COMP155: Week 1

Lecturer: Yan Wang
Department of Computing
Macquarie University
Office: E6A, room: 339
Email: yanwang@ics.mq.edu.au
Consultation time: Friday, 3pm-5pm

Staff

Dr Yan Wang
E6A339
9850 9539
Unit Convenor, Lecturer
yanwang@ics.mq.edu.au
Consultation Times: Friday, 3pm-5pm, E6A, room: 339

Administrator

Lisa Chanell
E6A 314
9850 9314
comp155-admin@ics.mq.edu.au

Note that Lisa Chanell is available on Tuesdays and Thursdays in E6A314 to help with student problems.

Material relating to the course will be posted on the Web at:
http://online.mq.edu.au/pub/COMP155/

- Study guide
- Summaries of lectures
- Tutorials
- Practicals
- Assignments
- What’s new
- FAQ (Frequently Asked Questions)
Timetable

3 hours of lectures, 1 hour tutorial and 1 hour practical each week

Lectures  Mon 4 pm, Tue 2 pm, Fri 2pm

Practicals  commence in Week 1

Tutorials  commence in Week 2

Check the time-table maker:

Textbook


It will also be used in COMP125


Assessment

There are 3 components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practicals</td>
<td>10%</td>
</tr>
<tr>
<td>Assignments</td>
<td>20%</td>
</tr>
<tr>
<td>Final Examination</td>
<td>70%</td>
</tr>
</tbody>
</table>

In order to pass the unit (i.e. obtain a grade of PC or better) you are required to

• submit satisfactory attempts to at least 8 of 12 practicals
• submit satisfactory attempts to at least 2 of the 3 assignments
• perform satisfactorily in the final examination.

Workload

COMP155 is a 3-credit-point unit. It is therefore expected that a student will spend approximately 12 hours per week on this unit throughout the semester. Since there are five hours of timetabled classes, this means you can expect to spend around seven hours working on COMP155 outside of class.

You should note that it is extremely unlikely that the practical and assignment work can be completed solely within your scheduled practical class time – you are expected to complete this work at other times.
Assignments

There are 3 assignments. Students must submit satisfactory attempts to at least 3 of them.

<table>
<thead>
<tr>
<th>Assignment Topic</th>
<th>Available Checkpoint</th>
<th>Due</th>
<th>Worth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 simple</td>
<td>Week 1</td>
<td>Week 3</td>
<td>Week 4</td>
</tr>
<tr>
<td>2 function</td>
<td>Week 4</td>
<td>Week 7</td>
<td>Week 8</td>
</tr>
<tr>
<td>3 arrays</td>
<td>Week 8</td>
<td>Week 11</td>
<td>Week 12</td>
</tr>
</tbody>
</table>

The assignments are a significant amount of work. Checkpoint will be provided about one week before the due date.

The assessment of assignments relies on 2 important components:

• **Automarking** (Your program will be run with some input and the obtained output will be compared to the correct expected one).

• **Handmarking** (Prac tutors will assess the correctness and the quality of your code).

Submission of Assignments

Assignments are to be submitted electronically via the submit link on the unit web page. You should submit your work early and often.

To submit your assignment:

[Submit link](http://submit.ics.mq.edu.au/)

Upload your program.

There is a late penalty of 20% for one day late, 60% for two days late, 100% for three days late.

Note that crowded laboratories, equipment failure or loss of your files are not valid reasons for late submission of an assignment.

Submission of Assignments

You are encouraged to:

1. Set your personal deadline earlier than the actual one
2. Keep backups of all your important files
3. Ensure someone else does not pick up your printouts

You must not send your assignments to lecturers by email in case the submit system crashes.

Plagiarism

The Department is particularly keen to ensure students are aware of its zero tolerance attitude to plagiarism.

Link to the Department policy:

[Plagiarism](http://www.student.mq.edu.au/plagiarism/)
Final Examination

A 3-hour written examination will be held at the end of semester. It will consist of 3 sections:

Section A will consist of multiple choice questions
Section B will consist of simple programming questions
Section C will consist of programming questions

Note that performing well in Section A only will not be sufficient to pass the unit.

An adequate performance in each of Sections A, B and C will be required to pass the unit.
COMP155 and COMP115

- The same textbook, assignments and exam
- Different students
- More (difficult) examples and tut, prac questions in COMP155

COMP155 and COMP115

- If you are taking a Computing degree (e.g. BIT, BIS, BCompSci) but:
  - your UAI is above 84.33, or
  - you are enrolled in BCompSci.

