C Programming

**Why Learn C?**

* C is most commonly used programming language for writing Operating System.
* UNIX was the first operating system written in C. Later Microsoft Windows, Mac OS and GNU/ Linux were all written in C.

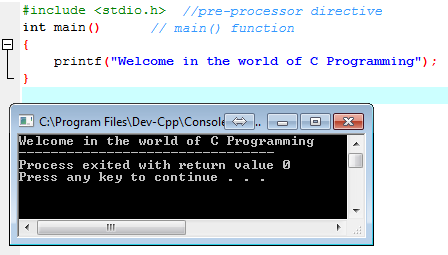
**Definition:**

The C Programming Language is standardized programming language developed in the early 1972s by Ken Thompson and Dennis Ritchie for use on the UNIX Operating System. C is prized for its efficiency and is most popular programming language for writing System Software, though it is also used for writing applications.

**Facts about C**

* C was invented to write an operating system called UNIX.
* C is a successor of B language, which was introduced around 1970.
* The UNIX OS was totally written in C by 1973.
* Today, C is the most widely used and popular System Programming Language.
* Most of the state-of-the-art software’s have been implemented using C.
* Today's most popular Linux OS and RBDMS My SQL have been written in C.

**Simple Program**



**Line 1**: the #include is a “pre-processor” directive that tell the compiler to put code from the header file stdio.h (**Standard Input Output Header File**) and add the contents of that file to this program. By including header files, we can gain access to many different functions (i.e. printf() function).

**Line 2**: this line tells the compiler that there is a function named main (every full c program begins inside a function called “main”) and that the function returns integer value. The “curly braces” (**{** and **}**) signal the **beginning** and **end** of the functions and other code blocks.

A **Function** is simply a collection of commands that do “something”. The main() function is always called when the program first executes.

**Line 4**: the printf() function is the standard C way to displaying output on the screen.

**Save** above program as a **filename.c** file and compile it

**Compiling Program:**

Compilation is basically **translation**. A Computer Program called the compiler takes out C Source Code and translates it into the **Binary Language** use by computers.

**Comments:**

Comments are added to make a program more readable to you but the compiler must ignore the comments.

1. Single Line Comment

// this is single line comment

1. Multi Line Comment

/\* this is

Multi Line

Comment

\*/

**One More Example:**

#include <stdio.h>

int main()

{

printf(“This is a line of text to output.\n”); // \n for new line

printf(“And this is another”);

printf(“line of text.\n\n”);

printf(“This is third line.”);

}

**Variable:**

Before we try to receive input, we must have a place to store that input. In programming, input and data are stored in **variables**. There are several different types of variables; when we declaring a variable, we must include the **data type** along with the name of the variable. Several basic types include char, int and float. Each type can store different types of data.

A variable of type **char** stores a single character, variables of type **int** store integers (numbers without decimal places) and variable of type **float** store numbers with decimal places.

**Definition:**

Variable is a named memory location which can store a value.

A variable name in C can be anything from a single letter to a word. The name of a variable must begin with an alphabetic letter or the underscore (\_) character which can be further followed by the following -

1. a……z (any letter from a to z)
2. A……Z (any letter from A to Z)
3. 0……9 (any digit from 0 to 9)
4. \_ (underscore character)

**Syntax:** to declare a variable you use the syntax

Datatype VariableName;

**Data Type:**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Range** |
| **char** | **1 Byte** | **-128 to 127** |
| **int** | **2 Byte** | **-32768 to 32767** |
| **long** | **4 Byte** | **-2,147,483,648 to 2,147,438,647** |
| **float** | **4 Byte** | **3.4e-38 to 3.4e+38** |
| **double** | **8 Byte** |  |

**For example:**

int a, b;

a = 32766; // correct

b = 40000; // Incorrect, because a integer data type can store -32768 to 32767

**Example 1:**

int number;

Data Type Variable Name

**Example 2:**

int number;

number = 400;

**Example 3:**

int a, b, c

a = 10;

b = 30;

c = 540;

**Example 4:**

int x, y;

x = 10;

y = x; // we can also assign the value of another variable.

