e-Lecture on C Programming

for

B.Sc. Second Semester
Paper code: 201

Dr. Rajesh Kumar Goutam
Assistant Professor
Department of Computer Science
University of Lucknow

Introduction

- C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A..
- It was initially designed for programming in UNIX operating system.
- Now the software tool as well as the C compiler is written in C. Major parts of popular operating systems like Windows, UNIX, Linux is still written in C.

Features of C Programming Language:

- C is one of the most popular languages used today.
- C is a robust programming with an impressive set of built-in functions and a variety of operators which you can use to write any complex program.
- C programs are fast and efficient. This is because C uses a powerful set of data types and operators.
- C combines the power and capability of assembly language with the user friendly features of a high-level language.
- C is the most widely used older programming language. It continues to go strong while older programming languages such as BASIC and COBOL have been virtually forgotten.
- C is very much portable, which means programs written on a machine using C can be used on other machines as well without any modification.
- A C program consists of a number of functions that are supported by C library. In fact, you can create your own function, which can then be added to the C library.

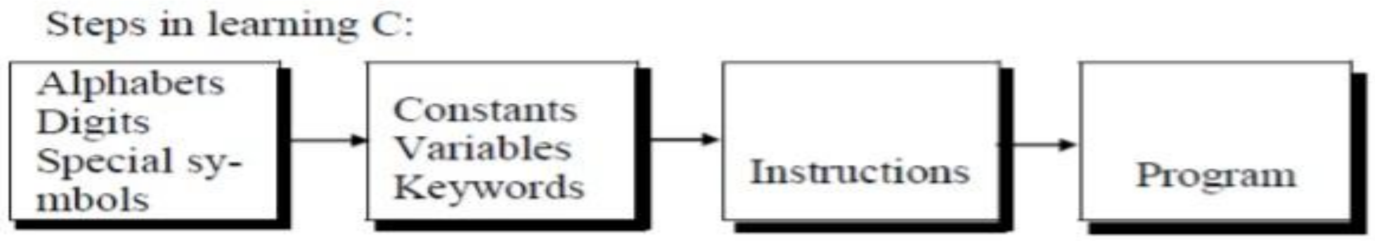
Characteristics of C Language:

- C is a General Purpose Programming Language. This means C can be used to write a variety of applications. It is often referred to as a “system programming language.”
- C is a middle level language, which means it combines the features of high level language with the functionality of an assembly language.
- C is a structured programming language, which means as a programmer, you are required to divide a problem into a several different modules or functions.
- C is renowned for its simplicity and is easy to use because of its structured approach. It has a vast collection of keywords, operators, built-in functions and data types which make it efficient and powerful.
- C is portable, which means a C program runs in different environments. C compilers are available for all operating systems and hardware platforms. Additionally, you can easily write code on one system and port it to another.
- C is popular not just because it can be used as a standalone programming language, but also as it can be used as an interface to other more visual languages.

Characteristics of C Language:

- C is a very flexible language; it is convenient and portable, like a high level language and flexible like a low level language. It can be interfaced with other programming languages.
- C is super fast. The compilation and execution of programs is much faster on C than with most other languages.
- C is modular, which means C programs can be divided into small modules, which are much easier to understand.
- C is easily available. The C software is easy to access and can be easily installed on your computer. The installation of C hardly takes a few minutes.
- C is easy to debug. The C compiler detects syntax errors quickly and easily and displays the errors along with the line numbers of the code and the error message.
- C makes available a number of in-built memory management functions that save memory and improve the efficiency of the program such as malloc(), calloc() and alloc().
- Recursion is one of the common techniques used in C, where in a function calls itself again and again.
- Finally, C has a rich set of library functions and supports graphic programming too.

Learning C is easier. Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an *instruction*. A group of instructions would be combined later on to form a *program.* a computer program is just a collection of the instructions necessary to solve a specific problem. The basic operations of a computer system form what is known as the computer's instruction set. And the approach or method that is used to solve the problem is known as an algorithm.



Languages

- In ***machine level language*** computer only understand digital numbers i.e. in the form of 0 and 1. So, instruction given to the computer is in the form binary digit, which is difficult to implement instruction in binary code
- The ***assembly language*** is on other hand modified version of machine level language. Where instructions are given in English like word as ADD, SUM, MOV etc. It is easy to write and understand but not understand by the machine.
- ***High level languages*** are machine independent, means it is portable. The language in this category is Pascal, Cobol, Fortran etc. High level languages are understood by the machine. So it need to translate by the translator into machine level. A translator is software which is used to translate high level language as well as low level language in to machine level language

Compiler and Interpreter

Compiler and ***interpreter*** are used to convert the high level language into machine level language. The program written in high level language is known as source program and the corresponding machine level language program is called as object program. Both compiler and interpreter perform the same task but there working is different. Compiler read the program at-a-time and searches the error and lists them. If the program is error free then it is converted into object program. When program size is large then compiler is preferred. Whereas interpreter read only one line of the source code and convert it to object code. If it check error, statement by statement and hence of take more time.

Comment and Preprocessor Directive

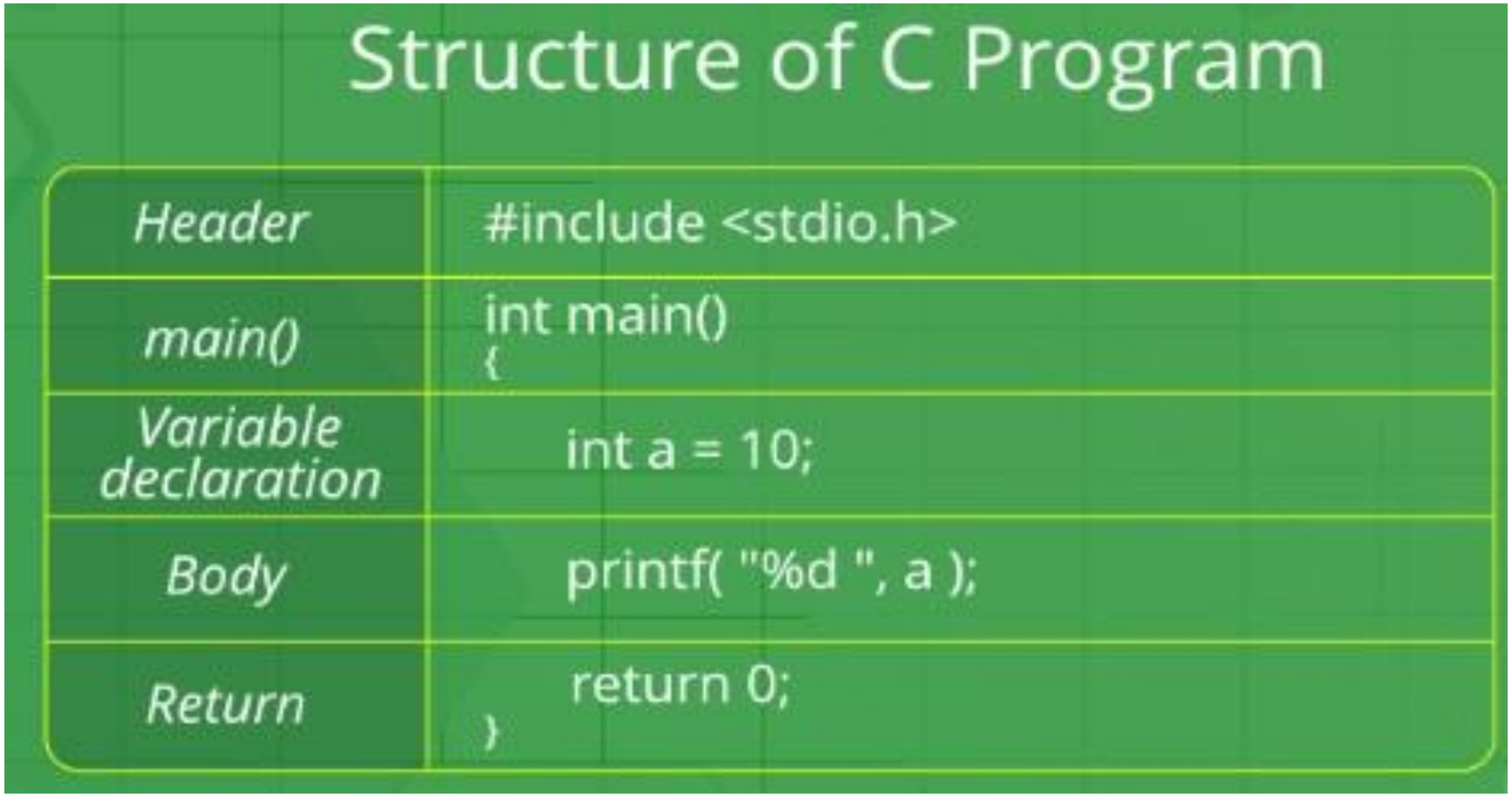
C Comments

It indicates the purpose of the program. It is represented as /*…………………………….*/ Comment line is used for increasing the readability of the program. It is useful in explaining the program and generally used for documentation. It is enclosed within the decimeters. Comment line can be single or multiple line but should not be nested. It can be anywhere in the program except inside string constant & character constant.