  You should enroll in COMP155 (Introduction to Computer Science – advanced). Change your enrolment as soon as possible (and no later than the 4th March) by filling a change of program form (deleting COMP115 and adding COMP155).

COMP155 and COMP115

- If your UAI is close to 84.33, talk to the convener of Comp155 (Dr. Yan Wang; E6A339) – you may qualify to join Comp155.

Features of COMP115/155

- Sometimes it is trivial (week 1)
- a lot of rules
- It is a highly practical course
- 5-15 minutes per lecture or tutorial for doing exercise
My suggestion

- Touch it!
- Learn by doing!

Outline

- Basic knowledge on computer
- Programming language
- Problem solving process
- Basic components of a C++ program
- Simple data types
- Arithmetic expressions
- Input and output
- Your first C++ program

The 1st Computer in the World

- The ENIAC was finally assembled in operation at the University of Pennsylvania during the fall of 1945.
  (Electronic Numerical Integrator and Calculator) using vacuum tubes
- Low speed
- Small storage space
- Huge volume
- But a big step
History of Computers

- In the 1950s, the second generation – Transistors
  Computers are very large devices only accessible to very few people
  Simple operations as word processing, calculation, accounting were done with computers

- Late 1950s and early 1960s - Integrated Circuit (IC)
  Multimillion-dollar computers emerged accessible to only very large companies

- 1970s, Microprocessor, Semiconductor memory
  Computers become cheaper and smaller

- In the 1980s, with LSI and SLSI, the first mini then microcomputers appeared

History of Computers: Moore's Law
(http://www.intel.com/research/silicon/mooreslaw.htm)

Moore's Law (formulated in 1965) (Gordon Moore, co-founder of Intel)
- chip "size" (# transistors) doubles every 18 months
- 2,500 transistors Intel 8008 (1972)
- 275,000 Intel 80386 (1985)
- 3.1 million Intel Pentium (1992)
- 42 million Intel Pentium IV (2000)

Elements of Computer System

Categories of Computers
- Mainframe computers
- Midsize computers
- Micro computers (personal computers)
- PDA, Embedded system
Elements of Computer System

- Hardware
- Software

Elements of Computer System

Hardware
- CPU (Central Processing Unit)
- MM (Main Memory)
  - RAM (Random Access Memory)
- Input devices, keyboard, mouse, scanner, secondary storage
  - Secondary Storage, hard disk, tape, floppy disk, CD-ROM, USB-disk
- Output devices, monitor, printer, secondary storage

Software

- MM is an ordered sequence of memory cells
- Each cell is used to store information
- Each cell has a unique address

Figure 1-1 Hardware components of a computer

Figure 1-2 Main memory with 100 storage cells
Elements of Computer System

Software: Software is the program written to perform specific tasks
- System programs, operating system (OS), Windows, Unix, Linux, DOS
- Application programs
  - Provided by vendors
  - User-developed

Computer Science
- It is a young discipline (1947)
- Study of the mathematics and technology of computers and their applications
- It is the art of programming a computer to design efficient and elegant methods of getting a computer to solve problems

Other domains
- Data Base
- Artificial Intelligence
- Scientific Computing
- Architecture
- Software Engineering
- Operating Systems
- Graphics
- Many other sub-disciplines such as security, communication and natural language processing
Language of a Computer: Machine Language

The language of a computer is a sequence of 0s and 1s.
It is called machine language.
The digit 0 or 1 is called a binary digit (bit).
A sequence of 0s and 1s is also referred to as binary code.
A sequence of 8 bits is called a byte.
1 kilobytes (KB) = \(2^{10}\) bytes = 1024 bytes.
1 megabytes (MB) = \(2^{20}\) bytes = 1048576 bytes.
1 gigabytes (GB) = \(2^{30}\) bytes = 1073741824 bytes.

