First Year Computing (C++ Course)
Dr John Joss Chinn

Week 1.

Outline of the Course.

Why use Computer Programs?

The basic idea of a computer program, as with many other aspects of computing in general, is the manipulation of data. In structural mechanics one may wish to model the deformation of a component under a given loading. In fluid dynamics one could simulate the flow in or around an object or component. In the modelling of an internal combustion engine one may desire both to model say the slight temperature expansion of the component parts, the chemical reactions of combustion, the flow of the fuel/air mixture and resultant exhaust gasses etc. In business and Administration one may wish to manipulate a certain piece of data present in many thousands of individual data files. This may be for a census of a certain professions salaries, to ascertain the price fluctuation of a certain service or product with time, or whatever. Many aspects of Operational Research (scientific methods applied to business and even military strategy) require the use of computer programs. For every day applications, all of the software on your PC, including the operating system, are, of course, computer programs.

Why C++?

You may be pleased to note that in this basic introductory course in computing you will not be expected to write programs to perform any of these sophisticated tasks. The purpose of this course is simply to introduce you to computer programming. There are many computer languages in use at the present, C, C++, FORTRAN, BASIC, JAVA, etc. C++ is the successor of C. C++ now has wide usage, both in Science and Technology and in Business Administration. C++ is a modern computer language. It is perhaps easier to go from C++ to another language than visa versa. Most PC software is written in C++, if you ever want to get into that business. C++ is a mid-level language. Low level languages include ‘machine code’ and Assembler. These are languages which the computer understands, very lean, i.e. they contain little which the computer does not need. They work directly with the processor. However they do not contain very much which humans do need! Not very popular nowadays. High-level languages include BASIC and FORTRAN. Every program will include a whole host of stuff which the computer does not need. In C++ you need to include only the stuff which is necessary for your particular program, yourself. In this way it is leaner than BASIC and FORTRAN but, unlike machine code, is fairly readily understandable to humans too.

So What is in the Course?

The course is over twelve weeks, ten of which are lectures. It is simply not possible for you to learn all about C++ in that time and it really is not the purpose of the course for you to attempt to do so. The purpose of the course is to get you to become familiar with the way in which one writes computer programs. To that end you will find that if you subsequently go off and learn to

write computer programs in FORTRAN or BASIC or whatever, then the methods that you will have learned in this course will be of benefit to you. Eight of the ten weeks of lectures are dedicated to learning about C++. Within those eight weeks you will also learn a little of how to implement a computer programs to perform certain tasks which you will need in your engineering profession. For instance, solving for the roots of an algebraic equation using the Newton-Raphson method, or how to rearrange a list of numbers into ascending or descending order, in a 'Bubble Sort', routine, also how to multiply two matrices, etc. During the other two lecture weeks you will learn (a) how to solve simultaneous equations using matrices in a computer program, "Gaussian Elimination with Partial Pivoting" and (b) how to find an equation to fit a smooth curve to a set of given data points using "Natural Cubic Splines", this will actually be for your assignment.

Which Books?

We will be writing basic 'console' applications, and not visual (windows) applications, within Borland C++ Builder. That is the name of the 'compiler', the PC software package which turns your 'source code' (your C++ program, `filename.cpp`) into a usable executable (`filename.exe`). As far as text books for the course is concerned, my advice is the following. Take a good look around the bookshops to see what you find is the easiest book for you to get along with. You will not have trouble finding books on C++! You may find that the cheapest source is the web. For instance www.amazon.co.uk or www.bol.com. You will also need to find some books on Numerical Methods (e.g. "Numerical Analysis". Richard Burden and Douglas Faires. Pub: Prindle Webber and Schmidt) and Linear Algebra (e.g. "Elementary Linear Algebra". Stanley Grossman. Pub: Wadworth).

History of Computing

Many books have been written on the history of computing. Again, look on the web. I only intend to write a page or two on this!