Preprocessor Directive:

#include<stdio.h> tells the compiler to include information about the standard input/output library. It is also used in symbolic constant such as #define PI 3.14(value). The stdio.h (standard input output header file) contains definition &declaration of system defined function such as printf( ), scanf( ), pow( ) etc. Generally printf() function used to display and scanf() function used to read value

Structure of C Program

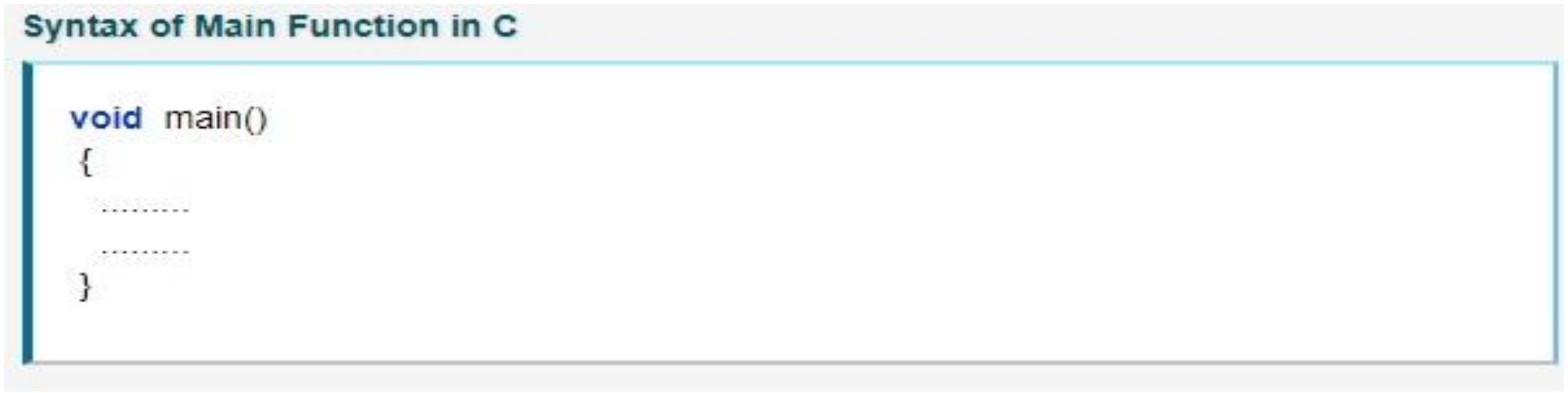


Structure of C Program

Header	#include <stdio.h>
main()	int main() {
Variable declaration	int a = 10;
Body	printf("%d ", a);

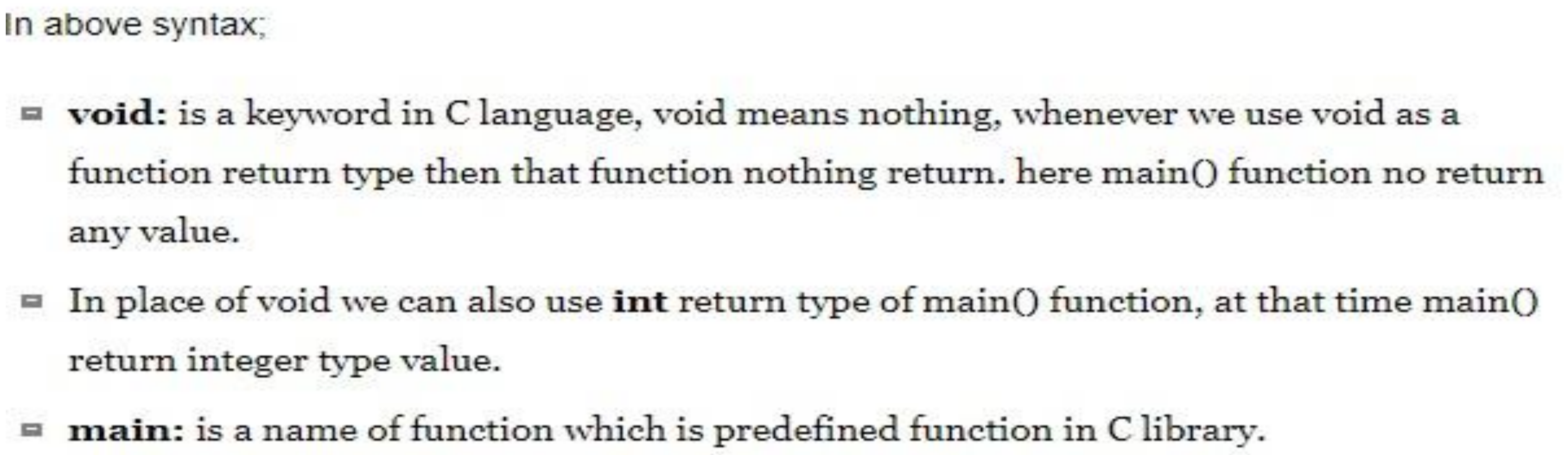
Return	return 0;
	}

Main function is the entry point of any C Program. It is the point from where the execution of program is started



Syntax of Main Function in C

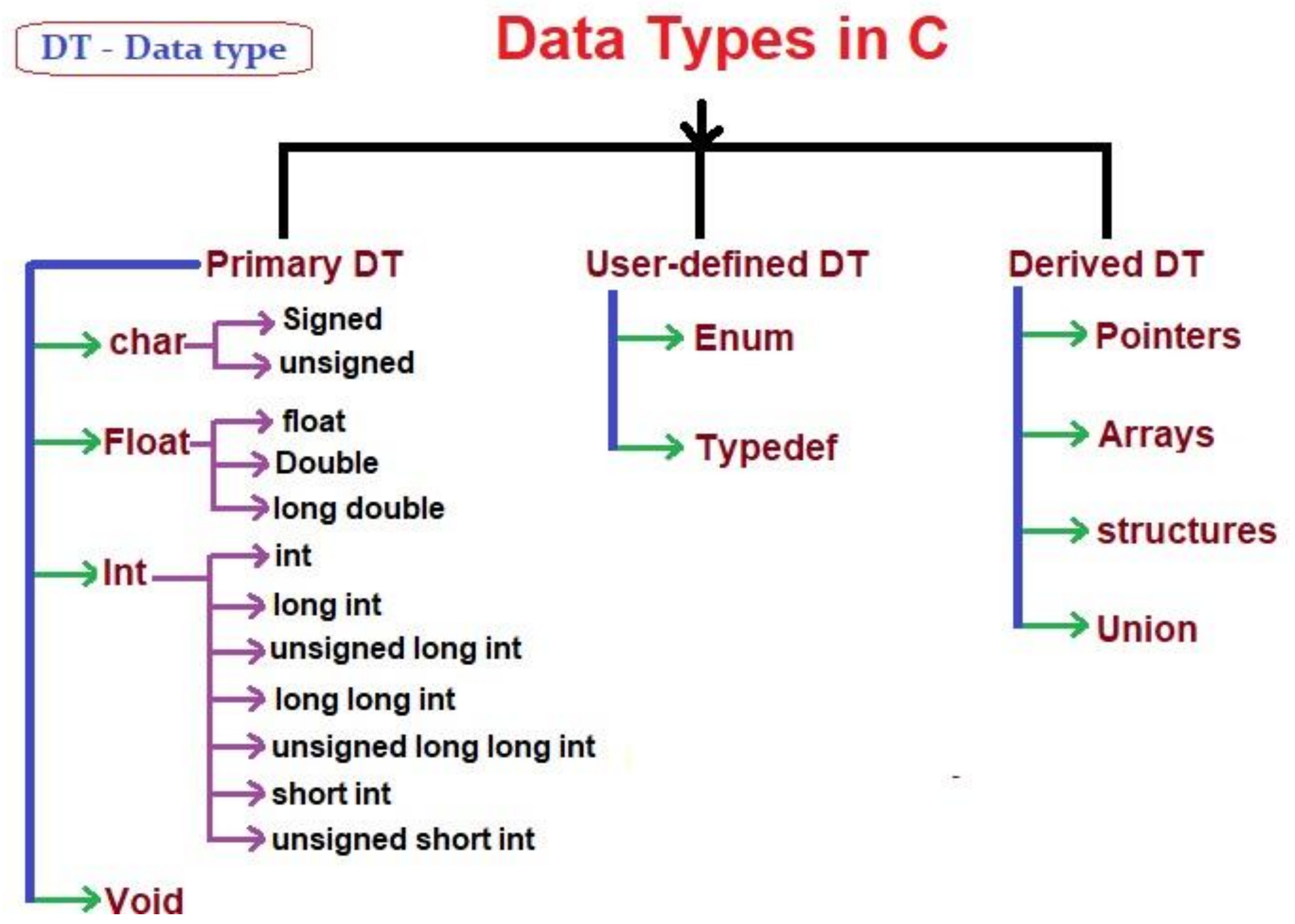
```
void  main()
{
  ..........
  ..........
}
```



