The University Of Sheffield.

Introduction to C Programming

The University Of Sheffield.

Who we are

Research & Innovation Support in IT Services

Dr Michael Griffiths

Dr Norbert Gyenge

The University Of Sheffield.

Course Outline

1. Introduction to C and Structured Programming
2. Using functions to build programs, managing data and computer memory
3. Data Structures, Strings and File Input/Output

The University Of Sheffield.

Course Format

- Course slides
- Basic C-programming examples - highlighting techniques and syntax
- Demonstrate techniques with practical examples
- Frequent breaks for practice sessions

The University Of Sheffield.

Course pre-requisites

- Course material
  - `git clone --branch introtoc2022v1 https://github.com/rcgsheffield/introc`
  - `http://rcg.group.shef.ac.uk/courses/introc`
- A C-Compiler or IDE
  - Codeblocks - https://www.codeblocks.org/downloads/
  - MSys2 - https://www.msys2.org/
  - Eclipse IDE (more advanced) - https://www.eclipse.org/downloads/packages/release/2022-06/r/eclipse-ide-cc-developers

The University Of Sheffield.

Resources

- Cplusplus
  - https://cplusplus.com/
- W3 Schools
  - https://www.w3schools.com/c/

The University Of Sheffield.

Learning Outcomes

- Layout and Syntax of a program
- Compiling and Running a program
- Programming Structures
- Data Types and Variables

The University Of Sheffield.

Program Development Steps

- Edit
  - Create program and store on system
- Preprocessor
  - Manipulate code prior to compilation
- Compiler
  - Create object code and store on system
- Linker
  - Link object code with libraries, create executable output
- Loader
- Execution
  - CPU executes each instruction

The University Of Sheffield.

Program Structure

- Collection of Text files
  - Source files
  - header files
- Resource files
- Source file layout
  - Function layout
- Program starts with a function called main
- Pre-processor directives

The University Of Sheffield.

Layout and Syntax of a C Program:Hello World

```
/* This is a hello world program*/

#include <stdio.h>    /*pre-processor statement*/

/*The program starts with a main function*/
/*This program will return an integer result*/
int main()
{

        /*Blocks of program statements are enclosed by pairs of
         * curly braces at the beginning and end of the block*/

        /*Use the printf function to Display a welcome message on the users screen*/
        printf("Welcome to the C Language!\n);"


   /*The program finishes when the final statement in the main program block
    * is executed or a return statement is reached*/

        /*Return the integer result 0 as the output of the program*/
        return(0);
}
```

The University Of Sheffield.

Features of a simple C-Program

- A C-Program is just a text file you can edit with most text editor
- Lots of comments - Enclosed by /* */
- Program blocks enclosed by curly braces {}
- Statements terminated with a ;
- Preprocessor statement
  - #include <stdio.h>
  - Enables functions to call standard input ouput functions (e.g. printf, scanf)
  - Not terminated with a ;
- printf is a function call from the standard C-library which uses escape sequence characters e.g. \n newline

```c
/*Bisection method for finding roots*/

/* here is an example use of the while statement
// which is used for finding the root of a polynomial
// which is known to lie within a certain interval.
// a is the lower value of the range
// b is the upper value of the range */
#include <stdio.h>
#include <math.h>
#include <float.h>

/*
  Note FLT_MIN, FLT_MAX and FLT_EPSILON
  defined in float.h
*/

float sign(float f){return(fabs(f)/f);}

int main(char **argv, int argc)
{
/*Main routines here see next slide*/
        return 0;
}
```

```
int main(char **argv, int argc)
{

        float x,fx;
        float a = 0;
        float fa = -FLT_MIN;
        float b = 3;
        float fb = FLT_MAX;
```

```
  while( fabs(b-a)>(FLT_EPSILON*b))
  {
    x = (a+b)/2;
    /*The function whose root is to be determined*/
    fx = pow(x,3)-2*x-5;
    if(sign(fx)==sign(fa))
    {
      a = x;
      fa = fx;
printf("a=%f fa=%f\n",a,fa);
    }
    else
    {
      b = x;
      fb = fx;
printf("b=%f fb=%f\n",b,fb);
    }

  }
  printf(" The root is :%f\n",x);
```

The University Of Sheffield.

Further Sessions

- Building Applications using Make
- From C to C++
- Boost your programming using the standard template libraries