C Programming Basics

Ashish Nanda

Ashish.Nanda@uts.edu.au

Introduction

- The C programming language is a general purpose programming language, which relates closely to the way machines work.
- Although C can be considered as "hard to learn", C is in fact a very simple language, with very powerful capabilities.
- C is a very common language, and it is the language of many applications such as Windows, the Python interpreter, Git, and many more.
- C is a compiled language - which means that in order to run it, the compiler (GCC) must take the code that we wrote, process it, and then create an executable file.

Header Files

- Every Program starts with a header, they define certain set of functions.
- Some basic Headers you should know about:
  - **stdio.h** – file and console IO: *perror, printf, open, close, read, write, scanf,* etc.
  - **stdlib.h** - common utility functions: *malloc, calloc, strtol, atoi,* etc.
  - **string.h** - string manipulation: *strlen, strcpy, strcat, memcpy, memset,* etc.
  - **errno.h** –used for reporting system errors: *errno*
  - **math.h** – math functions: *ceil, exp, floor, sqrt,* etc.
  - **time.h** – time related facility: *asctime, clock, time_t,* etc.

The Preprocessor

- The C preprocessor permits you to define simple macros that are evaluated and expanded prior to compilation.

- Commands begin with a ‘#’. Abbreviated list:
  - **#define** : defines a macro
  - **#undef** : removes a macro definition
  - **#include** : insert text from file
  - **#if** : conditional based on value of expression
  - **#else** : alternative
  - **#elif** : conditional alternative

The Basic Structure

The basic structure starts off with the header information.

This is followed by any preprocessors

And finally the Main Function:

The first code which will run will always reside in the main function. For example:

```
    int main (void) {
        ….your code
    }
```

```
#include <stdio.h>
#include <stdlib.h>


#define a = 10;


int main (int argc, char *argv[])
{
        if (a == 10)
        {
                printf("value of a is 10 !\n");
        }
        exit(0);
}
```

Lets try a simple program

First, we need to open a file to write our code in. Open the terminal, using your preferred terminal, create a hello.c file:

> gedit hello.c

Type the code shown on the right and same the file.

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
		printf(“Hello World!\n”);
		exit(0);
}
```

Compiling the program

Once you file is saved, go back to the terminal and type the following command:

```
> gcc -o hello hello.c
```

Here ‘-o’ stands for output file, ‘hello’ is the executable file and ‘hello.c’ is our program file.

Note: The executable file ‘hello’ does not have an extension as it’s a converted file.

If the terminal does not show any errors, you program has successfully complied. To run the program, use:

```
> ./hello
```

Lets start with Data Types

- Integers - whole numbers which can be either positive or negative. Defined using:

char	-128 to 127
short	-32768 to 32767
int	-32768 to 32767
long	-2147483648 to 2147483647

- Unsigned integers - whole numbers which can only be positive. Defined using:

unsigned char	0 to 255
unsigned short	0 to 65535
unsigned int	0 to 65535
unsigned long	0 to 4294967295

- Floating point numbers - real numbers (numbers with fractions). Defined using:

float	Approximate precision of 6 decimal digits
double	Approximate precision of 14 decimal digits.

0.0

Declaring a Variable

Correct	Incorrect	What is Wrong
int x, y, z;	**int** if, 1, -p;	Don’t use symbols, numbers or reserved characters.
short number_one;	**short** number+one;	Only ‘_’ can be used to combine words
long TypeofCar;	**long** #number	Missing ‘;’
unsigned int positive_number;	**unsigned int** num = -4;	Unsigned is always positive
char Title;	**char** My Title;	Do not use space
float commission, yield = 4.52;	**float** a b = 4.52;	Multiple Variables need to be separated with a comma ‘,’
int my_data = 4;	**int** my_data = a;	Wrong input

Example of Reserved Characters

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

Input and Output

Standard Output can be done using printf()

- It can be used to output a statement:

> printf (“Hello World \n”);

- It can also be used to print a variable:

> printf (“The value is = %d”, a);

  - Here %d identifies the type of variable and position where the value is inserted and a is our variable defined outside the quotes.

Standard Output can be done using scanf()

- It can be used to get user input:

> scanf (“%d”, &a);

  - Here %d identifies the type of value which is saved in a (‘&’ will only be used before the variable for scanf as it refers to the address of the variable where the value is stored)

Input and Output

Both printf() and scanf() can have multiple values

printf can be used to print multiple variables:

> printf (“The values are = %d and %d”, a, b);

Basic operations can be performed within the statement:

> printf (“The sum is %d”, a+b);

scanf can be used to get multiple user input:

> scanf (“%d %d”, &a, &b);

Specifier	Data Type
%d	int
%c	char
%f	float
%lf	double
%hd	short int
%u	unsigned int
%li	long int
%lu	unsigned long int
%c	signed char
%c	unsigned char

Basic Operators and String Constants

Symbol	Operator
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus Division

String Constant	Function
‘ ’	Single quotes are used to define a character value
\n	Used to change line (Similar to pressing Enter)
\t	Used to give spacing (Similar to pressing Tab)
\\	Will give an output: \
\“	Will give an output: “

Lets try this together !

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
        int int1, int2, sum;
        printf("Enter first integer \n");
        scanf("%d", &int1);
        printf("Enter second integer\n");
        scanf("%d", &int2);
        sum = int1 + int2;
        printf("Sum is %d \n", sum);
        return 0;
}
```

Logical and Relational Operators

Symbol	Logic
&&	AND
||	OR
!	NOT

Symbol	Relation
<	Less than
<=	Less than or equal to
=>	More than or equal to
>	More than
==	Equal to
!=	NOT Equal to

Wrap-up Challenge

Using all the information from above, try to make a simple program that takes a user First name and Last name as separate inputs and Outputs a greeting using their name.

For Example:

Input:

```
> John

> Doh
```

Output:

```
> Hello John Doh, How are you doing?
```