What is Instruction In 'C'

C instructions are the commands in the **program** that instructs the compiler to do certain action. Basically it gives the **instruction** to the compiler on how to achieve the goal of the **program**. For example, when we say add two numbers; **C** compiler will not understand how to do it.

There are three types of instructions in C. They are as follows:
1) Type Declaration Instruction
2) Arithmetic Instruction
3) Control instruction

1) Type Declaration instruction is used to declare the type of variables used in C. Any variable we want to use in the program must be declared before using it. This declaration is done using Type declaration instruction. This declaration is done at the beginning of the main() function. **Example: int a,b; float x,y; char ch;**

2) Arithmetic instructions in C are used to perform arithmetic operations on variables and constants. Here we will learn some new terms. These are operands and operators. The variables and constants on which arithmetic operation is done by arithmetic operators are called operands.
Example 1: int a=5, b=10, c;
C=a+b;
Here a, b, c are called operands and =, + are called as operators

Example 2:

```
int a;
float b, c, d;
a = 10;
b = 0.05;
c = 1.5;
d = a +b * c;
```

Here,
a is integer variable.
b, c and d are real variables.
= is the assignment operator.
+ and * are arithmetic operators.
10 is integer constant.
0.05 and 1.5 are real constants.

Three types of C Arithmetic Statement

1) ***Integer mode arithmetic statement*** : This is an arithmetic statement which all operands are either integer or integer constants.

```
int i, j, l, m;
i=i+1;
m=i* j +l;
```

2) ***Real Mode Arithmetic Statement*** : These are arithmetic statement in which all operands are either real constant or real variable.

```
float si, roi, p, q ;
si = roi*p*q/100.0;
```

3) ***Mixed mode arithmetic statements*** : this is an arithmetic statement in which some of the operands are integer and some of the operands are real.

```
int a, b, c, num ;
```

```
avg = ( a + b+ c + num)/4;
```

Control Instruction:

The ‘Control Instructions’ enable us to specify the order in which instructions in a program are to be executed. or the control instructions determine the ‘flow of control’ in a program.

There are four types of control instructions in C.

1. ***Sequence Control Instruction***: The Sequence control instruction ensures that the instructions are executed in the same order in which they appear in the program.
2. ***Selection or Decision Control Instruction***: Decision instructions allow the computer to take a decision as to which instruction is to be executed next.
3. ***Repetition or Loop Control Instruction***: The Loop control instruction helps computer to execute a group of statements repeatedly.
4. ***Case Control Instruction*** same as decision control instruction.

C input and output instruction

Input means to provide the program with some data to be used in the program and **Output** means to display data on screen or write the data to a printer or a file. C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.

All these built-in functions are present in C header files, we will also specify the name of header files in which a particular function is defined.

scanf() and printf() functions

The standard input-output header file, named stdio.h contains the definition of the functions printf() and scanf(), which are used to display output on screen and to take input from user respectively.

printf(): is an output formatted function which is used to display output on the screen.

Syntax: printf(“prompt string with format specifier”, list of variable);

Prompt string mean any string which you want to print on the screen. **Format specifier** are used to print the different type of value from memory. **List of variable** are the name which you want to get values from it.

Example for printf() function

```
#include <stdio.h>
int main()
{
  char ch = 'A';
  char str[20] = "fresh2refresh.com";
  float flt = 10.234;
  int no = 150;
  double dbl = 20.123456;
  printf("Character is %c \n", ch);
  printf("String is %s \n" , str);
  printf("Float value is %f \n", flt);
  printf("Integer value is %d\n" , no);
  printf("Double value is %lf \n", dbl);
  printf("Octal value is %o \n", no);
  printf("Hexadecimal value is %x \n", no);
  return 0;
}
```

Output:

```
Character is A
String is fresh2refresh.com
Float value is 10.234000
Integer value is 150
Double value is 20.123456
Octal value is 226
Hexadecimal value is 96
```

Scanf(): it is an input formatted function which is used to read the value from the keyboard and assign it into the variable during the program execution. Scanf() function waits the user response until type any value.

Format String	Meaning
`%d`	Scan or print an integer as signed decimal number
`%f`	Scan or print a floating point number
`%c`	To scan or print a character
`%s`	To scan or print a character string. The scanning ends at whitespace.

Syntax: scanf("format specifier", &list of variable);

In this function both the argument are compulsory . You must put **&** in front of the variable used in scanf. The reason why will become clear once you learn about **pointers**. It is easy to forget the & sign, and when you forget it your program will almost always crash when you run it.

#include Preprocessor directive

In the C Programming Language, the #include directive tells the preprocessor to insert the contents of another file into the source code at the point where the #include directive is found. Include directives are typically used to include the C header files for C functions that are held outsite of the current source file.

Syntax
#include <headier file>
Or
#include ”header file”

header_file

The name of the header file that you wish to include. A header file is a C file that typically ends in ".h" and contains declarations and macro definitions which can be shared between several source files.

Main() functions

The **main()** function is the first function in your program that is executed when it begins executing, but it's not the first function executed. The *first* function is **start()**, which is typically provided by the C runtime library, linked in automatically when your program is compiled. The details are highly dependent on the operating system and compiler toolchain.