COMP2014

Object-Oriented Programming

Dr Alex Dong
School of Computer, Data and Mathematical Sciences
Western Sydney University

Part A

Unit Overview

General Information

- COMP2014 Object-Oriented Programming
  - Level 2 (1st year 2nd semester)
  - 10 Credit Points
- Prerequisites
  - COMP1005 Programming Fundamentals or equivalent
- Assumed Knowledge
  - Basic knowledge in programming fundamentals, including flow of control, one-dimensional array, functions, I/O operations
  - Able to write simple programs in Java, C or C++
- Contact details

Dr Alex Dong

Zoom: 607 668 1320 (passcode **COMP2014** if required)

Email: A.Dong@westernsydney.edu.au

Teaching team

- Unit Coordinator and lecturer: Dr **Alex Dong**
  - Lectures: Tuesday 2pm – 4pm
  - Zoom id: 897 4041 6917
  - Email: A.Dong@westernsydney.edu.au
- Tutor 1: Dr **Alex Dong**
  - Practical classes: all classes in Parramatta South
  - Zoom id: 607 668 1320
  - Email: A.Dong@westernsydney.edu.au
- Tutor 2: Mr **Vishal Patel**
  - Practical classes: all classes in Kingswood
  - Zoom id: 802 327 2508
  - Email: V.Patel2@westernsydney.edu.au

Brief summary of the unit

This unit presents the concepts and principles of programming with the emphasis on object-oriented paradigm. It addresses the importance of the separation of behavior and implementation, as well as effective use of *abstraction*, *data hiding*, *encapsulation*, *inheritance* and *polymorphism*.

The teaching language of this unit will be C++ but Java is allowed for assignment 2.

Presentation

- 4 Hours per week:
  - Two-hour lecture per week
  - Two-hour practical per week from Week 2
- At least **6 hours** self-study per week
- Lecture sessions: scheduled for **13 weeks** lectures
- Practical: scheduled for **12 practical sessions** (attendance is compulsory)
- Feedback giving and taking
  - Two online questionnaires (1 bonus mark each)
  - Face-to-face checking for practical and assignments
- PASS: Peer Assistant Study Sessions
  - Voluntary group work for students to help each other building up programming skills
  - Lead by Alex Dong

Unit Schedule

Week	Topics	Readings	Tutorial and Practical
1	Unit Introduction C++ Basics	UO, LG & Ch1-2	Online survey 1
2	Function calls & parameter passing	Ch3-4	Practical 1
3	Arrays (multi-dimensional arrays)	Ch5	Practical 2
4	Classes & data abstraction	Ch6	Practical 3
5	Class design & implementation	Ch7	Practical 4 Online Tutorial 1
6	Pointers & dynamic arrays	Ch10	Practical 5
7	Strings, Streams, Static Variables & file I/O	Ch7,9&12	Assignment 1 due

Unit Schedule

Week	Topics	Readings	Tutorials and Practicals
8	Middle break		
9	Inheritance	Ch14	Assignment 1 marking Online Tutorial 2
10	Polymorphism & virtual functions	Ch15	Practical 6
11	Operator overloading & references	Ch8	Practical 7
12	Class template & linked data structures	Ch16&17	Practical 7 (ctn) Assignment 2 due
13	Standard template library	Ch19	Online Tutorial 3
14	Unit review		Online survey 2

Mandatory components

Mandatory components:

- Attending all 12 practical sessions (face-to-face in campus or via zoom)
  - 10 sessions with attendance checking
  - 2 sessions for assignment demonstration without attendance checking
- Attempting all 3 online tutorials
  - Online quizzes are checked by vUWS system automatically
- Attempting and demonstrating 2 assignments
- Attending the final exam:

Note that all practical tasks and assignments will be individually checked during the practical sessions. No marks will be given based on email or vUWS submissions. If your tutor does not get time to check your work at the class of the due date, you need to contract the tutor via email on that date to arrange another time.

