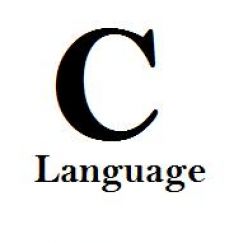
**C programming report**



**Content**

1. Aim 3
2. Objectives 3
3. Introduction to C programming 4
4. Features of C programming 4
   1. The use of **#define** and **#include** 4
   2. The use of ***printf* ()** and ***scanf* ()** 4/5
   3. Arrays 5
   4. Strings 6
   5. Pointers 7
5. Conclusion 7
6. References 8

# Aim

I aim to report on the role C has in terms of programming. I will more specifically identify certain features of C and explain how they work and their uses.

# Objectives

* To investigate C’s role and features
* To analyse the use of C programming

# Introduction

During the 1980’s, the original C language had evolved into what we know today, and it is now a mature, general-purpose language that is widely available on most if not all machines and operating systems. *“C is a general-purpose language which has been closely associated with the* [*UNIX*](http://cwis/AS/CC/GL/ccglu.html#5) *operating system for which it was developed - since the system and most of the programs that run it are written in C.”*

(<http://www.le.ac.uk/users/rjm1/cotter/page_06.htm>, 2011) C is a small language and in programming the smaller the better, C is appealing because of its powerful operators and unfettered nature. Therefore for a novice programmer, C is a good choice to start learning. However there are disadvantages, *“Its main drawback is that it has poor error detection which can make it off putting to the beginner. However diligence in this matter can pay off handsomely since having learned the rules of C we can break them. Not many languages allow this. This if done properly and carefully leads to the power of C programming.”*

(<http://www.cs.cf.ac.uk/Dave/C/node4.html#SECTION00420000000000000000>, 1999)

# Features of C

C programming is known to be very useful when it’s used to compile coding. It is fast and very reliable since it executes the programming at a fast rate and shows the errors in accurate manners. It is also known to be reliable when using elements such as arrays and pointers.

In C, a string is an array of characters, and an array name by itself is a pointer, a pointer is an address of an object in memory.

* 1. **#define & #include**

The C compiler has a pre-processor, lines that begin with **#** are called pre-processing directives. An example would be:

**#define LIMIT 100**

**#define PI 3.14159**

If these occur in a file that is being compiled, the pre-processor changes all occurrences of the identifier **LIMIT** to **100** and all the occurrences of **PI** to **3.14159** the only exception to this is when quoted in strings and comments. **#include** is a pre-processing directive that causes a copy of whatever is being included to be added at this point in the file when compilation occurs. *“The C system provides a number of standard header files. Some examples are stdio.h, string.h, and math.h. These files contain the declarations of functions in the standard library, macros, structure templates, and other programming elements that are commonly used.”* (Kelley, 1990)

* 1. **Printf() & Scanf()**

The function **printf()** is used for output, **scanf()** is used for input, the function prototypes for **Printf() & Scanf()** are included in stdio.h, therefore the header file should be included whenever the functions are used. *“The* ***printf statement allows you to send output to standard out.*** *For us, standard out is generally the screen (although you can redirect standard out into a text file or another command).”*

(<http://computer.howstuffworks.com/c5.htm>, 1999)The functions may include conversion specifications such as %d this would print the value of an expression as a decimal integer. *“To print the letters abc on the screen, we could use the statement* ***printf(“abc”);*** *Another way to do this is with the statement* ***printf(“%s”, “abc”);****”* (Kelley, 1990)

### Arrays

Arrays are used when many variables which are all the same type are needed. The elements of an array are accessed as a[0], a[1] and a[2] for example. The index of an array always starts at 0, an example of a working array would be:

#include <stdio.h>

int main()

{

int x;

int y;

int array[8][8]; /\* Declares an array like a chessboard \*/

for ( x = 0; x < 8; x++ ) {

for ( y = 0; y < 8; y++ )

array[x][y] = x \* y; /\* Set each element to a value \*/

}

printf( "Array Indices:\n" );

for ( x = 0; x < 8;x++ ) {

for ( y = 0; y < 8; y++ )

{

printf( "[%d][%d]=%d", x, y, array[x][y] );

}

printf( "\n" );

}

getchar();

}

(<http://www.cprogramming.com/tutorial/c/lesson8.html>, 2005)

### Strings

In C, a string is an array of characters using macros known as ***getchar*()** and ***putchar*(),** these macros are defined in*stdio.h*. Although different a macro is used in the same way a function is used, the macros ***getchar*()** and ***putchar*()** are used to read characters from the keyboard and to print characters on the screen respectively. An example of this would be:

/\* Have a nice day! \*/

#include <stdio.h>

#include <ctype.h>

#define MAXSTRING 100

main()

{

char c, name[MAXSTRING];

int i, sum = 0;

printf("\nHi! What is your name? ");

for (i = 0; (c = getchar()) != '\n'; ++i {

name[i] = c;

if (isalpha(c))

sum += c;

}

name[i] = '\0';

printf("\n%s%s%s\n%s",

"Nice to meet you ", name, "."

"Your name spelled backwards is ");

for (--i; i >= 0; --i)

putchar(name[i]);

printf("\n%s%d%s\n\n%s\n",

"and the letters in your name sum to ", sum, ".",

"Have a nice day!");

}

(Kelley, 1990)

* 1. **Pointers**

A pointer is an address of an object in memory, because an array name itself is a pointer, the uses of arrays and pointers are intimately related. In C, arrays, strings and pointers are closely related. An example would be:

**Char \*p, s[100];**

This creates the identifier **p** as a pointer to **char** and the identifier **s** as an array with 100 elements of type **char**. Both **p** and **s** are pointers to **char**, although **p** is a variable pointer while **s** is a constant pointer that points to **s[0]** . It is possible to increment **p** by adding **++** behind the pointer however this is not the same for s as it is a constant pointer and its value cannot be changed therefore the expression **++s** is wrong.

**5 Conclusion**

As a result of this report I have showed how C is a simple yet versatile programming language which is fast and reliable a good place to start for a novice programmer or a simple program design. The features described in this report **#define** and **#include,** ***printf* ()** and ***scanf* (),** Arrays, Strings and Pointers are all main features in C and as such will be used often and will be invaluable when creating C programs.

**6 References**

Al Kelley and Ira Pohl. (1990) *A Book on C Programming in C,* Redwood City, California: The Benjamin/Cummings Publishing Company, Inc.

Author: N/A (1997-2005) Arrays in C - Cprogramming.com [online]

[Accessed 5 November 2011] Available at:

<http://www.cprogramming.com/tutorial/c/lesson8.html>

Author: Dave Marshall (1997-1999) C Basics [online]

[Accessed 3 November 2011] Available at:

<http://www.cs.cf.ac.uk/Dave/C/node4.html#SECTION00420000000000000000>

Author: HowStuffWorks, Inc (1998-2011) HowStuffWorks “The Basics of C Programming” [online] [Accessed 4 November 2011] Available at:

<http://computer.howstuffworks.com/c5.htm>

Author: N/A (?-2011) A Brief History of C: [online]

[Accessed 4 November 2011] Available at:

<http://www.le.ac.uk/users/rjm1/cotter/>