Chapter 1 Fundamentals of C programming.

1.1-character set, identifiers and keywords, Data types, constants, Variables.

Character Set- the character set refers to a set of all the valid characters that we can use in the source program for forming constants, variables and keywords.

Alphabets	A, B, ….., Y, Z a, b, ……, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /

Identifiers- identifiers represent the name in the C program.

Example-

int a = 90;

char section = 'A';

Rules for naming identifiers are:
1) name should only consist of alphabets (both upper and lower case), digits and underscore (_) sign.
2) first characters should be alphabet or underscore.
3) name should not be a keyword.
4) since C is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc. are different identifiers.

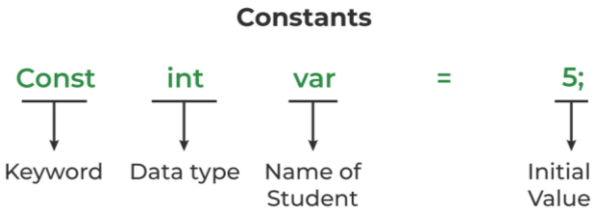
Keyword- There are certain words reserved for doing specific task, these words are known as reserved word or keywords.

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

Data types- A data type specifies the type of data that a variable can store such as integer, floating, character, etc.

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Constants- The constants are variables whose values cannot be modified once they are declared in the C program.



Example-

1. const int int_const = 25;
2. const char ch = 'A';

Variables- A variable in C is a memory location with some name that helps store some form of data.

Structure of C program-

Comment line
Preprocessor directive
Global variable declaration
main function( )
{
Local variables;
Statements;
}

Example-

```
/* message display program*/          (labeling)
#include <stdio.h>                    (library / header file)
int a=90;                             (global variable)
int main()                            (indicates start of program)
{
  Instructions
  Return 0;                           (end of program)
}
```

1.2- Operators- Arithmetic, Relational(comparator), Logical, Assignment, Unary, Conditional, Bitwise, Comma and others. Expression, statements, library functions, preprocessor.

Expression- Expressions are the combination of variables, operands, and operators.

Example- C= a + b;

Statements- programming instructions are called statements**.**

Example- `printf("Hello World!");`

Library Functions-

<math.h>	The math.h header defines various mathematical functions and one macro. All the Functions in this library take double as an argument and return double as the result.
<stdio.h>	The stdio.h header defines three variable types, several macros, and various function for performing input and output.
<time.h>	Defines date and time handling functions.
<string.h>	Strings are defined as an array of characters. The difference between a character array and a string is that a string is terminated with a special character '\0'.

1. Arithmetic Operators-

Operator	Name of the Operator	Arithmetic Operation	Syntax
+	Addition	Add two operands.	x + y
–	Subtraction	Subtract the second operand from the first operand.	x – y
*	Multiplication	Multiply two operands.	x * y
/	Division	Divide the first operand by the second operand.	x / y
%	Modulus	Calculate the remainder when the first operand is divided by the second operand.	x % y

Program-

```
#include <stdio.h>
 int main()
{
   int a = 10, b = 4, C;
   printf("a is %d and b is %d\n", a, b);
   C = a + b;
   printf("a + b is %d \n", C);
   C = a - b;
   printf("a - b is %d \n", C);
   C = a * b;
   printf("a * b is %d\n", C);
   C = a / b;
   printf("a / b is %d\n", C);
   C = a % b;
   printf("a %% b is %d\n", C);
   return 0;
}
```

Output-

```
a is 10 and b is 4
a + b is 14
a - b is 6
a * b is 40
a / b is 2
a % b is 2
```

2. Relational Operator-

Operator Symbol	Operator Name
==	Equal To
!=	Not Equal To
<	Less Than
>	Greater Than
<=	Less Than Equal To
>=	Greater Than Equal To

Program-

```
#include <stdio.h>
 int main()
{
   int a = 10, b = 4;

   if (a > b)
      printf("a is greater than b\n");
   else
      printf("a is less than or equal to b\n");

   if (a >= b)
      printf("a is greater than or equal to b\n");
   else
      printf("a is lesser than b\n");

   if (a < b)
      printf("a is less than b\n");
   else
      printf("a is greater than or equal to b\n");

  if (a <= b)
      printf("a is lesser than or equal to b\n");
   else
      printf("a is greater than b\n");

   if (a == b)
      printf("a is equal to b\n");
   else
      printf("a and b are not equal\n");
```

```
    if (a != b)
        printf("a is not equal to b\n");
    else
        printf("a is equal b\n");

    return 0;
}
```