Early computers were programmed in machine language.
It is a sequence of 0s and 1s (binary code).
To calculate wages = rates * hours in machine language, the following sequence of instructions might be needed:

```
100100 0000 010001 // load
100110 0000 010010 // multiplication
100010 0000 010011 // store
```

(if 100100 stands for load, 100110 stands for multiplication and 100010 stands for store)

Language of a Computer: Assembly Language

An instruction in assembly language is an easy-to-remember form.
An assembler is a program that translates a program written in assembly language into an equivalent program in machine language.
Using the assembly language instructions, the equation wages = rates * hours can be written as follows:

```
LOAD rate
MULT hours
STOR wages
```

Language of a Computer: High-Level Languages

High-level languages include BASIC, FORTRAN, COBOL, PASCAL, C++, C, JAVA.
A compiler is a program that translates a program written in a high-level language to an equivalent machine language.
The equation wages = rate * hours can be written in C++ as wages = rate * hours.
Problem Solving Process

- **Step 1 - Analyze the problem**
  - Outline the problem and its solution requirements
  - Design steps (algorithm) to solve the problem

- **Step 2 - Implement the algorithm**
  - Implement the algorithm in a programming language
  - Verify that the algorithm works

- **Step 3 - Maintenance**
  - Maintenance requires using and modifying the program if the problem domain changes

Structured Design

- **Structured design** - dividing a problem into smaller subproblems
- Also known as *top-down design, stepwise refinement, and modular programming*
- The problem is divided into smaller problems
  - each subproblem is then analyzed
  - a solution is obtained to solve the subproblem
  - The solutions of all subproblems are then combined to solve the overall problem
- The process of implementing a structured design is called *structured programming*
Object-Oriented Programming

- Identify components called **objects**
- Specify relevant **data** and possible **operations** to be performed on that data
- Each **object** consists of data and operations on that data
- An object combines data and operations on the data into a single unit
- A programming language that implements OOD is called an object-oriented programming (OOP) language

**Standard C++ and ANSI/ISO C++**

- ANSI/ISO C++ was established in 1998.
- Most today's compilers comply with the standard.
- The syntax sets of different compilers may have extensions.

**Programming—some terms**

- **Computer program** - a sequence of statements designed to accomplish some task
- **Programming** - a process of planning and creating a program
- **Programming language** - a set of rules, symbols, and special words
- **Syntax** - rules that tell us which statements (instructions) are legal
- **Semantic** rules - determine the meaning of the instructions

**C++ Programs**

- A **C++ program** is a collection of one or more subprograms, called functions
- A **subprogram** or a **function** is a collection of statements that when activated (executed) accomplishes something
- Every C++ program has a function called **main**
C++ Programs

#include <iostream>
#include <cstdlib>
...
using namespace std;
int main ()
{
    statement1;
    ...
    ...
    statementn;
    return 0;
}

C++ Programs

The smallest individual unit of a program written in any language is called a token
1. special symbols
   - + - * / . ; ? , <= = >
2. word symbols/reserved words/keywords
   - return float int class
   - void true for while

C++ Programs

The smallest individual unit of a program written in any language is called a token
3. identifier
   - consists of letters, digits and underscore _
   - begins with a letter or a underscore
   - C++ is case sensitive—uppercase and lower case letters are different
   - Some of the predefined identifiers are cout and cin
   - Unlike reserved words, predefined identifiers may be redefined, but it would not be wise to do so

C++ Programs

legal and illegal identifiers

1. legal identifiers
   - name Name _name
   - score_of_COMP155 scoreofcomp155

2. illegal identifiers
   - 1st_name 2nd_name one+two
   - employee salary
**C++ Programs**

**Data Type:** a set of values together with a set of operations

1. Simple data type
2. Structured data type
3. Pointers

**Integral Data Type**
- char 1 byte, 2^8 values -128~127
- short 2 bytes -32768~32767
- int 4 bytes, 2^32 values -2147483648~2147483647
- long 4 bytes, 2^32 values -2147483648~2147483647
- bool (boolean) 1 byte, true, false
- unsigned char 1 byte, 0-255
- unsigned short 2 bytes 0~65535
- unsigned int 4 bytes, 2^32 values 0~4294967295
- unsigned long 4 bytes, 2^32 values 0~4294967295

**Floating-Point Data Type**
- deals with decimal numbers

**Enumeration Data Type**
- user-defined data type

**Appendix F**
- Dev-C++.lnk
C++ Programs

int Data Type
- Examples:
  - -6728
  - 0
  - 78
- Positive integers do NOT have to have a + sign in front of them
- No commas are used within an integer
- In C++ commas are reserved for separating items in a list

C++ Programs

bool Data Type
- The data type bool has two values, true and false
- The central purpose of this data type is to manipulate logical (Boolean) expressions
- True and false are called logical (Boolean) values
- In C++ bool, true, and false are reserved words

