Chapter No: 2

Introductions to C Language

Question No	Question
1	Explain structure of C Program
2	What is Token?
3	Define Term Identifier, Keywords & Constants
4	What is Tri-Graph character?
5	Explain different data types available in C.
6	Define the term precedence and associativity.
7	Explain different operators available in C.
8	Explain implicit type casting & explicit type casting.
9	What is the use of header file?
10	Explain printf() and scanf() function with example.
11	Explain getch(), getche() and getchar() with example.
12	Explain putchar() with example.

1. Explain Structure of C Program.

Answer:

Basic structure of C program:

Documentation Section
Link Section
Definition Section
Global declaration Section
main()
{
    Declaration Part;
    Executable Part;
}
Subprogram Section
User defined Function

Various sections of C program are as under:-

Documentation Section (Optional)	It contains set of comment lines giving the details of the program like name of program, author name and date of
Link Section	It contains the list of standard header files which are needed to execute the program. For example ***stdio.h*** and ***conio.h***
Definition section (Optional)	It defines pre-processor directives
Global section (Optional)	Variables that are common for more than one function are declared in global
main()	It is a function that must be present in every C program. It is the first function that is called up when the program is executed. It is used with two curly brackets to group the statements together
Subprograms (Optional)	The sub program section contains user defined functions.

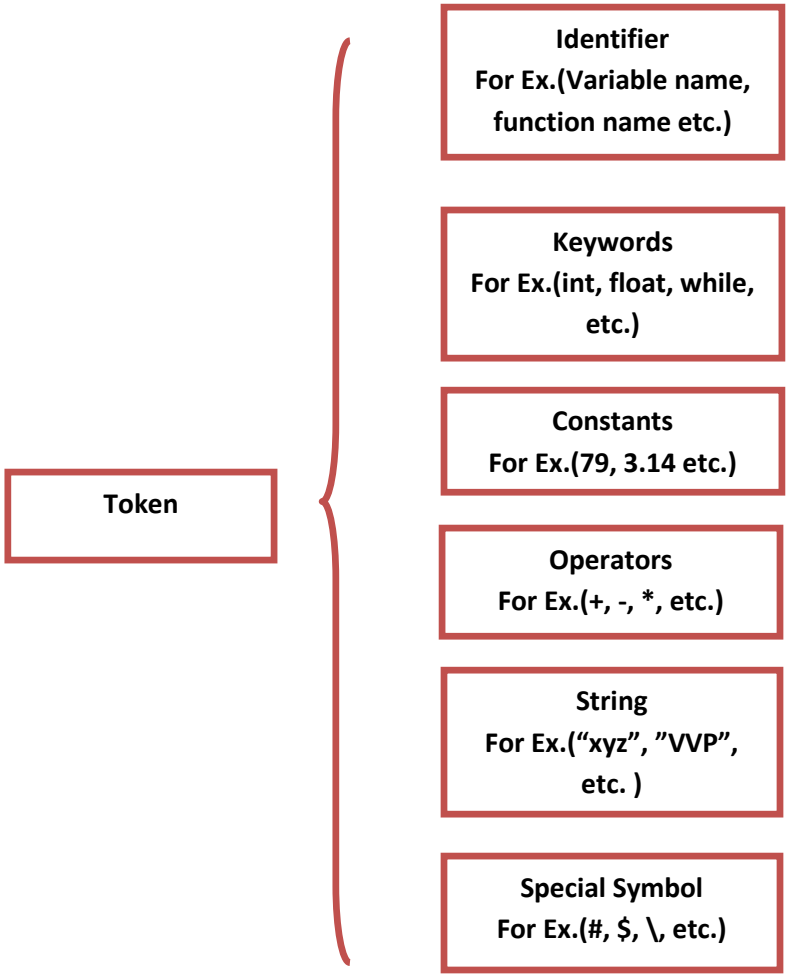
Example:-

```
// a program to print hello world  (Documentation Section)

#include <stdio.h>
#include<conio.h>                    } Link Section

void main()
{
        Printf("Hello
        World");                     } main Section
       getch();
}
```

2. What is Token?

Answer:

- The smallest individual unit of C program is known as ***C tokens***.
- The C compiler breaks down the instructions into various components.
- A ***token*** is a source program text which compiler does not break into further elements.
- This basic elements recognized by the compilers are known as ***"tokens".***
- These elements are further classified into keywords, identifiers, Constants, Operators, Strings and Special Symbols.



