C - PROGRAM

A C program basically consists of the following parts —

- Preprocessor Commands
- Functions
- Variables
- Statements and expressions
- Comments

```
/* Hello world Program */
#include <stdio.h>
int main(){
/* first program */
printf("Hello world!");
return 0;
}
```

output
Hello world

◇ Let us take a look at the various parts of above 'Hello world' program —

⇨ The first Line of the program #include <stdio.h> is a pre-processor command which tells a compiler to include stdio.h file before going to actual compilation.

⇨ The next Line int main() is the main function where the program executing begins.

⇨ The next Line /*--------*/ will be ignored by the compiler. So such lines are called comments in the program.

⇨ The next Line printf(----) is a another function which causes the message "Hello world" to be displayed on screen.

⇨ The next Line return 0; terminates the main function and return the value 0.

DATA TYPE

◇ Data type refers to an extensive system used for declaring Variables or functions of different types.

◇ The type of a variable determines how much space it occupies in storage.

◇ Different data types have different ranges to store

◇ Basically data types are divide into 3 different type —

- Primary Data type [Int, char, float]
- User defined Data type [Enum, Typedef]
- Derived Data type [Pointers, Arrays, Structure, union]

Primary Data type

☞ Primary data types are also known as fundamental data types because they are pre-defined in C Language.

☞ Primary Data Type in C are basically 4 types:
int, char, float and double.

	Data Type	Memory (bytes)	Range	Format Specifier
1	int	4	-2^{31} to $2^{31}-1$	%d
2	char	1	-128 to 127	%c
3	float	4	1.2E-38 to 3.4E+38	%f
4	double	8	2.3E-308 to 1.7E+308	%lf

Integer Data Type

☞ The int data type is used to store integer values.

- It can be signed or un-signed
- It has generally 32 bits (4 bytes)
- By default integer is signed

Range
signed int - -2^{N-1} to $2^{N-1}-1$
un-signed int - 0 to 2^N-1

Character data type

☞ The char data type is used to store the characters values.

- characters must be inside single quotes
- It has usually 1 byte ; Range:- -2^{N-1} to $2^{N-1}-1$
- Whenever we enter a character type variable, the characters is used to stored as integer value in the address Location.

Float Data Type

☞ The float data type is used to store the floating point numbers.

- The numbers that have a fractional part are called floating point numbers.
- It has generally 4 bytes.
- These can represent a much larger and wider range of digits as compared to integer data type

Range: 1.2E-38 to 3.4E+38

Double Data Type

☞ The double data types is used to store floating numbers.

- It occupies twice as much memory as float data type.
- It has generally 8 bytes

Range: 2.3E-308 to 1.7E+308

Example

int ⇒ 10, 20, 30, 100, 1000, 586 etc
char ⇒ 0.7, 2.32, 8.21, 6.432 etc
float ⇒ 'A', 'B', 'a', 'b', 'x', 'z' etc
double ⇒ 92.334000, 194.46983300 etc

Conversion specifier ⇒ The conversion specifier characters specifies whether to interpret the corresponding argument as a character, a string, a pointer, an integer or a floating number.

%d	signed int / int	%ld	Long int	%p	Pointer
%u	unsigned int	%lld	Long Long int	%s	String
%hi	Short signed int	%lu	unsigned Long int	%x	Hexadecimal
%hu	Short unsigned int	%llu	unsigned Long Long int	%o	Octa Integer

Variable

Variable ⇒ A variable is a name given to storage area that our program can manipulate. / A variable is basically a name given by a memory location where we can store any type of data.

- Two main concepts of Variable
  - Declaration
  - Initialization and usage

Variable Declaration ⇒ when we create a variable that time the process is called as variable declaration.

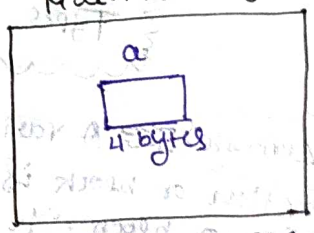
Syntax

datatype variable name;

ex

```
int number;
```

Main Memory



Variable Initialization

Syntax

* after declare variable

** Variable name = value;

↓ assignment operator

[Help us to assign or store the value]

ex

```
int number;
number = 10;
```

How to Print the Value of a Variable

if you want to print or display the value of a variable in c then use printf function.

Syntax

Printf ("format sp", var name);

Program to display var value in c

```
#include<stdio.h>
int main()
{
  int number = 10;
  Printf("%d", number);
}
```

How to write a additional Masage in console with the value

```
Printf("The number is %d", number);
```

Rules to create a Variable name ✓

(1) A variable can have alphabets, digits, and underscore.

(2) A variable name can start with the alphabet and underscore only. It can't start with a digit

(3) No whitespace is allowed within the variable name

(4) A variable name must not be any reserved word or keyword

Ex — _Srujan, Ayon_Poojari, Rimc6213, Aryan_812

Don't declare Variable name

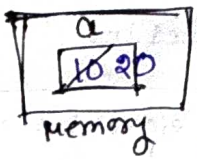
(1) Ayon Poojari [it contains wide space in between Ayon and Poojari]

(2) 13Srujan [It is starting with a number so we can't declare it as a Variable]

(3) Void, char, int [we can't declare them as Variables because they have already assigned some functions in the c program Library]

Variable Value is changing

```
int a = 10;
int a = 20;
Printf("%d", a);
Printf("%d", a);
```



Hence Value is changing.