Assessments

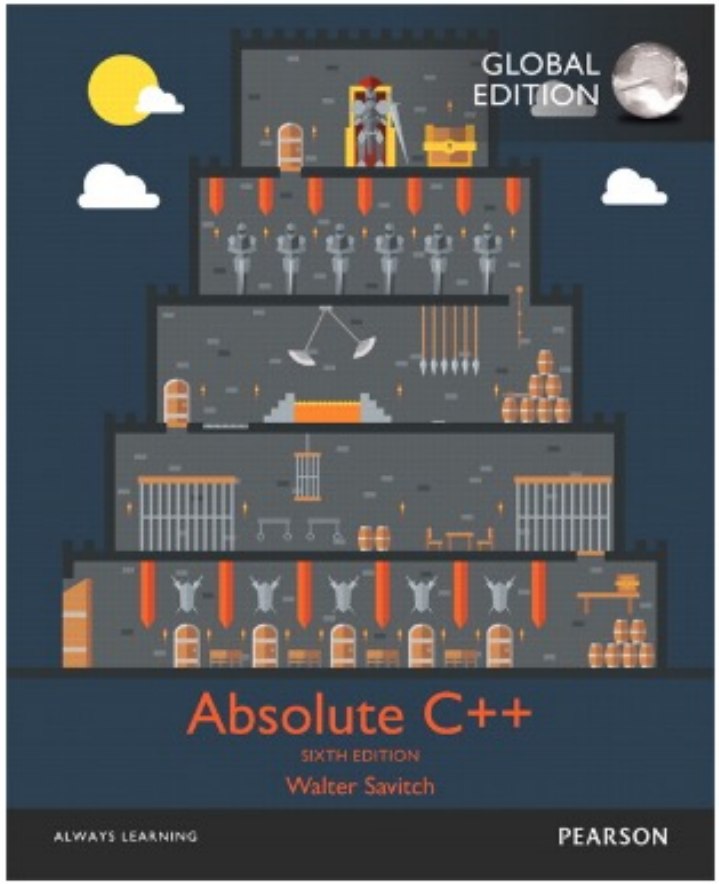
Four assessment components:

- Assignment 1 (15%): due at 5pm Friday 9 September 2022
- Assignment 2 (15%): due at 5pm Friday 28 October 2022
- Practical and tutorials (20%) : FNS if attendance is less than seven exclusive two assignment checking sessions.
  - Online tutorial performance (40-60 true/false, multiple choice or very confusing multiple answer questions in three sessions): 6%
  - Practical performance: 14%
- Final exam (50%): two-hour, open-book
- Online survey participation: 2 bonus marks (no bonus marks if your continuous assessment marks have reached 50)

Assessments

- To receive a pass grade, you must achieve 50% of overall marks and complete all mandatory components.
- Assignment submissions and demonstration
  - Code: You are required to submit your code and declaration to vUWS by the deadlines.
  - Demonstration: Your demonstration will be at your practical session of the week immediately after the submission deadlines.

Textbook: new version



Absolute C++, Global Edition (6e) : 9781292098593

▶ I'd like to request an inspection copy

Absolute C++, Global Edition (6e)

Walter Savitch
University of California, San Diego
...more...

Edition	6th
ISBN	9781292098593
ISBN 10	1292098597
Published	25/05/2016
Published by	Pearson Higher Ed USA
Pages	1008

Available once published

Book + Access Code

$117.95

Add to Cart

Add to Shortlist

Questions

Part B

C++ Basics
for non-C++ Programmers

Topics covered by this part

- Structure of C++ programs
- Input and output
- Flow of control
  - Selection
  - Repetition
- Random numbers

Compare to Java code

```java
import java.util.Scanner;

public class Hello {

   public static void main(String[] args)
   {
      String name;
      System.out.println("Enter your name:”);
      Scanner input = new Scanner(System.in);
      name = input.nextLine();
      System.out.println("Hello " + name+”!”);
   }
}
```