3. Define the term Identifier, Keyword & Constant.

Answer:

IDENTIFIERS:

Identifier means that every element in the program must have its own unique name.

There are certain rules for identifiers:-

1) It consists of letters and digits.
2) First character must be an alphabet or underscore.
3) Underscore (_) is only allowed as a special symbol.
4) Both upper and lower cases are allowed but they are not equivalent. (NUM is not same as num).

CONSTANTS:

Constant is a value stored by giving a name and which cannot be changed. The value stored in a constant cannot be changed during program execution.

e.g. A +10= 98

In the above example, 10 is known as constant.

C has two types of constants each with its own specific uses.

1) **Literal Constant**
2) **Symbolic Constant**

1) Literal Constant:

A literal constant is a value that is typed directly into the source code wherever it is needed.

Literal constant is further divided into four types:-

1) **Integer (80, 90, 100 etc)**
2) **Real (3.21, 45.13 etc)**
3) **Character ('a', 'y' etc)**
4) **String ("VVP", "tejas" etc)**

2) Symbolic Constant

A symbolic constant is a constant represented by a name in your program.

Like a literal constant, symbolic constant cannot change.

The actual value of symbolic constant needs to be entered only once, when it is first defined.

C has two methods for defining a symbolic constant.

a) **#define Directive**:-

#define directive is one of C's preprocessor directives.

There is no '=' sign while assigning a value to the constant while using define.

The #define directive is used as follows:-

Ex:

#define X 3.14

#define MESSAGE "Hello how r u"

b) **Using Const Keyword:-**

The second way to define a symbolic constant is with the ***const keyword.***

Const is a modifier that can be applied to any variable declaration.

A variable declared to be const cannot be modified during program execution only initialized at the time of declaration

KEYWORDS:

Keyword is a word which has a specific meaning for a particular language.

Keywords are reserved words which have a standard and a predefined meaning and cannot be used as an identifier.

In C total 32 keywords are available as below.

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**

4. **What is Tri – Graph?**

Answer:

- Some characters are not available in some keyboard.
- So, for these characters ANSI C introduced the new concept of “tri graph”.
- Tri graph provides a way to enter characters that are not available on some keyboard.
- Tri graph consists of three characters. (First two question mark and last some other character)

Tri graph characters	Translation
??=	#
??(	[
??)	]
??<	{
??>	}
??/	\
??-	~

Q-5 Explain Different Data types available in C.

Answer:

Data type defines following:

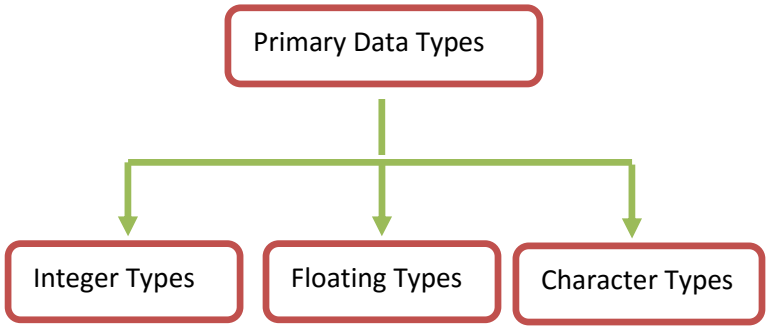
1) Which type of value a variable can store
2) Size of variable in memory
3) The range of value that a variable can store

There are two types of data types in C.

1) Primary data types
2) Derived data types

Primary Data Types:

There are four different types of primary data types.



1) Integer Types:

Integer is used to store whole numbers like 23, 256 etc.

To declare variable as integer **int** keyword is used.

Type	Size (Bytes)	Range
int or singed int	2	-32,768 to 32,768
unsigned int	2	0 to 65535
long int or singed long int	4	-2,147,483,648 to 2,147,483,647
Unsigned long int	4	0 to 4,294,967,295

2) Floating Types:

Floating types are used to store real number like 2.3, 34.12 etc.

To declare variable as floating, float keyword and double keyword is used.

Type	Size (Bytes)	Range
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

3) **Character Types:**

Character types are used to store characters value.