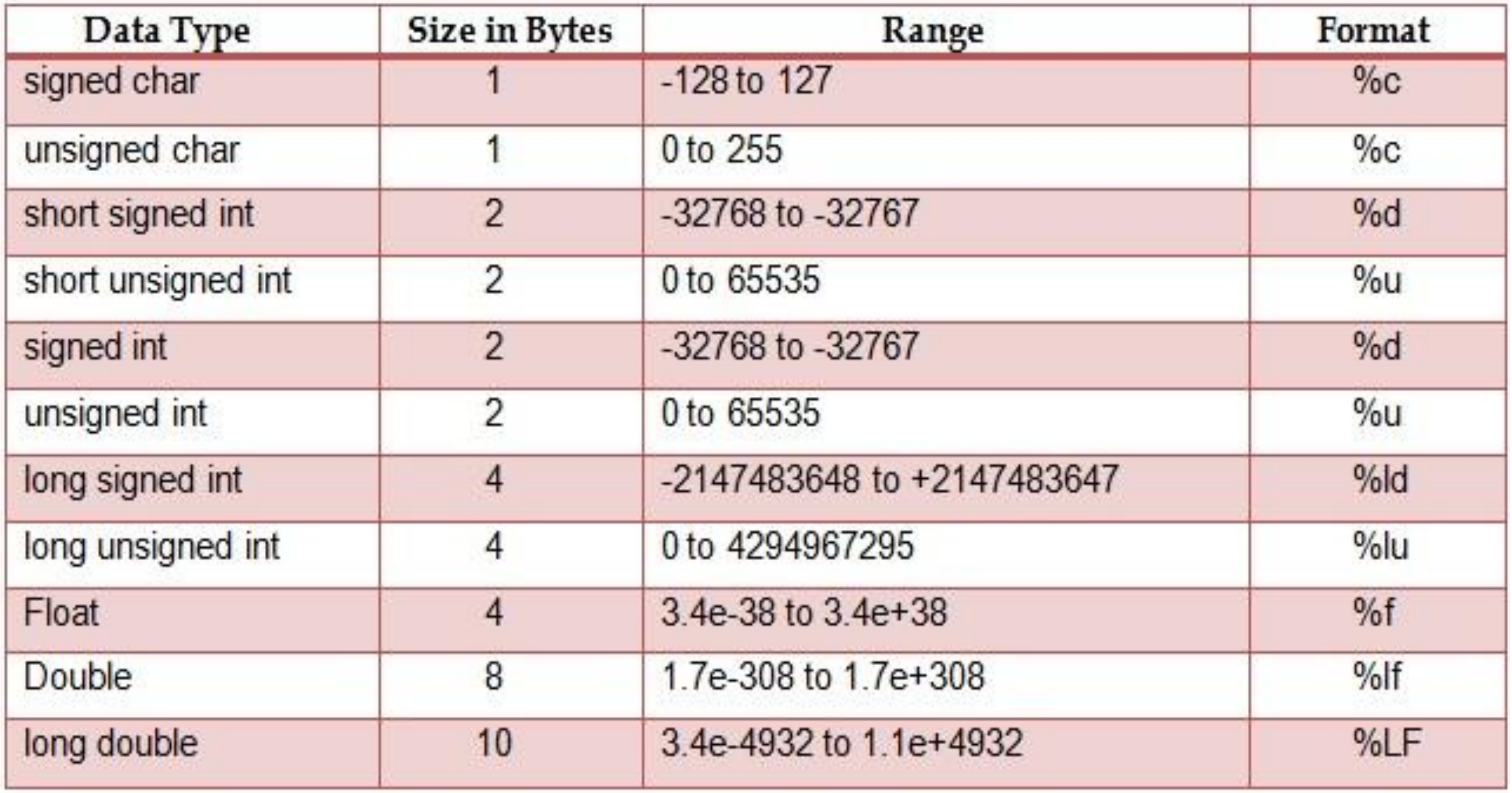
In above syntax;

- **void:** is a keyword in C language, void means nothing, whenever we use void as a function return type then that function nothing return. here main() function no return any value.
- In place of void we can also use **int** return type of main() function, at that time main() return integer type value.
- **main:** is a name of function which is predefined function in C library.

Data types in C

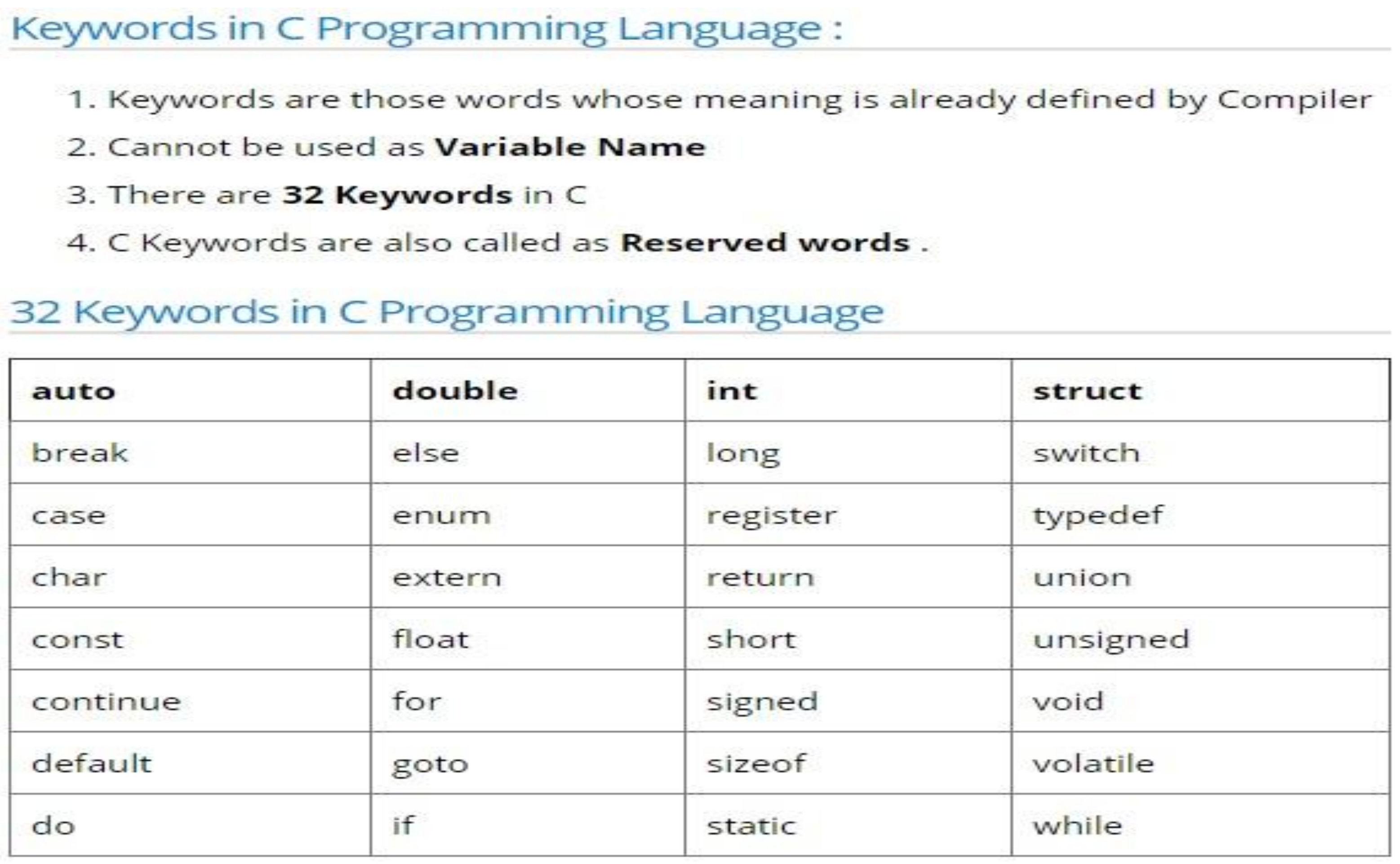


Data types Ranges



Data Type	Size in Bytes	Range	Format
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short signed int	2	-32768 to -32767	%d
short unsigned int	2	0 to 65535	%u
signed int	2	-32768 to -32767	%d
unsigned int	2	0 to 65535	%u
long signed int	4	-2147483648 to +2147483647	%ld
long unsigned int	4	0 to 4294967295	%lu
Float	4	3.4e-38 to 3.4e+38	%f
Double	8	1.7e-308 to 1.7e+308	%lf
long double	10	3.4e-4932 to 1.1e+4932	%LF