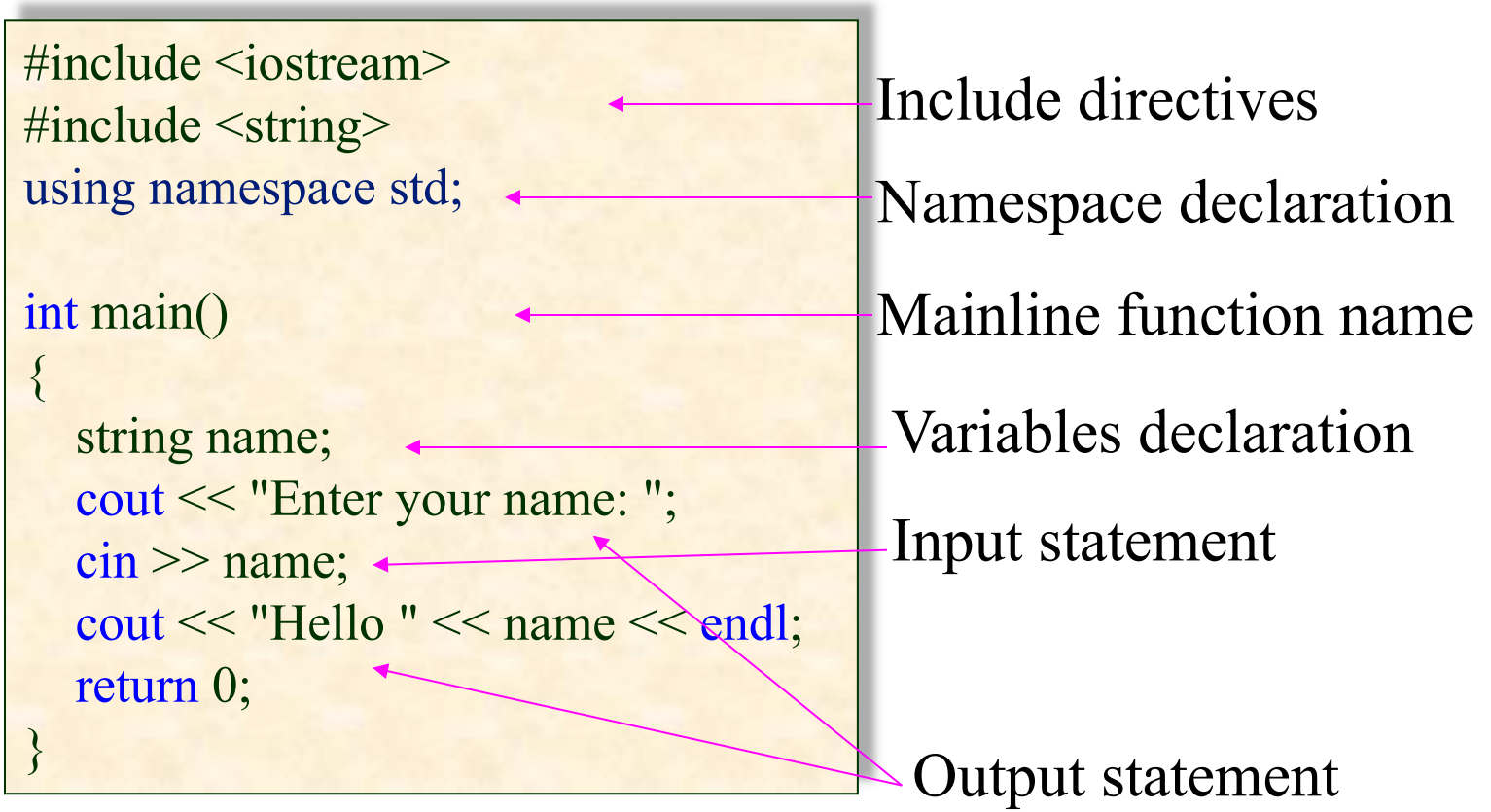
```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
   string name;
   cout << "Enter your name: ";
   cin >> name;
   cout << "Hello " << name << endl;
   return 0;
}
```

Download code: Hello.java

Download code: Hello.cpp

A Simple C++ Program

Hello.cpp



The Structure of a C++ Program

- A C++ program consists of:
  - Pre-processor directives (introduced by #, similar to import in Java)
  - Declarations (class, function, variable etc.)
  - Definitions/implementation (class members, functions, variables)
- Different from Java, a C++ program allows free functions (not a method of any class), including the main() , function
- There must be a file with extension .cpp which contains the main() function as the executable point.