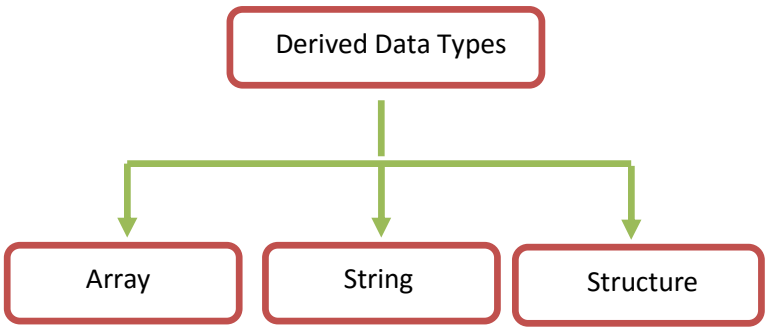
To declare variable as character, char keywords are used.

Type	Size (Bytes)	Range
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

Derived Data Types:

Data type which are derived from fundamental or primary data type known as derived data type.

Basically three types of derived data types.



1) **Array:**

Array is collection of same type data items known as single name. Using array we can enter multiple value of same data type into one variable.

2) **String:**

It is collection of characters. We can say string is array of characters ended by null character.

3) **Structure:**

It is collection of variables of different data types under a single name. We can say structure is user defined data type which can store different values of different data types.

Q-6 Define the terms precedence and associativity.

Answer:

When more than one operator is used in an expression, C language has predefined rules of to evaluate expression based on priority of operators. This rule of priority operators is known as precedence of operators.

In arithmetic operators,

Operator	Priority	Associativity
*, /, %	First	Left to Right
+, -	Second	Left to Right

Associativity:

If operators have same priority in one expression then rules define which operator is executed first in which order. This rule is known as associativity.

For example:

If expression contains * and / operators then both have same priority, so associativity rules will be applied for evaluating expression.

Ex:

D = A + B – C.

In this + and – both have same priority

Consider following table for both priority (precedence) and associativity. In this table top most operator has highest priority.

Operator	Meaning	Associativity
( ), ++, - -	Parenthesis, postfix increment, postfix decrement	Left to right
++, - -, sizeof	Prefix increment, prefix decrement, sizeof operator	Right to left
*, /, %	Multiplication, division, modulus	Left to right
+, -	Addition, subtraction	Left to right

Q-6 Explain different types of operators used in C.

Answer:

Operator: It is a symbol which is used to tell the compiler which mathematical or logical operations are to be applied on operands (data).

C language has following 8 different types of operators.

1) Arithmetic operators
2) Relational operators
3) Logical operators
4) Assignment operators
5) Increment or decrement operators
6) Bit wise operators
7) Ternary (Conditional) operators
8) Other operators

Arithmetic operators:

It is used to perform arithmetic operation like addition, subtraction, division, multiplication and modules.

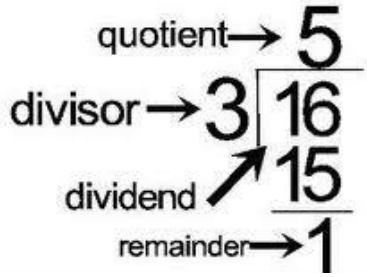
Operator Name	Meaning

+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Modulus

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,c;
        a=10,b=20;
        clrscr();
        c=a+b;
        printf("%d",c);
        getch();
}
```

Note:
Division - / operator is used to find the **quotient.**
Modulus - % operator is used to find **remainder.**



Relational Operator:

These operators are used to find the relationship between two operands.
The answers of these operators are always either true or false.
Consider a = 10 and b = 5 for following table.

Operator	Meaning	Answer
a==b	Check the value of a and b are **same or not**	False
a>b	Check the value of a **is greater than** the value of b	True
a<b	Check the value of a is **less than** the value of b	False
a>=b	Check the value of a is **greater than or equals** to the value of b	True
a<=b	Check the value of a is **less than or**	False

	equals to the value of b	
a!=b	Check the value of **a is not equal** to the value of b	True

Logical Operators:

Logical operators are used to combine the more than one condition.

C language has 3 different types of logical operators.

The output of logical operators is always either true or false.

Consider a = 10, b = 5 and c = 3 for following table.

Operator	**Example**	**Output**
&& - Logical and	(a > b) && (a>c)	True
|| - Logical or	(b > a) || (a>b)	True
! – Logical not	!(a>b)	False

Assignment Operators:

Assignment operator is used to assign the value to variable.

To assign the value to variable following syntax is used.

Syntax:

<Variable name> = <expression>;

Expression can be any constant, variable name or any valid expression.

Example: a = 10;

C language also support shorthand operator.

Shorthand operator can be used with statement having following form:

<Variable name> = <variable name> <operator> <expression>;

Here both variable names must be same.

This form is converted into following form using shorthand operator.

<Variable name> <operator> = <expression>;

Assignment operator	Shorthand
a = a + 5;	a+=5;
a = a – 5;	a-=5;
a = a * 10;	a*=10;
a = a / 20;	a/=20;

Increment or Decrement operator:

It is unary operator that means it requires only one operator.

Operator can be written either after or before the operand.

Operator	Meaning
++	Increment operator
--	Decrement operator

Increment operator is used to increase the value of variable by one and decrement operators is used to decrease the value of variable bye one only.

If increment operator is placed after variable then it is known as post increment and if increment operator is placed before variable then it is known as pre increment.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a1,b1;
        a1=10;
        b1=a1++; // post increment
        printf("%d",b1);
        printf("%d",a1);
        getch();
```