Computers

- **Abacus**, used in China for three or four thousand years, maybe older.
- **Stonehenge**, an astronomical calculator (depending on who you read!), built between 1900 to 1600 BC.
- **Quipus** (knotted cords), Incas, South America about 500 years ago.
- **Napiers Bones and Logarithms**. John Napier, Scotland 1550-1617.
- **Slide Rule**. Early seventeenth century, some place.
- **Pascal's Adder**. Blaise Pascal, France, 1642. Gears and wheels, invented to calculate taxes.
- **Leibnitz's Calculator.** Gottfried Wilhelm von Leibnitz, Germany, 1670. Similar to Pascal's but more reliable and accurate. Addition, subtraction, multiplication and division. Not changed much for the next hundred years. Things are a bit different today!
- **Weaving Loom**. Joseph Marie Jacquard, France, 1801. First to use the 'stored program concept', metal cards with holes in them to direct the pattern of the cloth. Eleven thousand produced in first ten years.

- **Difference Engine**. Charles Babbage, England, 1822. Used the stored program concept. Used to calculate polynomials. Military use:- height and distance of projectiles, etc.
- **Analytical Engine**. Babbage and Ada Augusta (the first programmer).
- **Punched card system similar to Babbages**. Herman Hollerith, USA. Used to compile the 1890 census. Used 'electrical sensors'.
- **International Business Machines**. Hollerith formed his own company in 1896. This became IBM in 1924.
- **Mark I**. Howard Aiken et al, USA, 1944. Babbage's Analytical Engine with electronics.
- **ENIAC**. John Astanasoff, Iowa State Uni., 1942. First totally electronic computer. Contained 18,000 thermionic valves and 1,500 relays. 20' x 40'. About 1,000 times faster than the Mark I. Military use.
- **UNIVAC**. J. P. Eckert and J. WE. Mauchly, Pennsylvania Uni., 1951. First commercially available general computer to be sold to science and industry. Programs had to be 'wired' in, using a sort of switchboard. Lots of 'down' time! Six figure price tag.
- **IBM 7090**. 1959. Used transistors in place of vacuum tubes. First 'Second-Generation' computer. Smaller, used less power and generated less heat.
- **IBM PDP-8.**1963, first minicomputer. $18,000.
- **IBM System/360.** 1964. First third generation computer, using integrated circuits. You may have spotted that this would have been a good time to buy IBM shares!
- **Intel 4004 Microprocessor**. Robert Noyce, 1971. Contained many thousands of transistors. A very large scale integrated circuit (VLSI). This was essentially the start of the fourth generation of computers.
- **Apple II**. Steven Jobs and Steve Wozniak, 1977. One of the earliest viable Personal Computers (PC's). This was followed by the first IBM PC in 1981.

Software

- **UNIX**. 1971. First operating system. Used in all types of computers from PC's to main frames. Very widely used today.
- **MS-DOS.** Microsoft. 1981. Operating System for PC's. A few years later Apple and then Microsoft produced Graphical User Interfaces (GUI's) to their operating systems, i.e. windows. This would have been a good time to buy Microsoft shares!
- **LINUX**. Linus Torvalds, Helsinki Uni., 1991. UNIX type operating system for PC's. Free on the internet. Difficult to set-up, but reportedly more stable than Windows. Going from strength to strength but for Geeks (you need to know what you are doing to use it effectively).
- **FORTRAN**. (FORmula TRANslation). Developed for the IBM 704 by John Backus et al, 1954-56. One of the earliest high level languages. Fourth revision by 1962. Fifth revision in 1977 "FORTRAN 77". This is in general use today, as are "FORTRAN 90" and "FORTRAN 95".
- **BASIC, COBOL, PASCAL**. All developed at a similar time
- **ALGOL 60**. first high-level block structured language, similar to C. 1960.

