Using C programming language

Comments

These are statements written, not to be compiled, but for the benefit of the programmer or others as reminders and general information about the program instruction (s);

To create a multiline comment, open it by a forward slash followed with an asterisk (/*) and close it with an asterisk followed with a forward slash (*/)

i.e. /* multiline comment */

To create a single line comment use double forward slash to open it

i.e. // single line comment

Pre-processor directives

Before a C program is compiled in a compiler, the source code is processed by a program called a pre-processor. This process is called pre-processing. Special Commands used in pre-processor are called pre-processor directives and they begin with "#" symbol. and must start in the first line of the program.

e.g. #include <file_name> which is a header file inclusion where the source code of the file "file_name" is included in the main program at the specified place

use include to state and be able to use specified functions in the program e.g. use of ***printf*** to display output or ***scanf*** to take a keyboard input by the user.

#include <stdio.h>

Std.h is C's standard library, which deals with standard inputting, and outputting of data and it is declared before the main function. The .h extension indicates that this is a header file.

And to use strings, use

#include <string>

Functions

A function is a block of code to perform a particular task. e.g. a function to add two numbers.

Each program in C has at least one function known as ***main***. C looks for this function and executes the code within it.

Main ( )

Any code outside the ***main*** function cannot be executed.

To define main

Begin by the word main followed by round brackets then put curly brackets (curly braces) {}within which to write the code.

Main ( ) { }

The curly braces define the beginning and end of a block of code.

The variables that you declare inside the block are **local** to that block.

Note: a global variable is defined outside the main function because a global variable applies to the entire program.

```
/* Sample C program that outputs the word "Hello" */

#include <stdio.h>

int main()
{
     printf("Hello\n");
     return 0;
}
```

The opening curly bracket instructs the start of a program. This is where execution of the code begins when the program is run, while the closing curly brace instructs the end of the program.

Declaring a function

The function is declared by stating the type of function, then its name

int main ( ) is the declaration of the main function for any C program.

The keyword ***int*** specifies the return type of the function main. It tells the compiler that this function will return a value on completion.

Variables

A variable is a container for a value. Its content can change (vary). It includes a variable name or identifier, and a value. eg **value_1 = 2,** where **value_1** is the variable name and **2** is the value it represents.

Variable Names

i) These must start with a letter or an underscore. The remaining characters can be letters, numbers or the underscore only.
ii) A variable name must not be a programming language keyword such as if, for, else, or while.
iii) Variable names are case sensitive so, Age, AGE, aGE and AgE could be names for different variables.

Which of the following are valid variable names?

int idnumber;

int transaction_number;

int __my_phone_number__;

float 4myfriend;

float its4me;

double VeRyStRaNgE;

float while;

float myCash;

int CaseNo;

int CASENO;

int caseno;

Values

Values include characters and numerical representing variable names.

Characters are alphabets.

Numerical values include integers (whole numbers), and floating point values

Data Types

i) I**nteger** variables are used to store whole numbers (e.g. 1, 100, 500, 37 etc.). There are several keywords used to declare integer variables, including int, short, long, unsigned short, unsigned long. The difference deals with the number of bytes used to store the variable in memory (how large the number is).
ii) **Float** variables are used to store floating point numbers. Floating point numbers may contain both a whole and fractional part, for example, 52.7 or 3.33333333.

Semicolons

In a C program, each individual statement within a program must be ended with a semicolon;

How to declare variables

Declare the data type then the name of the variable

int age for integer

e.g.

int age = 38;

Float average for decimal numbers

e.g.

Float piValue = 3.14159;

Input/output functions

C output functions

Output to the standard output device such as the screen or screen terminal. These require a standard I/O header file <stdio.h>

Printf

Puts

Putchar sends a character to standard output device

eg.

```
#include<stdio.h>


int main ()
{
char c;
        printf("Type any character:");


        c = getchar();
        printf("You typed the letter %c \n", c);


        return 0;}
```

puts (ie put string) – sends a string to a standard output device such as the screen or terminal window

A string is a single dimensional array of characters terminated by a null character

Writing sentences

To print a line to screen use printf and put the sentence in inverted commas within round brackets

e.g.

printf (“this is a sentence”)

\n takes the cursor to the next line while \t creates a tab

printf (“\n”)

Example

printf (“I am %d years old\n”, age)

% is a place holder where decimal (integer) age will appear if the code is run.

