ______ and type ___ for x, we get a run-time error.

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?

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int x;
6       cout << "Input x: ";
7       cin >> x;
8       cout << "100 - x is ";
9       cout << 100-x;
10      cout << endl;
11      return 0;
12  }
```



Microsoft Visual Studio

Main.cpp

Some Unicode characters in this file could not be saved in the current codepage. Do you want to resave this file as Unicode in order to maintain your data?

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]

Linking error

Cause

function int main() already has a body ● ●

More than one `main()` function is written:

Src1.cpp

```
...
int main()
{
  ...
  ...
}
```

Src2.cpp

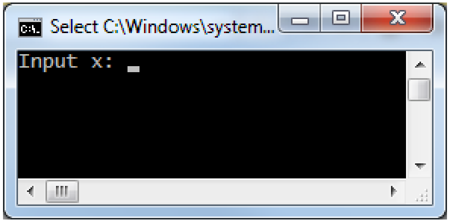
```
...
int main()
{
  ...
  ...
```

fatal error: cannot open ...\xxx.exe for writing ● ●

Missing a `main()` function in the program

unresolved symbol _main referenced ... ● ●

We start running the program:



```
Input x: _
```

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!)

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 (ie. input/output)

Before next lecture:

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

--- end ---