C++ Programs

Floating-Point Data Types
1. float - single precision – 4 bytes, any real number between -3.4E+38 ~ 3.4E+38 6 decimal places
2. double - double precision – 8 bytes, any real number between -1.7E+308 and 1.7E+308 15 decimal places
3. long double - On most newer compilers, the data types double and long double are the same

C++ Programs

To represent real numbers C++ uses scientific notation called floating-point notation

Table 2-3 - Examples of Real Numbers Printed in C++ Floating-Point Notation

<table>
<thead>
<tr>
<th>Real Number</th>
<th>C++ Floating Point-Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>75.924</td>
<td>7.592400E1</td>
</tr>
<tr>
<td>0.18</td>
<td>1.800000E-1</td>
</tr>
<tr>
<td>0.0000453</td>
<td>4.530000E-5</td>
</tr>
<tr>
<td>-1.482</td>
<td>-1.482000E0</td>
</tr>
<tr>
<td>7800.0</td>
<td>7.8000000E3</td>
</tr>
</tbody>
</table>
C++ Programs

String
- Programmer-defined type supplied in the standard library
- Sequence of zero or more characters ""
- Enclosed in double quotation marks "abcd"
- A string with no characters is called a null or empty string
- Every character has a relative position in the string
- The position of the first character is 0, the position of the second character is 1, and so on
- Length of a string is the number of characters in it

"abcd" length=4
a b c d

C++ Programs: Arithmetic Operators

+ addition
  2+3 is 5, 1.2+2.3 is 3.5
- subtraction
  5-3 is 2, 3-1.5 is 1.5
* multiplication
  2*3 is 6, 2*1.5 is 3.0
/ division
  6/3 is 2, 7/3 is 2 (quotient) 7=2*3+1
% remainder (modulo operator)
  6%3 is 0, 7%3 is 1 (remainder)

Unary operator - has only one operand
-5 +5
x++
x--

Binary Operator - has two operands
2+3 2*3

C++ Programs: Arithmetic Operators

Order of Precedence

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>()</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>* /</td>
<td>%</td>
</tr>
<tr>
<td>3</td>
<td>+ -</td>
<td></td>
</tr>
</tbody>
</table>

When operators are all on the same level, they are performed from left to right.
C++ Programs: Arithmetic Operators

- An Example

\[(3*7)-6*((2*5)/4)+6\]

=21-6*(10/4)+6

=21-6*2+6

=21-12+6

=9+6

=15

C++ Programs: Mixed Expressions

- Mixed expression - an expression that has operands of different data types
- Contains both integers and floating-point numbers
- Examples of mixed expressions:
  - 2 + 3.5
  - 6 / 4 + 3.9
  - 5.4 * 2 - 13.6 + 18 / 2

C++ Programs: Evaluating Mixed Expressions: implicit type conversion

- A) If the operator has the same types of operands, it is evaluated according to the type of the operands
  - 2+3 is 5    2.5+3.5 is 6.0 (or 6. not 6)    3/2 is 1   not 1.5
- B) If the operator has different types of operands,  2+2.5 =2.0+2.5= 4.5   3.0/2 is 1.5
  - during calculation the integer is changed to a floating-point number with the decimal part of zero and the operator is evaluated
  - the result is a floating-point number

C++ Programs: Evaluating Mixed Expressions: type casting/ explicit type conversion

- static_cast<int>(7.9)   // 7
- static_cast<int>(7.3)   // 7
- static_cast<double>(7)   // 7.0
- static_cast<double>(15) /2  // 7.5
- static_cast<double>(15 /2)  // 7.0
- static_cast<char>(65)  //'A'
- static_cast<int>('A')  // 65
C++ Programs: Assignment Statement

- **variable**: a memory location whose content may change during program execution
- **constant**: a memory location whose content is not allowed to change during the execution of the program
- The assignment statement takes the form:
  \[
  \text{variable} = \text{expression};
  \]

In C++ = is called the assignment operator.

C++ statement like:

```cpp
int i;
i=0;
i = i + 2; // Evaluates whatever is in i, adds two to it, and assigns the new value to the memory location i
```

3 steps of an assignment

1. evaluate the (right) expression
2. type conversion
3. assign the value

Examples:

```cpp
int i;
float x;
x=1+2; // x is 3.0
x=x+1; // x is 4.0
i=7.9; // i is 7
```

Variables can be initialized when they are declared.

