C++ Support Classes
Introduction

Dr. Johnson

School of Mathematics

2015
OUTLINE

1 THE BASICS
   • What is a computer?
   • Syntax

2 UNDERSTANDING DATA
   • Handling Data and Variables
   • Namespaces
   • Simple Input and Output

3 SUMMARY
OUTLINE

1 THE BASICS
   - What is a computer?
   - Syntax

2 UNDERSTANDING DATA
   - Handling Data and Variables
   - Namespaces
   - Simple Input and Output

3 SUMMARY
OUTLINE

1 THE BASICS
   • What is a computer?
   • Syntax

2 UNDERSTANDING DATA
   • Handling Data and Variables
   • Namespaces
   • Simple Input and Output

3 SUMMARY
Topics:
- Computers and Programs;
- Syntax and Structure of a Program;
- Data and Variables;

Aims:
- Understand the idea of programming a computer;
- Write a simple program to input and output data.
**AN IDEALIZED COMPUTER**

- **CPU - Central Processing Unit**
- **CPU and memory work together**
- **Input** may be from keyboard, mouse or a file
- **Output** may be to screen or a file
**An Idealized Computer**

- **CPU** - Central Processing Unit
- **CPU** and memory work together
- **Input** may be from keyboard, mouse or a file
- **Output** may be to screen or a file
What is a program?

A program is a sequence of instructions to enable a computer to complete a task.

- Computers can remember programs.
- Originally programs had to be written in machine code - low-level.
- Now we write programs in code - and let the computer write the machine code for us!
- C/C++ are best described as mid-level languages.
- Higher-level languages include Fortran, Matlab, and others.
**What is a program?**

A program is a sequence of instructions to enable a computer to complete a task.

- Computers can remember programs.
- Originally programs had to be written in machine code - low-level.
- Now we write programs in code - *and let the computer write the machine code for us!*
- C/C++ are best described as mid-level languages.
- Higher-level languages include Fortran, Matlab, and others.
WHAT IS A PROGRAM?

A program is sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - low-level
  - Now we write programs in code - and let the computer write the machine code for us!
- C/C++ are best described as mid-level languages
- Higher-level languages include Fortran, Matlab, and others.
**WHAT IS A PROGRAM?**

A program is a sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - low-level
- Now we write programs in code - *and let the computer write the machine code for us!*
- C/C++ are best described as mid-level languages
- Higher-level languages include Fortran, Matlab, and others.
What is a program?
A program is sequence of instructions to enable a computer to complete a task.

- Computers can remember programs
- Originally programs had to be written in machine code - *low-level*
- Now we write programs in code - *and let the computer write the machine code for us!*
- C/C++ are best described as mid-level languages
- Higher-level languages include Fortran, Matlab, and others.
The key elements of C/C++ syntax are:

- Semicolon used to mark end of statements
- Case is important
- Totally free form, lines and names can be as long as you like!
- Comments take the form /* C style comment */ or // C++ style comment
- Code blocks are surrounded by braces {}
STRUCTURING YOUR CODE

- Include libraries and methods
- Declare data types and variable names
- Input values for variables
- Required calculations are carried out in sequence
- Output to the screen or a file
Structuring Your Code

- Include libraries and methods
- Declare data types and variable names
- Input values for variables
- Required calculations are carried out in sequence
- Output to the screen or a file
A Very Simple C++ Code

- The following is a C++ program.

```cpp
main()
{
}
```

- There are no commands to execute, so the program will not do anything.
- Please see notes and web tutorials on how to compile and run your programs!
The following is a C++ program.

```c++
main()
{
}
```

- There are no commands to execute, so the program will not do anything.
- Please see notes and web tutorials on how to compile and run your programs!
A C++ code is saved as an ASCII text file and can written in any editor.

However, in this course we will encourage the use of an Integrated Development Environment (IDE).

The two recommended pieces of software are:
- Visual Studio (Windows only)
- Netbeans (Windows/Linux/Mac)

For our purposes in this course, it will **not** matter on which platform you write the code.

Visual Studio has limitations at the University and at home, but is probably the best choice.

**CLICK HERE** to get started.
Unlike higher level programming languages, there are almost no intrinsic functions in C++

This includes the ability to print to screen.

We can **include** standard libraries for:

- Input/Output
- Advanced Storage
- Strings
- Mathematical functions

The syntax for including libraries is:

```cpp
#include <library_name>
```

Include statements must appear before any other statements.
A simple example of the standard input/output library:

```cpp
// Divide two integers
#include <iostream>
using namespace std;

main()
{
    int i,j;
    cout << 'Input two integers' << endl;
    cin >> i >> j;
    cout << ' i/j = ' << i/j << endl;
}
```
DATA AND VARIABLES

**COMPUTERS ARE NOT HUMANS!**

Computers can only follow the instructions they are given. If they are not doing what you want it is because you haven't told them explicitly what to do!

- One of the most common mistakes when you programming is caused by swapping between data types
- Program may not work quite how we want...
- So we must understand how the computer uses data.
Computers are not humans!

Computers can only follow the instructions they are given. If they are not doing what you want it is because you haven't told them explicitly what to do!

- One of the most common mistakes when you programming is caused by swapping between data types
- Program may not work quite how we want...
- So we must understand how the computer uses data.
Computers are not humans!

Computers can only follow the instructions they are given. If they are not doing what you want it is because you haven't told them explicitly what to do!