% f for a float

% c for a (single) character

% f for a string

% .5 f for a float five decimal places

%d for an integer

% ld for long decimal number

%% for a percentage sign

eg printf (“I am a %s\n”, “Programmer”)

C input functions

These are used to input data or values into the program

Getchar fetches a character from a standard input device eg keyboard

Using scanf

Scanf is a function that calls for user keyboard input

e.g.

To have a program that calls for the user to input his or first and second name

Daclare the first and second name variables

char firstName, lastName;

printf (“State your first name and second name”);

scanf (“% s % s”, &firstName, &lastName);

the % sign or modulus is used as a place holder while the ampersand (&) is used to as an address to the declared variable apart from an array which does not need an ampersand as an address

Another example,

```
int main()
{
        int age, years, extent;
        float true_age;
        int function;
        age = 30;
        years = 10;
        extent = age*years;
        printf("What is your function number ? ");
```

```
    scanf("%d", &function);
    printf("The number is %d\n\n" , extent/function);
    return 0;
}
```

To do calculations

Declare the variables

e.g.

inte num1 = 12, num2 = 15, numAns;

float decimal_1 = 1.2, decimal_2 = 1.5, decimalAns;

printf(“integer calculation %d\n\n”, num2 / num2);

printf(“float calculation % f \n\n”, decimal_2 / decimal_1)

Math operators

What is used to manipulate integers;

Symbol	Operation
+	**addition**
-	**Subtraction**
*	**Multiplication**
/	**Division**
++	**Increment the variable by 1**
--	**Decrement the variable by 1**

Uses BODMAS to order the Mth operations in FORMULA.

To use mathematical functions requires the use of <math.h> header to access the math library.

Some common math functions are sqrt, pow for power eg. P = (2, 8) determines the value of 2 rqaised to power 8,

rand for random numbers requires the <stdlib> header

```
#include<stdio.h>
#include<math.h>
int main ()
{
    int a;
    float r;
    printf ("State the integer to determine its square root: \n");
```

```
    scanf("%d", &a);
    r = sqrt (a);
    printf("The square root of the number is %2.f \n", r);
    return 0;
}
```

Comparison operators

Used to evaluate two or more values or expressions

Operators	Operations
= =	Is equal to
<	Less than
>	Greater than
< =	Less than or equal to
>=	Greater than or equal to
!=	Not equal
&&	And

If then else

```
#include <stdio.h>

int main ()
{
    int x, y;

    printf("state the values of x and y \n");

    scanf("%d%d", &x, &y);

    if (x > y)
    {
        printf("x is greater thhan y \n");
    }
    else if (x <= y)
    {
```

```
            printf("it is of no consequence \n");
        }

        else
        printf("Good condition \n");

        return 0;
}
```

Program to calculate roots of a quadratic equation

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

int main()
{
    double a, b, c, determinant, root1,root2, realPart, imaginaryPart;

    printf("Enter coefficients a, b and c: ");
    scanf("%lf %lf %lf",&a, &b, &c);

    determinant = b*b-4*a*c;

    // condition for real and different roots
    if (determinant > 0)
    {
    // sqrt() function returns square root
        root1 = (-b+sqrt(determinant))/(2*a);
        root2 = (-b-sqrt(determinant))/(2*a);

        printf("root1 = %.2lf and root2 = %.2lf",root1 , root2);
    }

    //condition for real and equal roots
    else if (determinant == 0)
    {
        root1 = root2 = -b/(2*a);

        printf("root1 = root2 = %.2lf;", root1);
    }

    // if roots are not real
    else
    {
        realPart = -b/(2*a);
        imaginaryPart = sqrt(-determinant)/(2*a);
        printf("root1 = %.2lf+%.2lfi and root2 = %.2f-%.2fi", realPart,
imaginaryPart, realPart, imaginaryPart);
    }

    return 0;
}
```