Introduction to the C Programming Language

Michael Griffiths
IT Services
The University of Sheffield
Email m.griffiths@sheffield.ac.uk

Course Outline

- Part 1
  - Introduction to C Programming
  - Application development tools
- Part 2
  - Structured Program Development and Program Control
- Part 3
  - Functions
  - Pointers and Arrays
- Part 4
  - Characters and Strings
  - Data Types Structures
- Part 5
  - File Processing
  - Further Topics

Outline

- Introduction
- Structured program development
- Application development tools
- Compiling applications

Course Material

- C language (clanguage folder)
  - Examples illustrating basics of C programming
- Case Studies (casestudies folder)
  - Example research problems
  - Data Analysis (patientdatastudy)
  - Running Computational Models Monte Carlo (montecarlo)
- Numerical analysis
  - Example c programs for basic numerical analysis applications

The University Of Sheffield.

The University of Sheffield Research IT

Development Environment for Building Programs

- Codeblocks - http://www.codeblocks.org/
- Eclipse CDT - https://www.eclipse.org/cdt/
- Visual Studio – Free Community edition - https://visualstudio.microsoft.com/downloads/
- MinGW
  - Information about MinGW is here http://mingw-w64.org/doku.php

Examples

- C language
- Numerical Analysis Examples
- Scenarios
  - Patient data analysis study
  - Modelling the van-der-poll oscillator
  - Monte-carlo Ising Model
  - Solving the Schrodinger Equation - Numerov method for numerical integration

Introduction

- Developed late 70's
- Used for development of UNIX
- Powerful
  - If used with discipline
- ANSI Standard C
  - **C90** ANSI/ISO 9899: 1990
  - **C11** ISO/IEC 9899:2011 (formerly C1X)
    - the current standard for the C programming language, replaces the previous C standard, informally known as C99

Program Development

- 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

Parts of C Language

- Keywords
- Functions
- Operators
- Values and variables

Keywords

- This is a list of reserved keywords in C. Since they are used by the language, these keywords are not available for re-definition.

auto	float	signed
break	for	sizeof
case	goto	static
char	if	struct
const	inline (since C99)	switch
continue	int	typedef
default	long	union
do	register	unsigned
double	restrict (since C99)	void
else	return	volatile
enum	short	while
extern		

Functions

- Functions enable grouping of commonly used code into a reusable and compact unit.
- In programs containing many functions main should be implemented as a group of calls to functions undertaking the bulk of the work
- Become familiar with rich collections of functions in the ANSI C standard library
- Using functions from ANSI standard library increases portability

Operators

- Arithmetic operations
  =, -, /, %, *
- Assignment operations
  =, +=. -=, *=, %=, /=, !
- Increment and decrement (pre or post) operations
  ++, --
- Logical operations
  ||, &&, !
- Bitwise operations
  |, &, ~
- Comparison
  <, <=, >, >=, ==, !=

Values and Variables

- A variable is a container for a value we can have different types
  - Characters
  - Integer Values
  - Floating-point values
  - Memory locations

Variables

- Variables of the same type are compared using the comparison operator **==**
- Variable declaration using the assignment operator **=**

```
float myfloat;
float fanother=3.1415927;
```

- Other types using unsigned and long
  - long double, long int, short int, unsigned short int
- Precision and range machine dependent

Variable Types

Type	Size (Bytes)	Lower	Upper
int	4	-2^{31}	$+2^{31}-1$
float	4	$-3.2X10^{32}$	$3.2X10^{32}$
double	8	$-1.7X10^{302}$	$1.7X10^{302}$
char	1	-	-
unsigned char	1	0	255
unsigned short int	2	0	65536
short int	2	-32768	32767

Program Structure

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

Source file layout

- program.c

```
Pre-processsor directives
Global declarations

main()
{
     ……….
}

function1()
{
    …………
}

function2()
{
    ………….
}
```

Function Layout

```
vartype function(vartypes )
{
    local variables to function

    statements associated with function

        ……..
        ……..

}
```

Invoking the Compiler

- Compiling FORTRAN Programs
  - g77 –o mycode [options] mycode.f
- Compiling c/c++ Programs
  - gcc –o mycode [options] mycode.c

Example - Demonstration

- Compile the program welcome to C
  - Find the errors (the compiler will help)
- Add text to the puts statement in empty.c compile and run the code
- Challenge
  - Add another puts statement to empty.c, compile and run the code

Hello World

```
/*Program1: Hello World*/

#include <stdio.h>

main()
{
printf("Welcome to the White Rose Grid!\n");

/*Welcome banner on several lines*/
printf("Welcome to the \n \t White Rose Grid!\n");
}
```

Features of Hello World

- Lots of comments
  - Enclosed by /*   */
- 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 uses escape sequence characters

  e.g. \n   newline

  \t   tab character

Standard Conforming Hello World

```
/*Program1: Hello World*/

#include <stdio.h>

int main(void)
{
printf("Welcome to the White Rose Grid!\n");

/*Welcome banner on several lines*/
printf("Welcome to the \n \t White Rose Grid!\n");
        return 0;
}
```

Compilation

- To compile the program myprog.c using the GNU C Compiler
  - gcc myprog.c –o myprog
- Example compile arith.c
  - Modify program arith.c to test the effect of the decrement and increment operations
  - Modify program arith.c and test the assignment operations

Input and Output

- To process data computer programs need to input that data
- Processed data needs to output to the user
- The puts function can be used to display output
  - `puts("Welcome");`

Character Input and Output

- getchar()
- putchar()
- Require stdio.h header
- Work with integer values
- Stream functions

Input and Output Using stdio.h

- printf
  - Provides formatted input and output
  - Input for printf is a format specification followed by a list of variable names to be displayed
  - printf(“variable %d is %f\n”, myint, myfloat);
- scanf
  - Provided an input format and a list of variables
  - scanf(“%d”, &myint);
  - Note variable name has & in front
- Program arith.c is an example of the arithmetic operations, printf and scanf.

Input/Output Demonstration

- Examples in the folder clanguage/inputoutput
- Run the examples chario1...3.c
  - Change split the line in chario1.c so that putchar is used
  - chario2..3 demonstrate stream output
- Run the printf.c example
- Run scanf1.c and scanf2.c demonstrating string and integer input
  - For the scanf1.c change the input to a float

Example of printf and scanf

- Example program arith.c
  - Worksheet problem 3, use the decrement (--) and increment operators(++) to modify the variables sum and difference
  - Add a line requesting the user to input a floating point variable called f1
  - Add a line to multiply the new variable f1 by the variable sum
  - Add a line to display the variable f1

```
 /*Request input from the user*/
printf("Enter the first integer\n");
scanf("%d", &i1);       /*Read in the integer*/
printf("Enter the second integer\n");
scanf("%d", &i2);
```

Escape characters

Escape Sequence	Description
\n	Newline, position cursor at the start of a new line
\t	Horizontal tab, move cursor to the next tab stop
\r	Carriage return. Position cursor to the beginning of the current line; do not advance to the next line.
\a	Alert, sound system warning beep
\\	Backslash, print a backslash character in a printf statement
\”	Double quote print a double quote character in a printf statement.

Format Specifiers for printf and scanf

Data Type	Printf specifier	Scanf specifier
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
short	%hd	%hd
char	%c	%c

Practical Examples – basic coding

- Inspect, Compile and run the following
- Finding a root using the Bisection method
  - Modify the input function and experiment with the bracketing of the root
  - Control statements to be studied next session
    - While statement
    - If statement

Summary

- Program Structure
- Defining and using variables and values
- Displaying output
- Reading input
- Compilers and development Environments