Note:

Here first value of a1 is assigned to variable b1 and then value of a1 is increment by 1, because of it is post increment.

}

Bitwise operator:

C language also supports some operators which can perform operation on bit level. So using this operator we can perform operation on 0's and 1's bit. Consider a = 10 (1010) and b = 5 (0101).

Operator	Example	Output
& - Bitwise AND	c = a & b	0
| - Bitwise OR	c = a | b	15
^ - Bitwise Exclusive OR (XOR)	c = a ^ b	15
<< - Left Shift	c = a << 1	20
>> - Right Shift	c = a >> 1	5
~ - Bitwise 1's complement	c = ~a	-11

For Bitwise and (&) operator if both bits are 1 then only answer is 1.

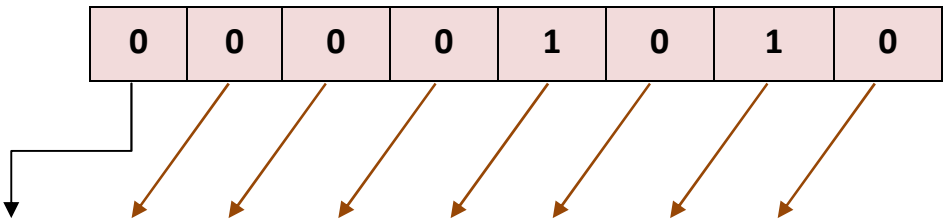
For Bitwise or(|) operator if one the bit is 1 then answer is 1 and if both bits are 0 then only answer is 0.

For Bitwise exclusive or (XOR) operator if both bits are different then only answer is 1 other wise answer is 0.

In left shift operator if we shift one bit to left then result will be in multiplication of 2, if we shift 2 bits in left side then result will be in multiplication of 4 and so on.

For Example:

a=10; if we write a<<1 then answer is 20 as below.



0	0	0	1	0	1	0	0

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a1,b1;
        a1=10;
        b1=a1<<1;
        printf("%d",b1);
        b1=a1>>1;
        printf("%d",b1);
        getch();
}
```

Note:

Here value of a1 is left shifted 1 bit so answer is 20.
Then value of a1 is right shifted 1 bit so answer is 5.

Conditional operator:

It is also known as **ternary operator**.
It works like if else statement.

Syntax:
<Condition> ? <Statement 1> : <statement 2>;

If condition becomes true then statement 1 will be executed.
If condition becomes false then statement 2 will be executed

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int max,a,b;
```

```
        clrscr();
        a=10, b=20;
        max = (a>b)? a : b;
        printf("maximum value is %d",max);
        getch();
}
```

Other operators:

In that some special operators like **comma operator**, **dot (.) operator**, **arrow (->) operator** and **sizeof** operator are used.

Comma operator is used to declare multiple variable, expression and also used in printf() or scanf() function.

Dot (.) operator is used to access the member of structure using structure variable.

Arrow (->) operator is used to access the member of structure using pointer variable.