- **BCPL.** Martin Richards, Cambridge, Late 1960's. used on UNIX operating system. Early block-structured language.
- **B**. (for Bell). Developed at AT&T Bell labs, from BCPL, for the UNIX OS on a DEC PDP-11. 1970.
- **C**. (i.e. "not B", an inspired name!) 1972. Dennis Ritchie. AT&T Bell. In 1983 the American National Standards Institute (ANSI) standardised C for all types of computers. A program written in C ought therefore to be portable between PC's, workstations and mainframes. A similar standardisation process took place with FORTRAN. ANSI C was finally arrived at as the standard in 1989.
- **C++**. I will simply cut and paste the blurb from the AT&T Bell web site! (www.att.com/technology/history/chronolog/83cplus.html):-

Tired of the lack of expressiveness of mainstream programming languages of the day and of the slowness of experimental languages, AT&T researcher Bjarne Stroustrup built the first version of C++ in **1983**. C++ combines the expressive power of OOP (object-oriented programming) with the speed, compactness, and flexibility of C, its systems programming language predecessor, which was invented at AT&T ten years earlier. C++ matches C in efficiency and adds facilities for building larger, more easily maintained, and more reliable systems.

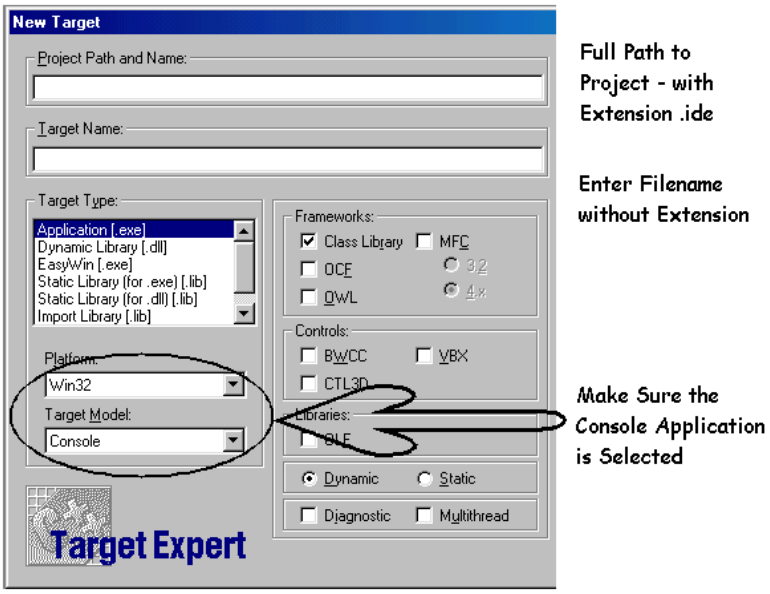
Stroustrup's creation, originally intended to improve the working lives of his colleagues, rapidly became one of the most influential programming languages in industry and academia worldwide. Today, upwards of a million programmers use C++ to write software for machines ranging from PCs to supercomputers. If you have used a computer, you have almost certainly used a program written in C++. Most PC and internet users do so daily. C++ is even used for software in gadgets such as cameras and elevators which are not usually associated with computers and programming.

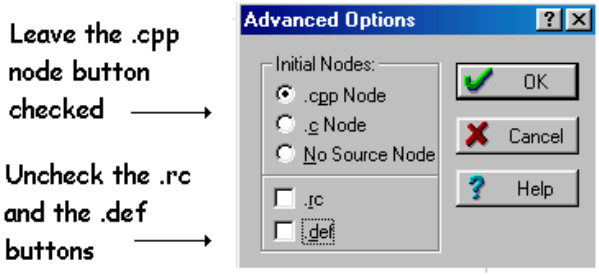
At AT&T, C++ has become embedded in transmission, switching, and operations systems. C++ is also used in scientific programming, data analysis, simulation, and other mainstays of telecommunications research. Constant research and refinement have kept C++ in the front line of systems development to this day.