C Keywords



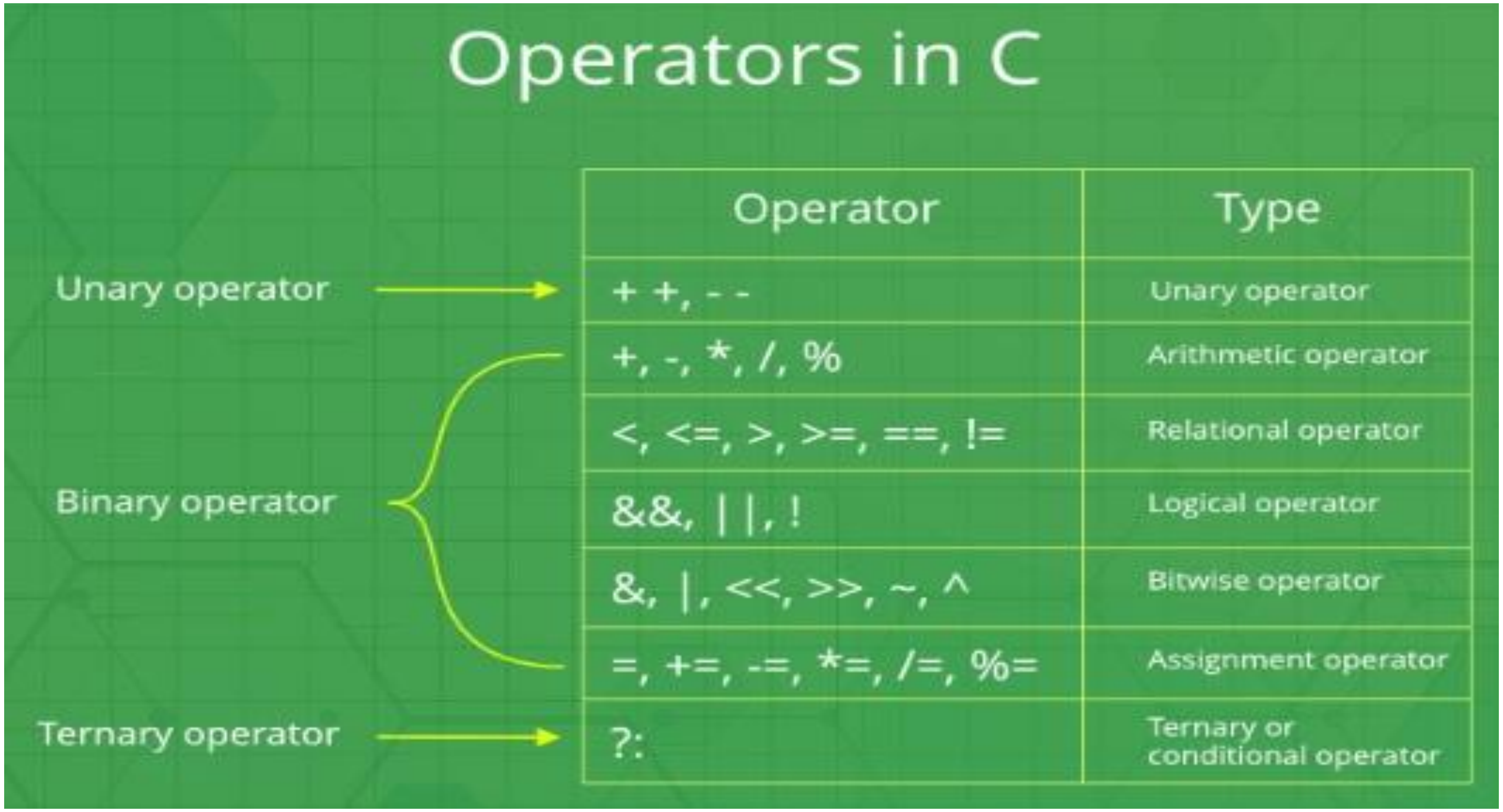
Keywords in C Programming Language :

1. Keywords are those words whose meaning is already defined by Compiler
2. Cannot be used as **Variable Name**
3. There are **32 Keywords** in C
4. C Keywords are also called as **Reserved words** .

32 Keywords in C Programming Language

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

Operators in C

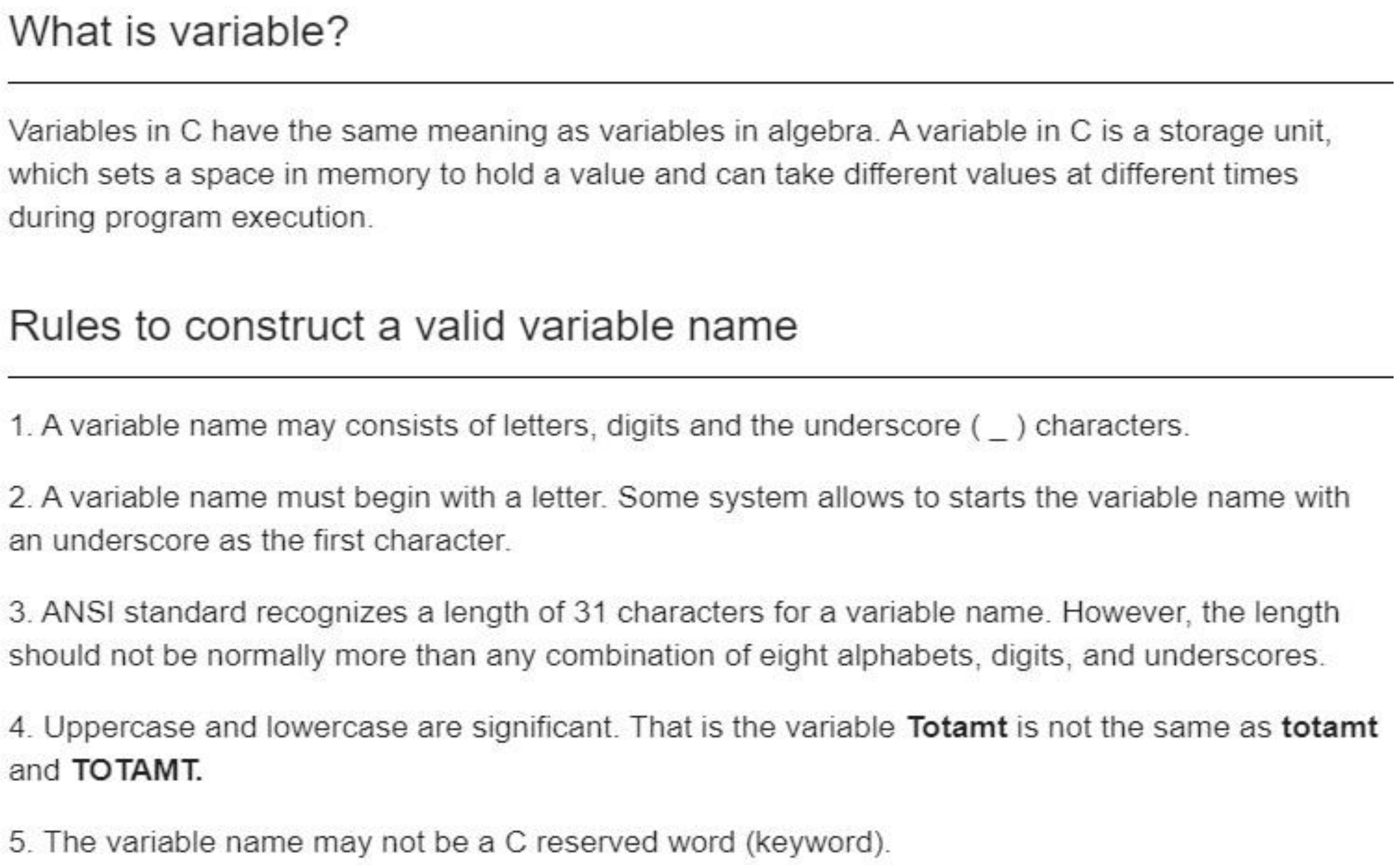


Operators in C

Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, ||, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

Unary operator
Binary operator
Ternary operator

C Variables



What is variable?