Basic C++ Keywords

bool short int long float double char void	if else switch case default	while do continue break for	main private protected public const static return	true false

String is not a built-in data type in C++

bool is called “boolean” in Java

Input and Output Statements

```
/* This program shows how to input and output
data. */

#include <iostream>
#include <string>

using namespace std;

int main() {

   string name, id;
   int age;

   cout << "Input your name:\n";
   cin >> name;

   cout << "Input your student ID:\n";
   cin >> id;

   cout << "Input your age:\n" ;
   cin >> age;

   cout << "\nMy name is " << name <<". " << endl;
   cout << "My student ID is " << id <<"." << endl;
   cout << "I am " << age << " year old." <<endl;

   return 0;
}
```

comments

Include string handling library

Using standard namespace

Arithmetic Expressions

volume.cpp

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
   const double Pi = 3.14159;
   double radius, volume;

   cout << "Please enter the radius of a sphere: ";
   cin >> radius;

   volume = 4.0/3.0 * Pi *pow(radius, 3.0);
   cout <<  "The volume of the sphere is: \n"
        << volume;
  return 0;
}
```

Include `math` library, which support the basic math fuctions

A constant which value can't change in the program.

Arithmetic expression

Selection Structures (if…else)

```
#include<iostream>
using namespace std;
int main() {
    int num;

    cout << "Input an integer";
    cin >> num;
    if (num > 0)
        cout << "Your input is a positive number.";
        cout << "which is " << num;
    else
        cout << "Your input is a non-positive number.";
        cout << "which is " << num;
    return 0;
}
```

Error message:
Misplaced else

Selection Structures (if…else)

```
#include<iostream>
using namespace std;
int main() {
    int num;

    cout << "Input an integer";
    cin >> num;
    if (num > 0)
        {
          cout << "Your input is a positive number.";
          cout << "which is " << num;
        }
    else
        {
          cout << "Your input is a non-positive number.";
          cout << "which is " << num;
        }
    return 0;
}
```

Operators

Relational Operators

<	less than
<=	less than or equal
==	equal
>	greater than
>=	greater than or equal
!=	not equal

Examples:

```
if    (hour < 0 || hour > 24)
    cout << "Incorrect Input";
```

```
if    (hour >= 0 && hour <= 24)
    cout << "Correct Input";
```

Boolean Operators	Meaning
&&	AND
||	OR
!	NOT

Repetition Structures

Form 1