Starting a Console Application in Borland C++

- Go to **START → Programs → Compilers → Borland C++ 5.02 (Windows NT only) → Borland C++**
- From **File** (on the menu bar, top left) select **New** and then **Project.** This opens a window (shown over the page) in which you need to do the following:-
- Alter the Project Path Name field type to read **C:\work\hello.ide,** for instance
- In the Target Name field type **hello**
- Alter the Target Model field to **Console**

- On the right of this window are some buttons. One labelled **Advanced** (little red hat with a propeller on it, [yeah, I know]) uncheck the **.rc** and **.def** boxes.





- Click OK in the advanced options and OK on the project window. This will open a window on the right of the screen.
- You should see an icon marked **hello.cpp.** Highlight this and press return. This will open a window for your source code. (Phew!)

- DO NOT INCLUDE “using namespace std;” at the top of your source code. (More about this next week.)
- Once you have typed in the source code (examples below). You need to ‘build’ the project to produce an executable. Go to ‘project’ on the menu bar and select ‘build project’
- You then need to run the executable. Go to ‘project’ on the menu bar and select ‘run’.

Input and Output. Some Examples

Let us look at a couple of simple examples of C++ programs. The first one simply prints out the words “`Hello World!`” to the computer screen, within a DOS box. This could be called `hello.cpp`.

```
#include<iostream> //needed for input and output
#include<conio>    //needed for getch();

void main()
{
   cout<<"Hello World!"<<endl;
   getch();       //just waits for a key press

}
```

Let us break this down into its component parts.

- The two lines at the top of the program are called headers. They contain extra source code which you need for your program. The first one, `iostream`, is necessary for input and output. The second one, `conio`, is necessary for the line `getch();` All that `getch()` does is make the program wait for input before continuing. Oddly enough you wouldn't need this line in the Microsoft C++ compiler. So you wouldn't need to `include` the header `conio` in Microsoft C++. So much for portability!
- Every C++ program contains a `main() { }` function. The stuff which goes into the function is contained within the two curly brackets.
- The bit that dos the work, for this program, is contained in the line, `cout<<"Hello World!"<<endl;` What this does is 'stream' out the text contained between the double quote marks. At the end of the line the word `endl` just puts in a carriage return, i.e. makes the cursor go to the next line.
- An important point is that most of the lines in a C++ source file end with a semi-colon (`;`).
- The text beginning with the two forward slashes (`//`) are simply comments.

Let us look at a slightly more complicated C++ source file.

```
#include<iostream>   //needed for input and output
#include<conio>      //needed for getch();

void main()
{
   int a,b;
   float c,d,e;

   cout<<"Please type a value for a ? ";
   cin>>a;
   cout<<"Please type a value for b ? ";
   cin>>b;

   cout<<" a + b = "<<(a+b)<<endl;
   cout<<" a - b = "<<(a-b)<<endl;
   cout<<" a * b = "<<(a*b)<<endl;
   cout<<" a / b = "<<(a/b)<<endl;

   cout<<endl;
   cout<<"Please type a value for c ? ";
   cin>>c;
   cout<<"Please type a value for d ? ";
   cin>>d;
   e=c/d;
   cout<<" c / d = "<<(c/d)<<endl;
   cout<<" c / d = "<<e<<endl;
   getch();         //just waits for a key press
}
```

This little piece of source code just deals with some simple input and output. You might want to call it `io1.cpp`.

- The line `int a,b;` initialises two variables, a and b, as integers.
- The line `float c,d,e;` initialises the three variables as real numbers, or 'floating point' numbers.
- The line `cout<<"Please type a value for a ? ";` Is similar to the line in the Hello World! program. It simply puts a message up on the screen.
- `cin>>a;` waits for the user to type in a number.
- You can probably figure out for yourself what the line
  `cout<<" a + b = "<<(a+b)<<endl;`
  does. The next three lines are similar.
- You will notice that the line
  `cout<<" a / b = "<<(a/b)<<endl;`
  prints out an integer, even if you divide say 10 by 3.
- To get around this, if you need a floating point (real number) answer, then you must use floating point variables.
- You will see that the lines
  `cout<<" c / d = "<<(c/d)<<endl;` and
  `cout<<" c / d = "<<e<<endl;`
  do exactly the same thing.