Variables in C have the same meaning as variables in algebra. A variable in C is a storage unit, which sets a space in memory to hold a value and can take different values at different times during program execution.

Rules to construct a valid variable name

1. A variable name may consists of letters, digits and the underscore (_) characters.

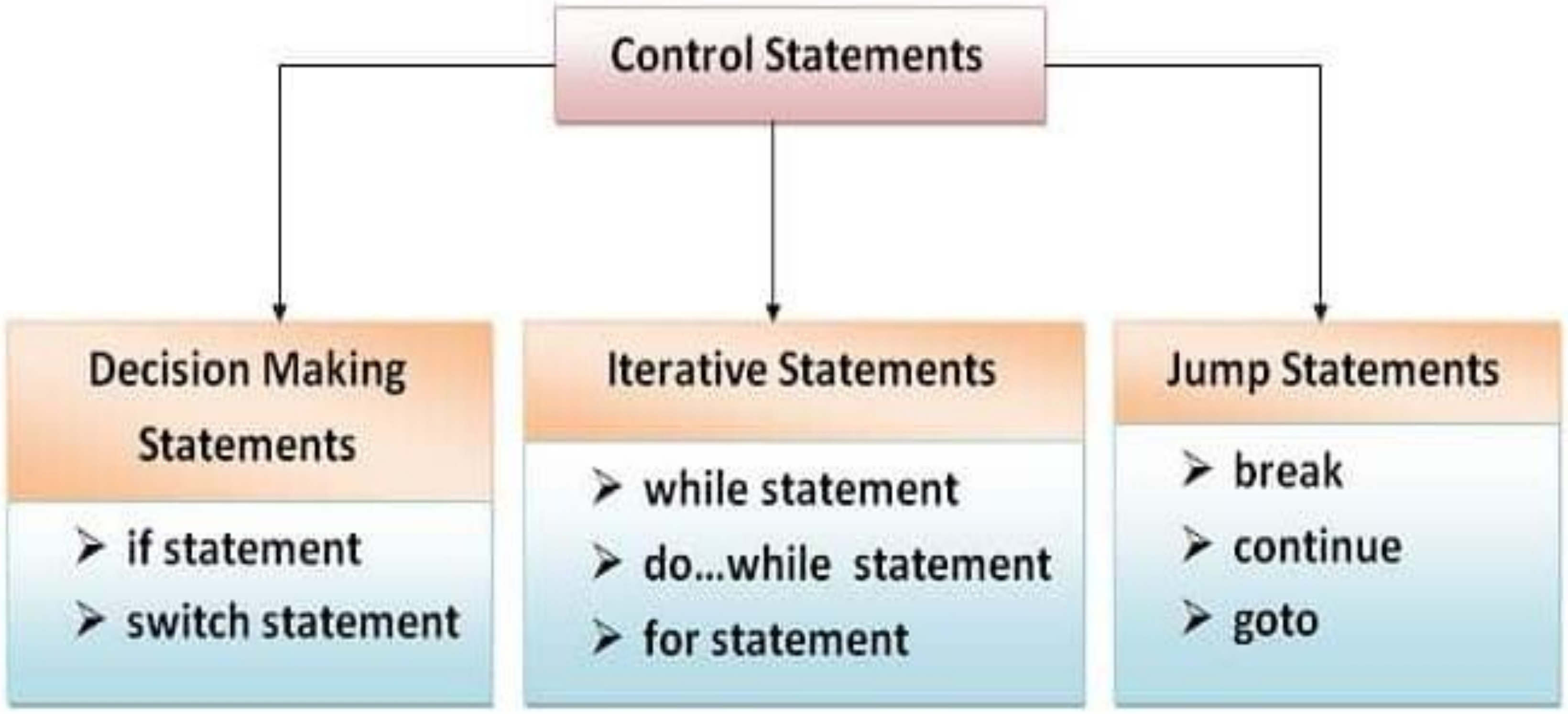
2. A variable name must begin with a letter. Some system allows to starts the variable name with an underscore as the first character.

3. ANSI standard recognizes a length of 31 characters for a variable name. However, the length should not be normally more than any combination of eight alphabets, digits, and underscores.

4. Uppercase and lowercase are significant. That is the variable **Totamt** is not the same as **totamt** and **TOTAMT.**

5. The variable name may not be a C reserved word (keyword).

Control Statements



If Statement

Syntax of if statement

```
if (condition)
{
    statements;
    ... ... ...
}
```

Flowchart of if statement

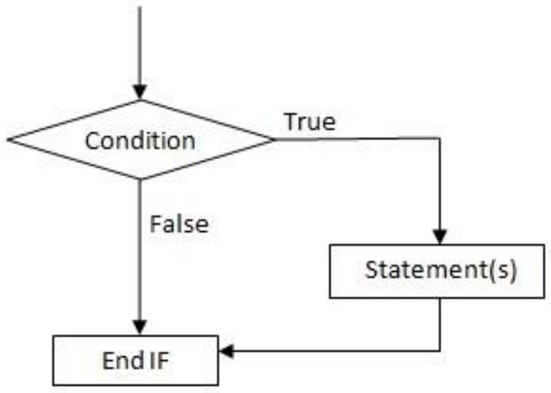


fig: Flowchart for if statement

If statement

Syntax of if statement

```
if (condition)
{
   statements;
   ... ... ...
}
```

Flowchart of if statement

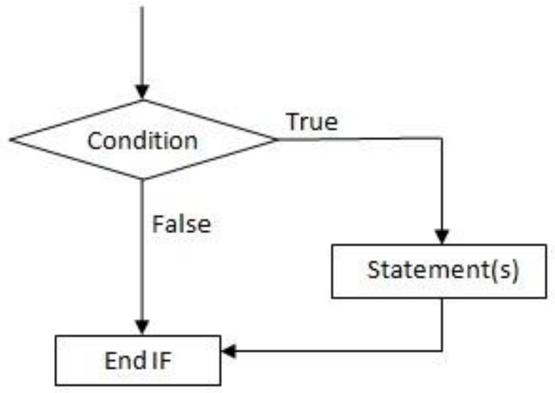


fig: Flowchart for if statement

```
#include<stdio.h>
int main()
{
   int n;
   printf("Enter a number:");
   scanf("%d",&n);
   if(n<10)
   {
      printf("%d is less than 10\n",n);
      printf("Square = %d\n",n*n);
   }
   return 0;
}
```

If else Statement

Syntax of if...else statement

```
if (condition)
{
   statements;
   ... ... ...
}
else
{
   statements;
   ... ... ...
}
```

Flowchart of if ... else statement

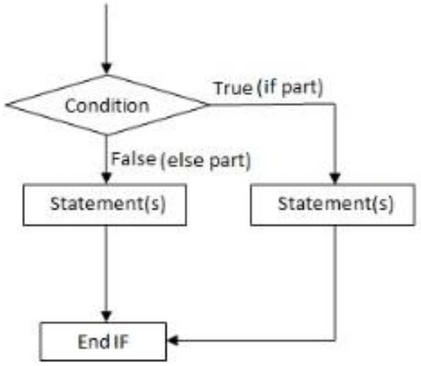
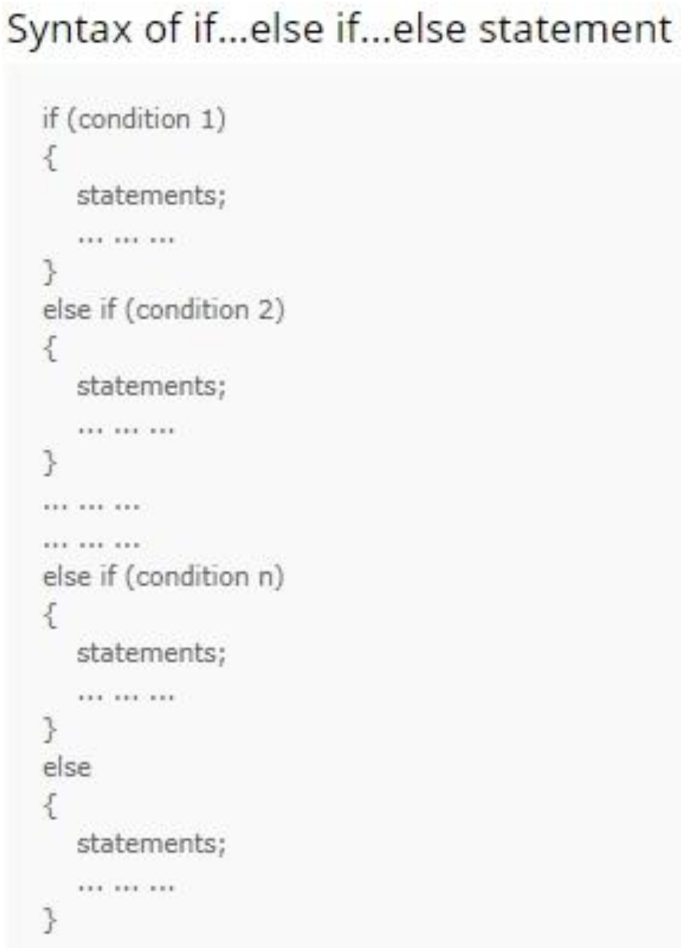