We can have multiple variables of the same type, but we can’t have multiple variables with the same name. Moreover, we can’t have variables and functions with the same name.

A Final restriction on variables is that variable declarations must come before other types of statements. So in C, we must declare all the variables before do anything else:

**Wrong**

#include <stdio.h>

int main()

{

/\* wrong! The variable declaration must appear first \*/

printf(“Declare x next”);

int x;

}

**Correct**

#include <stdio.h>

int main()

{

int x;

printf(“Declare x next”);

}

**The Most Commonly Needed Conversion specifiers:-**

|  |  |  |
| --- | --- | --- |
| **Specifier** | **Meaning** | **Types Converted** |
| %c | Single Character | Char |
| %d | Integer | int |
| %f | Floating point number | Float, double |

**Reading Input**

We’ll be using the scanf() function to read in a value. Let’s look at the program

#include <stdio.h>

int main()

{

int value;

printf(“Enter Number ”);

scanf(“%d”, &value);

printf(“You Entered %d”, value);

}

We have seen the #include and main() function before; main must appear in every program, and the #include gives us access to printf (as well as scanf).

We have a string containing %d – this tells the scanf() function to **read in an integer**. The second argument of scanf() is the variable. Using **&** in front of a variable allows you to get its location and give that to scanf instead to the value of the variable.

**Read character**

#include <stdio.h>

int main()

{

int ch;

printf(“Enter any Character : ”);

scanf(“%c”, &ch);

printf(“You Entered %c”, ch);

}

**Operators**

1. **Arithmetic Operators**

+ Addition Operator

- Subtraction Operator

\* Multiplication Operator

/ Division Operator

1. **Assignment Operator**(=)
2. **Relational Operators**

> Greater than Operator

< Less than Operator

>= Greater than or Equals to Operator

<= Less than or Equals to Operator

== Equals to Operator

**Addition Operator (+)**

**Example 1**

#include <stdio.h> //Standard input output header file

int main()

{

int x, y, sum;

x = 10;

y = 450;

sum = x + y;

printf(“Sum = %d”, sum);

}

**Example 2**

#include <stdio.h> //Standard input output header file

int main() {

int x, y, z, sum;

printf(“Enter First No ”);

scanf(“%d”, & x);

printf(“Enter Second No ”);

scanf(“%d”, & y);

printf(“Enter Third No ”);

scanf(“%d”, &z);

sum = x + y + z;

printf(“Sum = %d”, sum);

}

**Multiplication Operator (\*)**

**Example 1**

#include <stdio.h> //Standard input output header file

int main()

{

int x, y, result;

x = 10;

y = 4;

result = x \* y;

printf(“Multiplication = %d”, result);

}

**Q. Write a C Program to calculate simple interest.**

#include<stdio.h>

int main()

{

int amt, tm, rate, si;

printf(“Enter Amount : ”);

scanf(“%d”, &amt);

printf(“Enter Time : ”);

scanf(“%d”, &tm);

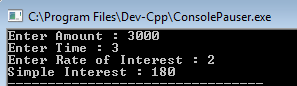
printf(“Enter Rate of Interest : ”);

scanf(“%d”, &rate);

si = (amt\*tm\*rate)/100;

printf(“\nSimple Interest : %d”, si);

}



**Decision Control Statements**

1. **If Statement**
   1. **If Statement**
   2. **If ... Else Statement**
   3. **If ... Elseif…Else Statement**
2. **Switch Statement**

**If Statement**

The structure of an If statement is as follow:

If (<condition>)

{

//statements

}

**Example**

#include <stdio.h>

int main()

{

int i;

i = 22;

if (i>=0)

{

printf(“No is Positive”);

}

}

**If Else Statement**

The structure of an If statement is as follow:

If (<condition>)

{

//statements

}

Else

{

//statements

}

**Example 1**

#include <stdio.h>

int main()

{

int i;

i = 22;

if (i>=0)

{

printf(“No is Positive”);

}

else

{

printf(“No is Negative”);

}

}

**Example 2(Write a program to check whether a given number is ever or odd.)**

#include <stdio.h>

int main()

{

int no;

no = 10;

if (no % 2 = = 0)

