COMP2004
Programming Practice
2002 Summer School

Kevin Pulo
School of Information Technologies
University of Sydney

About Me

- Kevin Pulo
- Room: Madsen G88
- Email: kev@cs.usyd.edu.au

- Consultation:
  - 30 mins before each lecture
  - Email to arrange an appointment during consultation times

Textbook

- Accelerated C++ by Koenig & Moo

- The C++ Programming Language by Stroustrup (3rd edition)
- C++ Primer by Lippman (3rd edition)
- Thinking in C++ by Eckel (2nd Edition)

- http://www.accu.org/

Course Information

- Programming Practice
  - so it will involve programming
- Using Development Tools
- Using Software Libraries

- http://www.cs.usyd.edu.au/~kev/pp/
- Friday's lecture: 10am in Carslaw 175
- Tutorials in Services bldg and LG45

Assessment

Assignment 0	0%
Assignment 1	10%
Assignment 2	10%
Assignment 3	20%
Final Exam	60%

- Need 45% in the exam and assignment components to pass

Assignment Policy

- Machine and hand marked
  - Follow output instructions exactly
  - Must work on department software/hardware
- Individual work - no groups or copying
- No late assignments without valid paperwork

The C++ Language

- The language we'll be using
- This is not a C course
- The syntax is similar to Java
- A textbook or reference is essential

C++ At Basser

- GNU C++ compiler

```
    g++  -Wall  -g  -o  hello  hello.cc
```

- hello.cc contains the C++ code
- Executable will be named hello
- -Wall turns on all warnings
- -g adds debugging information

A Simple C++ Program

- C++ programs start in a function called main

```
#include  <iostream>
int  main()  {
   std::cout << "Very simple" << std::endl;
}
```

- That's a complete C++ program

Variables and Constants

- C++ is strongly typed
- Basic types like Java (int, double, etc)
- Prefix with const for constants

Variables and Constants

- For example

```
#include  <iostream>
int  main()  {
   const  int  value  =  5;
   int  result  =  4;
   result  +=  value;
   std::cout  <<  result  <<  std::endl;
}
```

Enumerated Types

- Types that can only have certain named values
- Very useful for restricted domains
- Each name is an integer internally

```
enum  day_of_week  {Sun,  Mon,  Tue,
                    Wed,  Thu,  Fri,  Sat};
day_of_week  today  =  Wed;
```

Functions

- Like Java methods except
  - They are not part of a class
  - No object is used to call them

```
double  halve (int  number)  {
   double  result;
   result  =  number  /  2.0;
   return  result;
}
```

Control Flow

- if, switch, do, while, for - similar to Java
- Some types convertable to bool
- For numeric types:
  - Zero is false
  - Everything else is true

```
int  main()  {
   int  i = 5;
   if  (i)       // equiv to:   if  (i  !=  0)
      std::cout  <<  i  <<  " is true\n";
}
```

C++ Programming Style

- C++ is a flexible language
  - as a procedural language
  - as a modular language
  - as an object oriented language
  - as a generic language
- We'll progress through them all during the course

Basic Input

- std::cin is used to read input

```
#include  <iostream>
int  main()  {
  std::string  s;
  int  i;
  double  d;
  char  c;
  std::cin  >>  s  >>  i  >>  d  >>  c;
  std::getline(cin,  s);
}
```

Basic Output

- std::cout is used to print output

```
#include  <iostream>
int  main()  {
  std::string  s  =  "Hello";
  int  i  =  42;
  double  d  =  1.3;
  char  c  =  ' ';
  std::cout  <<  s  <<  c  <<  i  <<  c  <<  d;
}
```

Error Output

- std::cerr works like std::cout
- It is used for error messages
- std::cerr will be ignored for marking
- So debugging output should go to std::cerr

```
#include  <iostream>
int  main()  {
  std::cerr  <<  "Sent  to  std::cerr\n";
  std::cout  <<  "Sent  to  std::cout\n";
}
```

Streams

- cin, cout and cerr are just instances of streams
- Streams are used for files as well
- You can even use them for strings
- For now we'll stick to cin and cout...

More on Input

- You can test std::cin to see if input succeeded

```
#include <iostream>
int main() {
   int x, y;
   if (std::cin >> x >> y)
      std::cout << "worked" << std::endl;
   else
      std::cout << "failed" << std::endl;
}
```

How Input/Output can fail

- No input left - end of file
  - std::cin.eof() will return true
- Wrong data - eg. "abc" is not an int
  - std::cin.bad() will return true
- Hardware or system failure
  - std::cin.fail() will return true
- Call std::cin.clear() to recover
  - The stream can then be used
  - Throws an exception if it fails

Manipulators

- Manipulators are part of stream library
- #include <iomanip> to use them
- They perform operations on the stream
- But are used just like input/output items
- std::endl is an example
  - doesn't need #include <iomanip>
  - outputs '\n' to the stream
  - flushes the stream

setw and setfill

- setw sets minimum output width
- resets to 0 after next output operation
- setfill sets the fill characters
- Fill character defaults to space

```
std::cout << std::setw(10) << 123;
std::cout << std::setw(5) << std::setfill('#');
std::cout << "hi" << 123;
```

- Can also use std::cout.width(10)
- And std::cout.fill('#')

setprecision

- Number of digits after the decimal point
- Only applies to floating point output
- Setting to 0 sets to the default

```
double d = 2.0 / 3.0;
std::cout << d << '\n';
std::cout << setprecision(4) << d << '\n';
std::cout << setprecision(0) << d << '\n';
```

- Can also use std::cout.precision(4)

Buffered Output

- std::cout is buffered
  - output may be delayed until:
    - the buffer is full
    - std::flush or std::endl is output
- std::cerr is unbuffered
  - outputs immediately

flush

- Defined in #include <iostream>
- Flushes an unbuffered stream
- Used when you need the output to happen now

```
std::string name;
std::cout << "Enter name : " << std::flush;
std::cin >> name;
```

- Can also use std::cout.flush();

std::cerr for debug output

```
#include <iostream>
int main() {
  std::cout << "A";
  function_which_might_crash();
  std::cout << "B";
}

std::cout << "A" << std::flush;
std::cout << "A" << std::endl;
std::cerr << "A";
```