fig: Flowchart for if ... else statement

```
#include<stdio.h>
int main()
{
   int n;
   printf("Enter a number:");
   scanf("%d",&n);
   if(n%2 == 0)
      printf("%d is even",n);
   else
      printf("%d is odd",n);
   return 0;
}
```

If-else if Statement

Syntax of if...else if...else statement



```
if (condition 1)
{
   statements;
   ... ... ...
}
else if (condition 2)
{
   statements;
   ... ... ...
}
... ... ...
... ... ...
else if (condition n)
{
   statements;
   ... ... ...
}
else
{
   statements;
   ... ... ...
}
```

Flowchart of if ... else if ... else statement

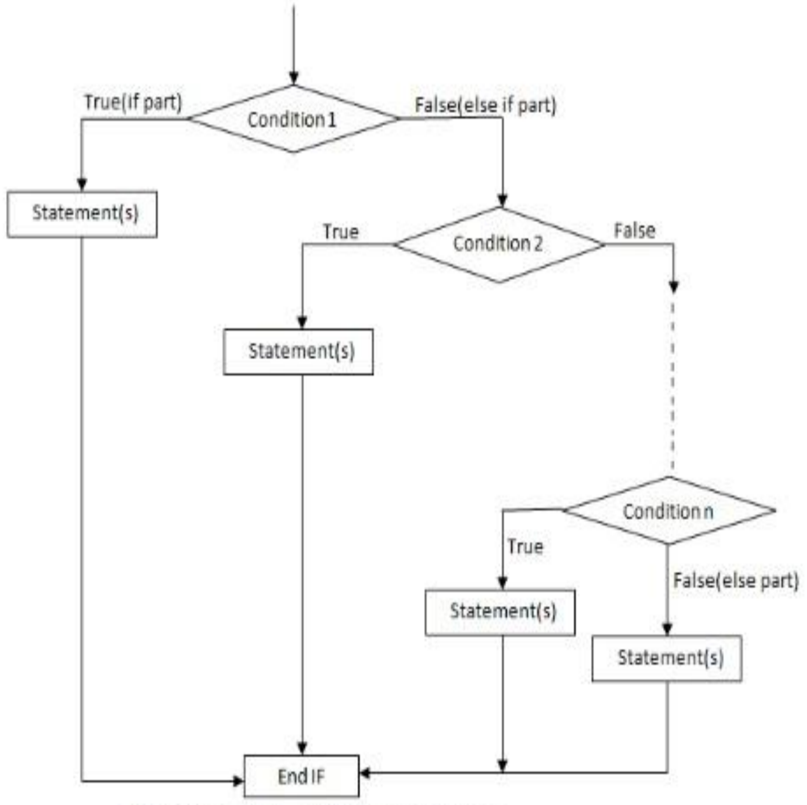
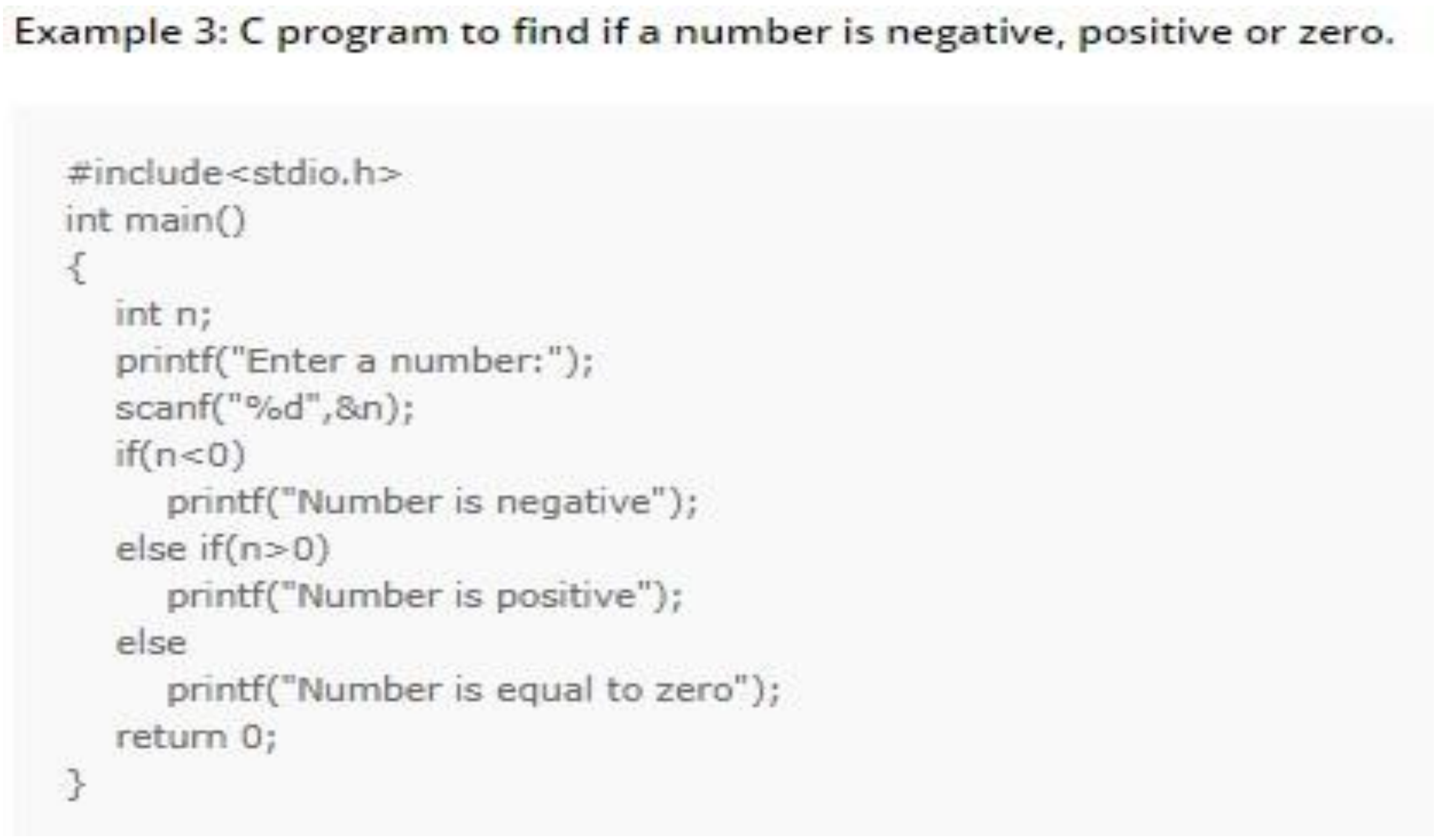