Types of Variable

(1) Local Variable ☞ A variable that is declared and used inside the function or block is called a local Variable. It is scope is limited to function or block. It can't be used outside the block. Local variable need to be initialized before use.

```
# Void function() {
    int x = 10;
}
```

(2) Global Variable A variable that is declared outside the function or block is called a global Variable.

```
#include<stdio.h>
int z = 10;                    // global Variable
Void function() {
    int x = 20;                // Local Variable
    static int y = 30;         // static Variable
}                              // static keyword
int main() {
    function();
    return 0;
}
```

(3) Static Variable A variable that retains its value between multiple functions call is known as a static Variable.

Keywords

These are Reserved words whose meaning is already known to compiler.
There are total 32 keywords available in C.

1 auto	6 continue	11 return	16 static	21 float	26 switch	31 Volatile	
2 break	7 default	12 register	17 int	22 for	27 typedef	32 while	
3 case	8 do	13 short	18 else	23 goto	28 union		
4 char	9 double	14 signed	19 enum	24 if	29 unsigned		
5 const	10 long	15 sizeof	20 extern	25 struct	30 void		

Operators

An operator is a symbol that tells the compiler to perform specific mathematical and logical operation.

ex: 5 (+) 10 = 15 — (+) is the operator; 5 and 10 are the operands

ARITHMETIC OPERATOR

operators	Description
+	Adds two operands
-	Substracts second operands from first
*	Multiplies both operands
/	Divides
%	Modulus operator and remainder
++	Increment operator
--	Decrement operator

Example

```
int main() {
    int a=10;
    int b = 4;
    Printf("Sum=%.d\n", a+b);
    Printf("Sub=%.d\n", a-b);
    Printf("Product=%.d\n", a*b);
    Printf("Divide=%.d\n", a/b);
    Printf("Rem=%.d\n", a%b);
    Printf("a=%.d\n", ++a);
    return 0;
}
```

RELATIONAL OPERATOR

op	Description
==	checks if the values of two operands are equal or not. If yes then the condition becomes true.
!=	checks if the values of two operands are equal or not. If no, then the condition becomes true.
>	The value of Left operand is greater than the value of right operand
<	Left operand is less than the value of right operand
>=	Value of Left operand is greater than or equal to right operand
<=	Value of left operand is less than or equal to right operand

Example

```
int main() {
    int a = 10;
    int b = 4;
    Printf("a>b=%.d\n", a>b);
    Printf("a<b=%.d\n", a<b);
    Printf("a>=b=%.d\n", a>=b);
    Printf("a<=b=%.d\n", a<=b);
    Printf("a==b=%.d\n", a==b);
    return 0;
}
```

Logical operator

operator	Description
&&	Logical AND ⇒ If both the operands are 0 [True] Then the condition becomes true
||	Logical OR ⇒ If any of the two operands is none zero then it becomes true.
!	Logical NOT ⇒ It is used to reverse the Logical state of its operand.

NOTE // True → 1 and False → 0

- short circuit incase of && → if there is a condition anywhere in expression that return false, then rest of the condition after that will not evaluated
- short circuit incase of || → if there is a condition anywhere in expression that return True. Then rest of the condition after that will not be evaluated.

Bitwise operator

op	Description
&	Binary AND = it copies a bit to the result if it exists in both operands.
|	Binary OR = It copies a bit if it exists in either operands
^	Binary XOR = It copies the bit if it is set in only one operand.
~	Binary one's complement = It has the effect of flipping bits
<<	Binary Left Shift = The value is moved Left by the number of bits specified by right operand
>>	Binary Right Shift = The Left operand value is moved right by the no of bits specified

Truth Table

P	Q	P&Q	P|Q	P^Q
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Miscellaneous operator

sizeof() ⇒ Return the size of the variable

```
int a = 10;
Printf("size is %d", sizeof(a));
```

& ⇒ address of operator Returns the address of variable

```
int a = 10;
Printf("address is %d", &a);
```

* ⇒ Use to declare a Pointer Variable

```
int a = 10;
int *ptr;
```

?: ⇒ conditional expression or Ternary operator

Size of operator [sizeof()]

⇨ sizeof() operator computes the size of variable (bytes)

- △ it is a unary operator
- ◇ for float values use like this (ex) 10.2f otherwise it will be treated as double (10.2).

TYPE CASTING

It refers to changing an variable of one datatype into another.

⇨ the compiler will automatically change one type of data into another if it make sense.

Types

IMPLICIT TYPE

when the type conversion is performed automatically by the compiler without programmers intervention such type of conversion is known as implicit type conversion.

ex

```
int x = 10;
char y = 'a';
int z = x + y;
printf("%d", z);
```

EXPLICIT TYPE

The type conversion performed by the programmer by passing the data type of expression of specific type is known as explicit type conversion.

ex

```
double x = 1.2;
int sum = (int)x + 1
printf("sum = %d", sum)
```

Syntax

(type) expression

INPUT AND OUTPUT

INPUT it means to feed some data into a program.

- an input can be given in the form of a file as from the command line
- output, it means to display some data on screen, printf or in any file.

ex:

```
#include<stdio.h>
int main(){
    int a;
    printf("Enter a number: ");
    scanf("%d",&a);
    printf("The number is: %d", a);
    return 0;
}
```

scanf() function
→ use to take input form user

printf() function
→ use to show the output