```
while (condition)
{
    statement list;
}
```

Form 2

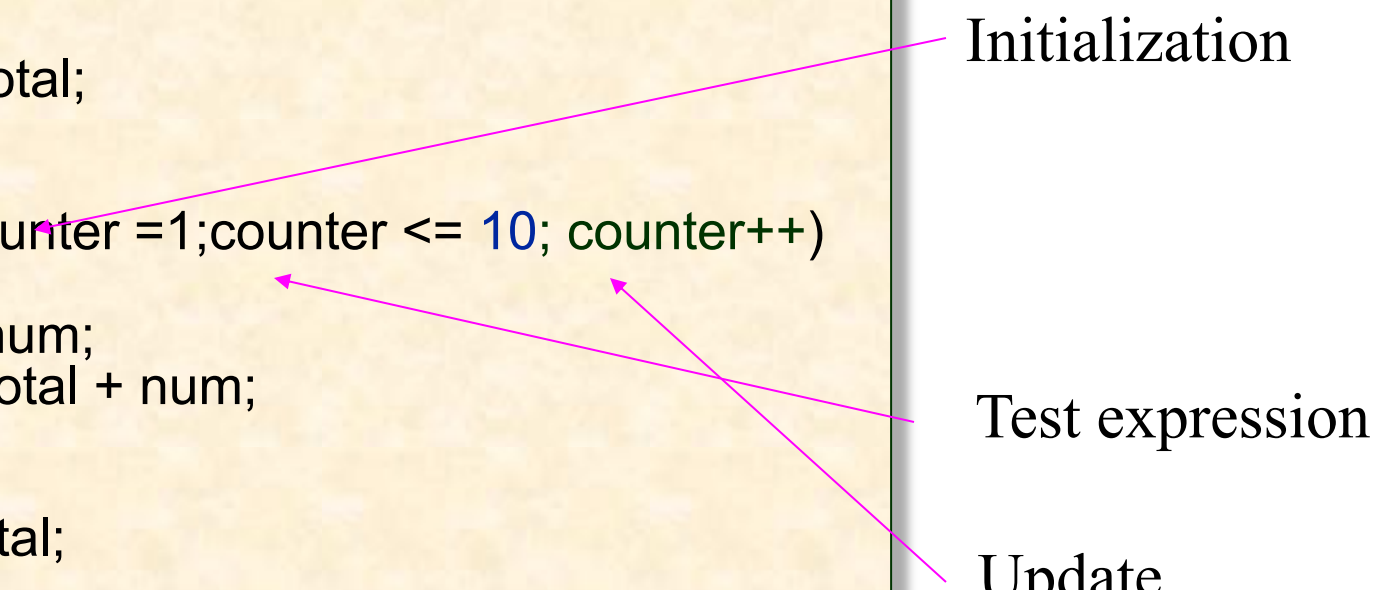
```
do
{
    statement list;
} while (condition)
```

Form 3

```
for (initialization; test expression; update)
{
    statement list;
}
```

for loop

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

int main() {

    int num, total;
    total = 0;

    for (int counter =1;counter <= 10; counter++)
    {
        cin >> num;
        total = total + num;
    }

    cout << total;
   return 0;
}
```



Random Number Generator

- Random numbers are widely used in simulations, games, special algorithms and so on.
- Call function rand() to get a random number
  - The function takes no argument
  - It returns an integer between 0 & RAND_MAX
  - RAND_MAX is a system constant, value relies on the C++ compiler you use.
- Scaling
  - Squeezes random number into smaller range, e.g.

    rand() % 6: *returns random value between 0 and 5*
  - % is modulus operator (remainder)
- Shifting
  - Shifts range between 1 & 6 (e.g., die roll)

    rand() % 6 + 1: *returns random value between 1 and 6*

Random Number Seed

- Pseudorandom numbers
  - Calls to *rand()* producing a "sequence" of pre-produced numbers. Different time you call generates the same sequence of numbers.
- Use function *srand()* to alter the starting point of the sequence

  srand(seed_value);
  - The "*seed value*" sets the start point of the random sequence, normally use srand(time(0)) to avoid the same sequence of numbers each time you run your program
  - Note that time() returns system time as numeric value. You might need to include the C standard library, which contains time() function if it is not default

    #include <cstdlib>

Random Examples

- Random int between 1 & 100:
  rand() % 100 + 1
- Random int between 10 & 20:
  rand() % 11 + 10
- Random double between 0.0 & 1.0:
  (RAND_MAX – rand())/(double)(RAND_MAX)
  - Type casting (double) is used to force double-precision division

IDE for C++

- You may use any IDE that is workable at your computer
- Typical IDEs:
  - Visual Studio Community: for Windows, come with compiler
  - xCode: for Mac OS, come with compiler
  - Visual Code: for both Windows and Mac OS, you need to install a compiler firstly
- compiler can be anything workable with your IDE, typically gcc or g++

Homework

- Read textbook: Chapters 1&2
- Complete the online questionnaire in vUWS. It will be turned off at the end of the week.
- Practical sessions start next week. Complete the practical tasks for week 2 at home and connect to your tutor's zoom at the time of your scheduled class.
- Warm up textbook Chapters 3-4.
- If you have not registered to a practical session, please do it as soon as possible. Contact your campus tutor or me for any questions.