fig: Flowchart for if ... else if ... else statement

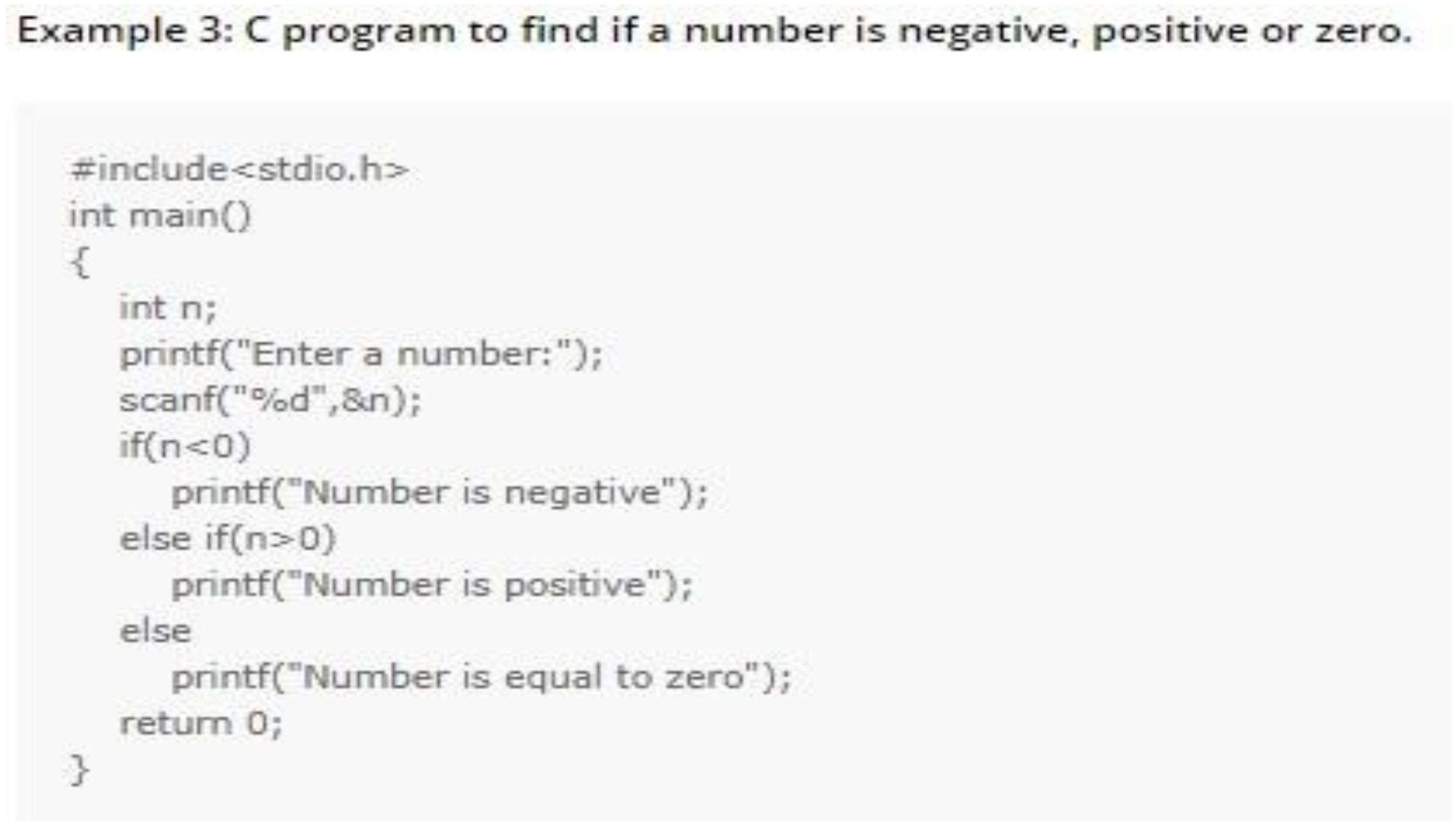
Example of if ... else if ... else statement



Example 3: C program to find if a number is negative, positive or zero.

```
#include<stdio.h>
int main()
{
    int n;
    printf("Enter a number:");
    scanf("%d",&n);
    if(n<0)
        printf("Number is negative");
    else if(n>0)
        printf("Number is positive");
    else
        printf("Number is equal to zero");
    return 0;
}
```

IF Statement

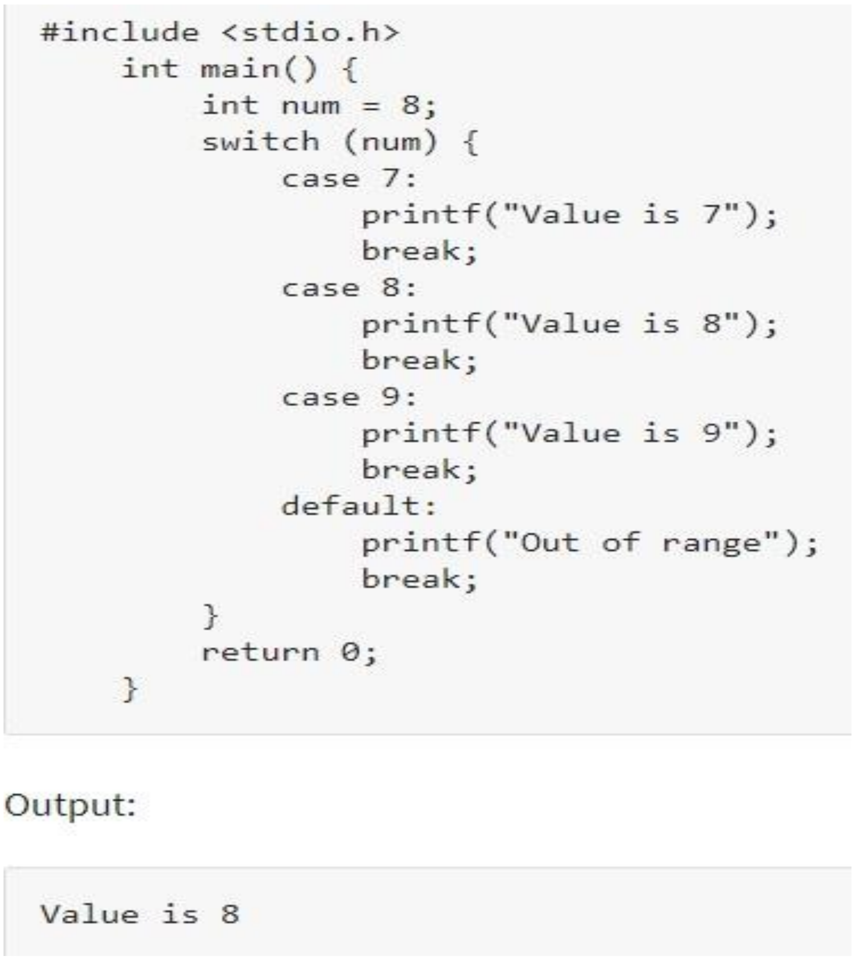


Example 3: C program to find if a number is negative, positive or zero.

```
#include<stdio.h>
int main()
{
    int n;
    printf("Enter a number:");
    scanf("%d",&n);
    if(n<0)
        printf("Number is negative");
    else if(n>0)
        printf("Number is positive");
    else
        printf("Number is equal to zero");
    return 0;
}
```

Switch Statement

A switch statement tests the value of a variable and compares it with multiple cases. Once the case match is found, a block of statements associated with that particular case is executed. The default case is an optional one. Whenever the value of test-expression is not matched with any of the cases inside the switch, then the default will be executed. Otherwise, it is not necessary to write default in the switch.



```
#include <stdio.h>
    int main() {
        int num = 8;
        switch (num) {
            case 7:
                printf("Value is 7");
                break;
            case 8:
                printf("Value is 8");
                break;
            case 9:
                printf("Value is 9");
                break;
            default:
                printf("Out of range");
                break;
        }
        return 0;
    }
```

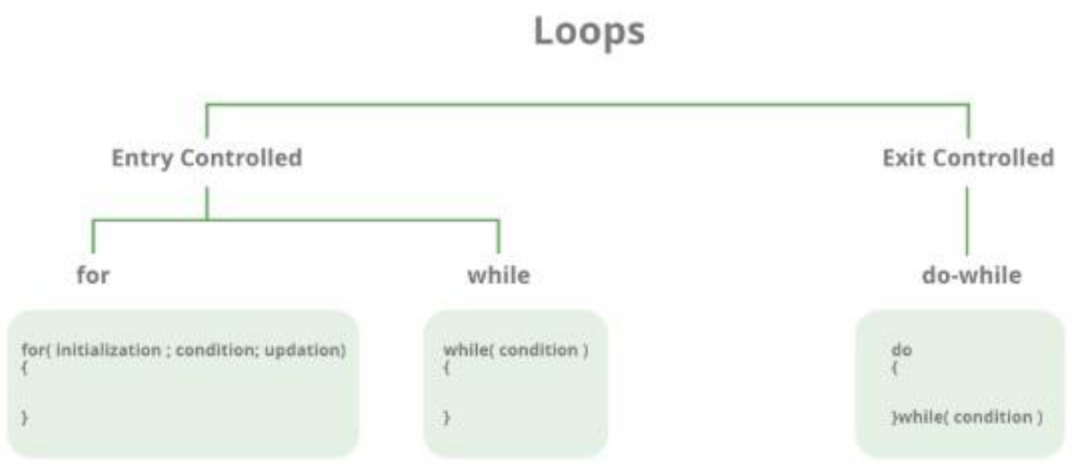
Output:

```
Value is 8
```

Looping Statements

In computer programming, a loop is a sequence of instructions that is repeated until a certain condition is reached.
There are mainly two types of loops:

Entry Controlled loops: In this type of loops the test condition is tested before entering the loop body. Loop and While Loops are entry controlled loops.
Exit Controlled Loops: In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. do – while loop is exit controlled loop.