{

printf(“No is Even”);

}

else

{

printf(“No is Odd”);

}

}

****

**Example 3(Write a program to find greatest of two numbers.)**

#include <stdio.h>

int main()

{

int a, b;

a = 100;

b = 230;

if (a > b)

{

printf(“a is greater”);

}

else

{

printf(“b is greater”);

}

}

****

**If Else If Statement**

The structure of an If statement is as follow:

If (<condition>)

{

//statements

}

Else if (<condition>)

{

//statements

}

Else

{

//statements

}

**Example 1**

#include <stdio.h>

int main()

{

int age;

printf (“Please Enter Your Age :”);

scanf(“%d”, &age);

if (age<100)

{

printf(“You are Preety Young”);

}

else if (age == 100)

{

printf(“You are Old”);

}

else

{

printf(“You are really old”);

}

}

****

**Example 2(Write a program to Find greatest of two numbers.)**

#include <stdio.h>

int main()

{

int x, y;

x = 100;

y = 234;

if (x > y)

{

printf(“X is Greater”);

}

else if (y > x)

{

printf(“Y is Greater”);

}

else

{

printf(“X and Y are Equal”);

}

}



**Example 3**

#include <stdio.h>

int main()

{

int percent;

percent = 67;

if (percent>= 60)

{

printf(“First Division”);

}

else if (percent>=45)

{

printf(“Second Division”);

}

else if (percent>=33)

{

printf(“Third Division”);

}

else

{

printf(“Fail”);

}

}



**Switch Statement**

The structure of a Switch statement is as follow:

switch (variable>)

{

case this-value :

//code

break;

case this-value :

//code

break;

default :

//code

}

**Example 1**

#include <stdio.h>

int main()

{

int wdn;

wdn = 3;

switch (wdn)

{

case 1 :

printf(“Monday”);

break;

case 2 :

printf(“Tuesday”);

break;

case 3 :

printf(“Wednesday”);

break;

case 4 :

printf(“Thursday”);

break;

case 5 :

printf(“Friday”);

break;

case 6 :

printf(“Saturday”);

break;

case 7 :

printf(“Sunday”);

break;

default :

printf(“Wrong Input”);

}

}

**Looping Statements**

* Loops are used to repeat a block of code.
* There are three types of loops: While, Do – While and For loop.

1. **While Loop**
2. **Do While Loop**
3. **For Loop**

**While Loop**

The basic Structure is

while (condition)

{

// code to execute while the condition is true

}

**Example 1:**

#include <stdio.h>

int main()

{

int x = 1;

while (x <= 4)

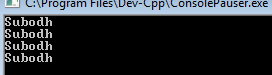
{

printf(“Subodh\n”);

x = x + 1;

}

}

****

**Example 2:**

#include <stdio.h>

int main()

{

int x = 1;

while (x <= 4)

{

printf(“%d”, x);

x = x + 1;

}

}

****

**Do While Loop**

Do – While loop are useful for things that want to loop at least once. The Structure is -

Do

{

// code

}

while (condition);

**Example 1:**

#include <stdio.h>

int main()

{

int x = 1;

do

{

printf(“Subodh\n”);

x = x + 1;

}

while (x <= 4);

}

**Example 2:**

#include <stdio.h>

int main()

{

int x = 1;

do

{

printf(“%d”, x);

x = x + 1;

}

while (x <= 4);

}

**Example 3:**

|  |  |
| --- | --- |
| #include <stdio.h>  int main() {  int x, no;  x = 1;  printf(“Enter Any Integer Number : ”);  scanf(“%d”, &no);  do  {  printf(“%d\n”, x\*no);  x = x + 1;  }  while (x <= 10);  } |  |

**For Loop**

For Loop is the most useful type. The syntax for a For Loop is

for (variable-initialization; condition; increment/decrement)

{

//code

}

**Example 1**

#include <stdio.h>

int main()

{

int x;

for (x = 1; x <= 10; x++)

{

printf(“%d”, x);

}

}



**Example 2**

#include <stdio.h>

int main()

{

int x;

for (x = 5; x >= 1; x--)

{

printf(“%d”, x);

}

}

