C PROGRAMMING

Intro to C Programming

Who - Dennis M. Ritchie

When - 1972

Why - To develop UNIX operating system

How it works - Editor > Compiler > Execution > Output

What we are going to learn -

About C Programming

- **Procedural Language** - Instructions in a C program are executed step by step.

- **Portable** - You can move C programs from one platform to another, and run it without any or minimal changes.

- **Speed** - C programming is faster than most programming languages like Java, Python, etc.

- **General Purpose** - C programming can be used to develop operating systems, embedded systems, databases, and so on.

Why learn C

•C helps you to understand the internal architecture of a computer, how computer stores and retrieves information.

•After learning C, it will be much easier to learn other programming languages like Java, Python, etc.

•Opportunity to work on open source projects. Some of the largest open-source projects such as Linux kernel, Python interpreter, SQL database, etc. are written in C programming.

Advantages of C

- Easy to learn
- Structured language
- It produces efficient programs.
- It can handle low-level activities.
- It can be compiled on a variety of computer platforms.

Rules for writing C program

- Each instruction is terminated by a semicolon (;).
- Case sensitive language – keywords are written in lowercase.
- Execution of program starts from main() function.

First C program

```
#include <stdio.h>
 void main()
{
   printf(“welcome");
}
```

C program structure

- The first line of the program *#include <stdio.h>* is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
- The next line *void main()* is the main function where program execution begins.
- The next line *printf(...)* is another function available in C which causes the message “welcome” to be displayed on the screen.

Turbo C programming Commands

F2	-	Save
Alt + F9	-	Compile
Ctrl + F9	-	Run / Execute
Alt + F5	-	See output
Alt + F3	-	Close
Alt + X	-	Quit / Exit
F5	-	Maximize / Restore

Keywords

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

Identifiers

- A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter “A to Z” or “a to z” or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

Valid Identifiers

- abc
- move_name
- a_123
- myname50
- _temp
- j
- a23b9
- retVal

Invalid Identifiers

- My name
- 123abc
- Name@company
- $x
- Last-name

White space character

- Whitespace is the term used in C to describe blanks, tabs and newline characters. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement ends and the next element begins.

Variables

- a variable is a container (storage area) to hold data.

- To indicate the storage area, each variable should be given a unique name (identifier). Variable names are just the symbolic representation of a memory location. For example:

```
int playerscore = 95;
```

Rules for naming variables

A variable name can have letters (both uppercase and lowercase letters), digits and underscore.

The first letter of a variable should be either a letter or an underscore.

There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if the variable name is longer than 31 characters.

Constants

- If you want to define a variable whose value cannot be changed, you can use the keyword. This will create a constant. For example

```
Pi = 3.14
Pi = 2.1  //Error
```

Data types

- **int**

Integers are whole numbers that can have both zero, positive and negative values but no decimal values. For example : 0, -5 , 10

- **float**

float is used to hold real number. For example : 0.5, 9.8, 4.6

- **char**

char is used for declaring character type variables. For example; ‘raju’, ‘Mayur’, ‘SIVA’

- **void**

void is an incomplete type. It means "nothing" or "no type". You can think of void as **absent**.

Data types

Type	Size (bytes)	Format Specifier
int	at least 2, usually 4	%d, %i
char	1	%c
float	4	%f
double	8	%lf
short int	2 usually	%hd
long int	at least 4, usually 8	%ld, %li
long long int	at least 8	%lld, %lli
long double	at least 10, usually 12 or 16	%Lf

Output

In C programming, printf() is one of the main output function.

The printf() function sends formatted output to the screen.

For example,

```
    printf("C Programming");
```

Input

In C programming, scanf() is one of the commonly used function to take input from the user.
The scanf() function reads formatted input from the standard input such as keyboards.

For Example

```
int X;
printf("Enter an integer: ");
scanf("%d", &X);
printf("Number = %d",X);
```

Operators

- Arithmetic Operators
- Increment and Decrement Operators
- Assignment Operators
- Relational Operators
- Logical Operators

Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after divisions

Arithmetic Operators
+, -, *, /, %

```
// Working of arithmetic operators


    int a = 9,b = 4, c;

    c = a+b;
    printf("a+b = %d \n",c);
```

Increment and Decrement Operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1.

These two operators are unary operators, meaning they only operate on a single operand.

Increment and Decrement Operators
+ +, - -

```
// Working of increment and decrement operators

    int a = 10, b = 100;

    printf("++a = %d \n", ++a);
    printf("--b = %d \n", --b);
```

Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

Assignment Operators
=, +=, -=, *=, /=, %=

```
// Working of assignment operators

    int a = 5, c;

    c = a;      // c is 5
    printf("c = %d\n", c);

    c += a;     // c is 10
    printf("c = %d\n", c);
```

Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

Relational Operators

= =, <, >, !=, <=, >=

```
// Working of relational operators

    int a = 5, b = 5, c = 10;

    printf("%d == %d is %d \n", a, b, a == b);
    printf("%d == %d is %d \n", a, c, a == c);
```

Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression ((c==5) && (d>5)) equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression ((c==5) || (d>5)) equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression !(c==5) equals to 0.

Logical Operators
&&, ||, !

```
// Working of logical operators

    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);
```