- One of the most common mistakes when you programming is caused by swapping between data types
- Program may not work quite how we want...
- So we must understand how the computer uses data.
There are six basic data types in C++:

- **char** – a character
- **(short/long) int** – integers with different sizes
- **float** – single precision real number
- **(long) double** – double (or higher) precision number
- **bool** – true or false binary number
- **void** – this is used when a function doesn’t return a value

The precision of numbers will affect the maximum and minimum values they can take and also how big errors are.
A variable is a named location in memory, used to store data.

We may declare variables anywhere in the code.

Variables will be localised to the block in which they are declared.

What is the output from the following?

```cpp
int i=0;
cout << " i= " << i << endl;
{
  int i=10;
  cout << " i= " << i << endl;
}
cout << " i= " << i << endl;
```
A variable is a named location in memory, used to store data.

We may declare variables anywhere in the code.

Variables will be localised to the block in which they are declared.

What is the output from the following?

```cpp
int i=0;
std::cout << " i= " << i << std::endl;
{
    int i=10;
    std::cout << " i= " << i << std::endl;
}
std::cout << " i= " << i << std::endl;
```
Using Variables

- We can assign a variable a value using ‘=’

```plaintext
data_1 = 10. + 21.5;
```

- The data types on both sides of ‘=’ must be compatible
- We can add/subtract, multiply/divide standard data types

```plaintext
double a, b, c;
b = 5. ; c = 4.1;
a = 10. * b + c;
```
**Using Variables**

- We can assign a variable a value using `='

```plaintext
data_1 = 10. + 21.5;
```

- The data types on both sides of `=' must be compatible
- We can add/subtract, multiply/divide standard data types

```plaintext
double a, b, c;
b = 5. ; c = 4.1;
a = 10. * b + c;
```
Other Operators

- We call +−*/ operators
- Other operators are:
  - a%b :: returns a modulus b
  - a++ :: increment a by 1
  - a-- :: decrement a by 1
  - a+=b :: set a equal to a plus b
  - a*=b :: set a equal to a times b
**Arrays**

- An array, like an array in maths is a collection of variables of the same type.
- We declare and reference arrays using square brackets `[]`.

```cpp
int array[100]; // 100 integer array
array[0] = 0;
array[1] = 1 + array[0]
```

- Arrays are indexed from 0, and this **cannot** easily be changed.
- Multidimensional arrays are declared in the obvious way.

```cpp
int array_2D[5][5]; // 2D array
array_2D[0][0] = 0;
```
An array, like an array in maths is a collection of variables of the same type.

We declare and reference arrays using square brackets \([\]\).

```plaintext
int array[100]; // 100 integer array
array[0] = 0;
array[1] = 1 + array[0]
```

Arrays are indexed from 0, and this cannot easily be changed.

Multidimensional arrays are declared in the obvious way:

```plaintext
int array_2D[5][5]; // 2D array
array_2D[0][0] = 0;
```
What is in a name?

- Each variable requires a unique name (in the same scope)
- In large projects this becomes very difficult
- A namespace is like adding a *surname* to a variable
- The prefix `std` means functions from the standard library
- We can use the statement:

```
using namespace std;
```

to assume an undeclared function, data type or variable is in the standard library.
We use *stream* variables to access the screen, keyboard and files.

They allow us to associate a name with a physical output.

We need to include stream libraries at the top of the program

```cpp
#include<iostream>
using namespace std
main()
{
    int i
    cout << " Enter a number. " << endl;
    cin >> i; // read in a number
    cout << " Your number is " << i << endl;
}
```
**Simple Input and Output**

- `cout` is the standard screen variable, and `cin` the standard keyboard variable.
- To pass data to and from the *stream* we use the `<<` and `>>` operators.
  - `<<` data is passed right to left, in the example the string is passed to `cout`.
  - `>>` data is passed left to right, in the example the integer is passed from `cin` to `i`.
- Multiple bits of data can be passed to the stream by stringing them together in the same command.
- Use `endl` to finish a line.
**Simple Input and Output**

- cout is the standard screen variable, and cin the standard keyboard variable
- To pass data to and from the *stream* we use the `<<` and `>>` operators.
- `<<` data is passed right to left, in the example the string is passed to cout
- `>>` data is passed left to right, in the example the integer is passed from cin to i
- Multiple bits of data can be passed to the stream by stringing them together in the same command.
- Use `endl` to finish a line.
SIMPLE INPUT AND OUTPUT

- **cout** is the standard screen variable, and **cin** the standard keyboard variable.
- To pass data to and from the *stream* we use the `<<` and `>>` operators.
- `<<` data is passed right to left, in the example the string is passed to `cout`.
- `>>` data is passed left to right, in the example the integer is passed from `cin` to `i`.
- Multiple bits of data can be passed to the stream by stringing them together in the same command.
- Use `endl` to finish a line.
To read and write to files we must include the `fstream` library.

Input streams have type `ifstream`, and output streams `ofstream`.

```cpp
ifstream file_input; // an input file stream
 ofstream file_output; // an output file stream
```

`ifstream` and `ofstream` have intrinsic functions to open and close files.

We can also check if the file is open with the `is_open()` function.

```cpp
file_input.open("input.in"); // open file input.in
if(file_input.is_open()) // check file is open
```
Topics:
- Computers and Programs;
- Syntax of C++;
- Data and Variables;
- Input and Output.

Aims:
- Understand the idea of programming a computer;
- Write a simple program to input and output data.