Output-

a is greater than b
a is greater than or equal to b
a is greater than or equal to b
a is greater than b
a and b are not equal
a is not equal to b

3. Assignment Operators-

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3

Program-

```
#include <stdio.h>
int main()
{
    int a = 10;
    a += 10;
    printf("Value of a is %d\n", a);
    a -= 10;
    printf("Value of a is %d\n", a);
    a *= 10;
    printf("Value of a is %d\n", a);
    a /= 10;
    printf("Value of a is %d\n", a);
    return 0;
}
```

Output-
Value of a is 20
Value of a is 10
Value of a is 100
Value of a is 10

4. Logical Operators-

Logical AND (&&)

Logical OR (||)

Logical NOT (!)

AND Program-

```
#include <stdio.h>
int main()
{
   int a = 10, b = 20;

   if (a > 0 && b > 0)
  {
      printf("Both values are greater than 0\n");
   }
   else
 {
      printf("Both values are less than 0\n");
   }
   return 0;
}
```

Output-
Both values are greater than 0

OR Program-

```
#include <stdio.h>
int main()
{
   int a = -1, b = 10;

   if (a > 0 || b > 0)
```

```
 {
      printf("Any one of the given value is greater than 0\n");
   }
   Else
  {
      printf("Both values are less than 0\n");
   }
   return 0;
}
```

Output-

Both values are less than 0

NOT Program-

```
#include <stdio.h>
int main()
{
   int a = 10, b = 20;

   if (!(a > 0 && b > 0))
   {
      printf("Both values are greater than 0\n");
   }
   Else
 {
      printf("Both values are less than 0\n");
   }
   return 0;
}
```

Output-

Both values are less than 0

5. Bitwise Operator-

Operator	Meaning of operator
&	Bitwise AND operator
|	Bitwise OR operator
^	Bitwise exclusive OR operator
~	One's complement operator (unary operator)
<<	Left shift operator
>>	Right shift operator

Program-

```
#include <stdio.h>
int main()
{

   int a = 5, b = 3;

   printf(" bitwise a & b is= %d \n", a & b);
   printf(" bitwise a | b is= %d \n", a | b);
   printf(" bitwise a ^ b is= %d \n", a ^ b);
   printf(" bitwise ~ of a is= %d \n", ~a);
   printf(" bitwise a leftshift by 1 bit is= %d \n", b << 1);
   printf(" bitwise a righttshift by 1 bit is= %d \n", b >> 1);
   return 0;
}
```

Output-

```
 bitwise a & b is= 1
 bitwise a | b is= 7
 bitwise a ^ b is= 6
 bitwise ~ of a is= -6
 bitwise a leftshift by 1 bit is= 6
 bitwise a righttshift by 1 bit is= 1
```

1.3 Data Input and Output- getchar(), putchar(), scanf(), printf(), gets(), puts().

getchar()-accept a single input from the user.
putchar()-is used to write a character.

Program-

```
#include <stdio.h>
int main()
{
   char ch;
   printf("enter any character= ");
   ch= getchar();
   printf("\nentered character is= ");
   putchar(ch);
   return 0;
}
```

Output-
enter any character= h
entered character is= h

printf()-function is used to print formatted text to output screen.

scanf()-reads formatted input from standard input stream.

Program-

```
#include <stdio.h>
int main()
{
   int ch, var;
   printf("\nenter any character= ");
   scanf("%c",&ch);
   printf("\nentered character is=%c",ch);
   printf("\nenter any integer= ");
   scanf("%d",&var);
   printf("\nentered integer is=%d",var);
   return 0;
}
```

Output-
enter any character= u
entered character is=u

enter any integer= 9
entered integer is=9

gets(), **puts()**-Both the functions are involved in the input/output operations of the strings. The gets() function is risky to use since it doesn't perform any array bound checking and keep reading the characters until enter is encountered.

Program-

```
#include<stdio.h>
int main()
{
   char s[10];
   printf("Enter the string= ");
   gets(s);
   printf("\nentered string is=");
   puts(s);
   return 0;
}
```

Output-
LRPSmweVOn.c:(.text+0x24): warning: the `gets' function is dangerous and should not be used.
/tmp/LRPSmweVOn.o
Enter the string= hello world
entered string is=hello world