For example:

```cpp
int first=13, second=10;
char ch='a';
double x=12.6, y=123.456;
```
C++ Programs: Assignment Statement

- In C++, `const` is a reserved word.
- The syntax to declare a named constant is:
  ```cpp
  const dataType identifier = value;
  ```
  e.g.,
  ```cpp
  const double PI = 3.14;
  ```

C++ Programs: Input (Read) Statement

- Syntax of `cin` together with `>>` is used to gather input:
  ```cpp
  cin>>variable>>variable. . .;
  ```
- The extraction operator is `>>`
- For example,
  ```cpp
  int age;
  cin>>age;
  ```
- It causes the computer to get a value of the type double and place it in the memory cell.

C++ Programs: Output Statement

- The syntax of `cout` together with `<<` is:
  ```cpp
  cout<<expression or manipulator<<expression or manipulator...;
  ```
- This is called an output statement.
- The `<<` operator is called the insertion operator or the stream insertion operator.
- Sometimes this is also called a cout statement.
C++ Programs: Output Statement

- Expression - is evaluated and its value is printed at the current cursor position on the screen.
- Manipulator - manipulates the output.
- `endl` - the simplest manipulator, which causes the cursor to move to the beginning of the next line.

The output of the C++ statement `cout<<a;` is meaningful provided the variable `a` has been given a value.

For example, the sequence of C++ statements:

```cpp
int a = 45;
cout<<a;
```

will produce an output of 45.

Example 1:

```cpp
int a, b;
cin>>a; //Line 1
cin>>b;  //Line 2
cout<<"a= "<<a<<endl;  //Line 3
cout<<"b= "<<b<<endl;  //Line 4
cout<<"a+b= "<<a+b<<endl;  //Line 5
```

Example 2:

```cpp
int a, b;
a = 65 ; //Line 1
b = 78 ;  //Line 2
cout<<29/4<<endl;  //Line 3 output: 7
cout<<3.0/2<<endl;  //Line 4 output: 1.5
cout<<"Hello there.\n";  //Line 5 output: Hello there
cout<<7<<endl;  //Line 6 output: 7
cout<<3+5<<endl;  //Line 7 output: 8
cout<<"3+5";  //Line 8 output: 3+5
cout<<endl;  //Line 9
```
C++ Programs: The New Line Character

- The new line character is `\n`
- Without this character the output is printed on one line
- This command tells the output to go to the next line
- When `\n` is encountered in the string, the cursor is positioned at the beginning of the next line
- A `\n` may appear anywhere in the string

C++ Programs: Common Escape Sequences

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\n</code></td>
<td>Newline</td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Tab</td>
</tr>
<tr>
<td><code>\b</code></td>
<td>Backspace</td>
</tr>
<tr>
<td><code>\r</code></td>
<td>Return</td>
</tr>
<tr>
<td><code>\\</code></td>
<td>Backslash</td>
</tr>
<tr>
<td><code>\'</code></td>
<td>Single quotation</td>
</tr>
<tr>
<td><code>\&quot;</code></td>
<td>Double quotation</td>
</tr>
</tbody>
</table>

C++ Programs: Preprocessor Directives

- Only a small number of operations are explicitly defined in C++
- Many of the functions and symbols that are necessary to run a C++ program are provided as a 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 `#`
- There is no semicolon at the end of these commands

C++ Programs: Preprocessor Directives

- The general syntax to include a header file (provided by the SDK) in a C++ program is:
  ```
  #include <headerFileName>
  ```
- The preprocessor directive causes the preprocessor to include the header file iostream in the program
- The syntax is:
  ```
  #include <iostream>
  ```
C++ Programs: Preprocessor Directives

- In Standard C++, header files have the file extension .h
- The descriptions of the functions needed to perform I/O are contained in the header file iostream.h
- The syntax is:
  - `#include <iostream.h>`
  - `#include <math.h>`

C++ Programs: Increment & Decrement Operators

- Increment operator - increment the value of a variable by 1
- Decrement operator - decrement the value of a variable by 1
- Pre-increment: `++variable`
- Post-increment: `variable++`
- Pre-decrement: `--variable`
- Post-decrement: `variable--`

If you have `x = 5; y = ++x;`
- `x=x+1;`
- `y=x;`
- after the second statement both x and y are 6.

If you have `x = 5; y = x++;`
- `y=x;`
- `x=x+1;`
- after the second statement y is 5 and x is 6.
int x, y;
x=5;
x++;  // x is 6
y=x;
x=5;
++x;  // x is 6
y=x;
x=5;
y=++x;  // y is 6
x=5;
y=x++;  // y is 5