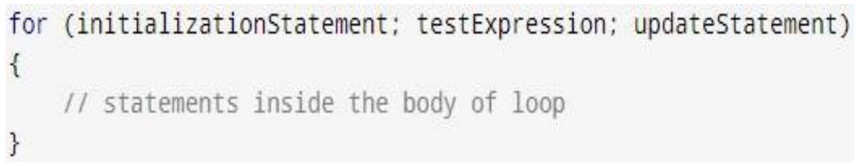


```
for( initialization ; condition; updation)
{

}
```

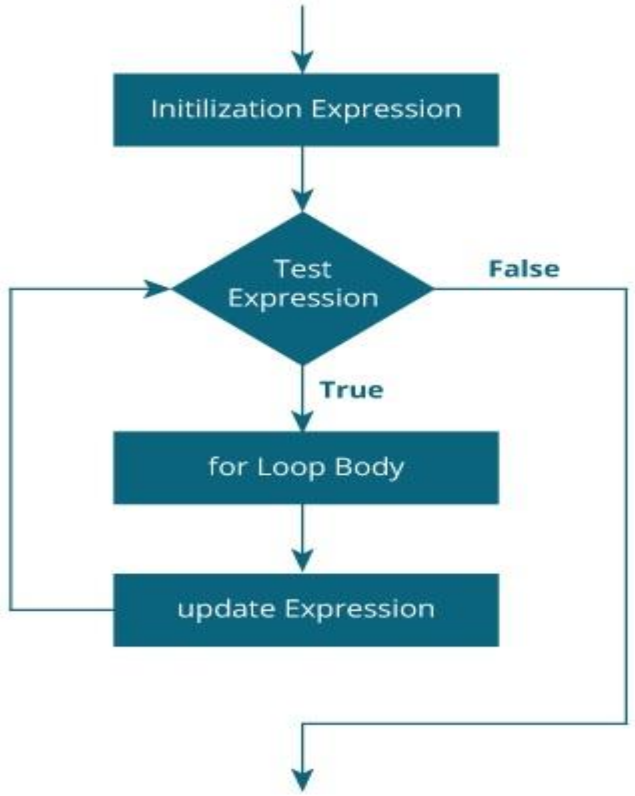
```
while( condition )
{

}
```

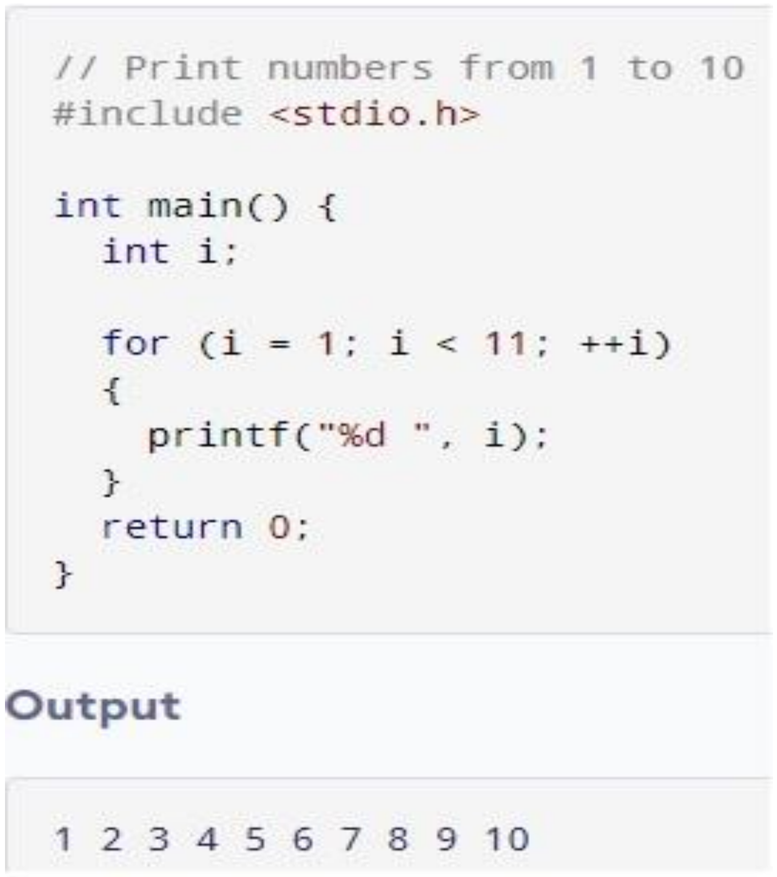
```
do
{

}while( condition )
```

For Loop Syntax and Example



```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}
```





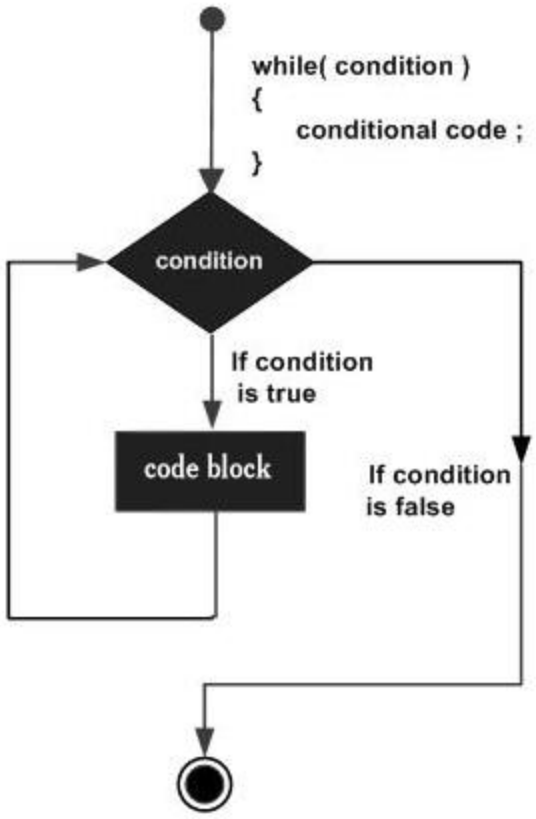
```
// Print numbers from 1 to 10
#include <stdio.h>

int main() {
  int i;

  for (i = 1; i < 11; ++i)
  {
    printf("%d ", i);
  }
  return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

While Loop Syntax and Example

```
while(condition) {
   statement(s);
}
```



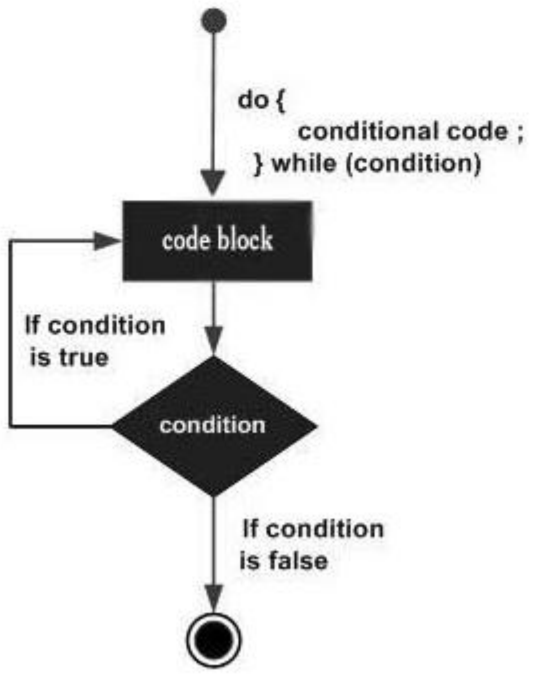
```
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

When the above code is compiled and executed,

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Do…..While Loop

```
do {
   statement(s);
} while( condition );
```



```
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* do loop execution */
   do {
      printf("value of a: %d\n", a);
      a = a + 1;
   }while( a < 20 );

   return 0;
}
```

When the above code is compiled and executed,

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Thanks

Dear Students

If you have queries, Please feel free to contact me at

e-Mail: rajeshgoutam82@gmail.com

Mobile No: 9453838526