Sizeof() operator is used to find **the no of bytes occupied by specified data type or variable**.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a;
        clrscr();
        printf("%d",sizeof(a)); // size of integer is 2 bytes so output is 2
        getch();
}
```

Q-7 what is type conversion? Explain implicit type conversion. (Type casting)

Answer:

When an expression of C language contains different types (data type) of variables and constant then it will be converted into same data type at the time of evaluation. This process is known as type conversion.

There are two type of type conversion in C language.

1) Implicit type conversion (Type casting)
2) Explicit type conversion

> **Note:**
> Expression means collection of variables, operators and or constant.
> For example: C = A + B

Implicit type conversion:

When C language expression contains different types of variables and constant then implicit type conversion rule is applied.

Rule:

Lower types of operands are automatically converted into higher types of operands. This rule is applied by C language compiler.

> **Note:**
> Operand means variables or constants.
> For Ex: C = A + 10. In this C, A and 10 are operands and + is operator.

Example:

```
#include<stdio.h>          //stdio – standard input output header file
#include<conio.h>          //conio – console input output header file
void main()                // entry point of C program
{
        int  c;            // c is variable of type integer
        float b, a;        // b and a is variable of type float
        clrscr();          // used to clear screen
        c = 10;            // 10 is assigned to variable c
        b = 5.5;           // 5.5 is assigned to variable b
        a = c / b;         // expression contains two different data types
                           //int and float
        printf("%f",a);    //used to display the output of variable a
        getch();           // get a character from keyboard
}
```

In this example expression a = c / b has two different data types.
Variable c is integer and variable b is float, so as per the rule of type conversion (implicit type casting) lower data type value is converted into higher data type value automatically by C language compiler. So variable c value is 10 which becomes 10.0 Now a = c / b means a = 10.0 / 5.5;

Explicit Type Casting (Type Conversion):

When user needs the type conversion explicitly, then type casting or explicit type casting is used.
Type conversion is automatically process by compiler of C language and type casting is performed by user.

Syntax:

(Type name) expression;

Here type name is the name of data type we want to convert the expression into. The converted values are used during evaluation of expression only.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int sum=47;          //sum is variable of type integer
    int n=10;            // n is variable of type integer
    float avg;           // avg is variable of type float
    clrscr();            // used to clear the output screen
    avg=sum/n;           // here answer of sum/n is integer
                         //and store in float variable avg
    printf("avg=%f",avg);
    avg=(float)sum/n;    //sum is converted into float
```

```
        printf("avg = %f",avg);
        avg=(float)(sum/n);        //sum/n is converted into float
        printf("avg=%f",avg);
        getch();
}
```

Note: Here avg=sum/n is type conversion process and after two bold expression is type casting process.

Output: 4, 4.7, 4.0

Q-9 what is use header file?

Answer:

Header file is file with extension .h.

Header file contains function declaration and definitions. There are two types of header files.

1) Inbuilt Header file
2) User defined header file

1) **Inbuilt Header File:**
   The file which comes with compiler is known as inbuilt header file.
   For Ex. stdio.h, conio.h, math.h, string.h etc.

2) **User Defined Header File:**
   The file which is created or writes by programmer is known as user defined header file.
   To use header file in program, use need to us preprocessing directives #include.
   For use of inbuilt header file following form is used.

Syntax:

```
#include<file name>
```

Here file name is the name of inbuilt header file with extension.

For use of user defined header file following form is used.

Syntax:

#include ”file name”

Here file name is the name of user defined header file.

Q-10 Explain printf() & scanf() with example.

Answer:

printf():

This function is used to print any message or value on output screen.

To use this function in program user need to include stdio.h header file.

Declaration of printf() in header file is as below:

Declaration:

int printf(const char *format, <variable name>…)

In this format tag can be written like:

%[flags][width][.precision][length] specifier

Here, % and specifier is compulsory to write and rests of all tags are optional.

Specifier	For which value it is used
c	Character
d or i	Decimal Integer (10,20, etc)
f	Decimal floating value (2.3, 4.56, etc)
s	String (no of characters)
lf	Double floating value

printf() return the number of characters printed by function.

Example:

```
#include<stdio.h>
```

```
#include<conio.h>
void main()
{
        int a=10;
        clrscr()
        printf("%d",a);
        getch();
}
```

Note:
Here we want to print value of which is integer, so we used as specified.

scanf():

This function is used to get the value of variable from user or get the value of variable at runtime.

To use scanf() we need to include stdio.h header file in program.

Declaration of printf() in header file is as below:

Declaration:

int scanf(const char *format, <Address of variable>…);

In this format tag can be written like:

%[flags][width]specifier

Specifiers are as per table shown in printf().

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a;
        clrscr();
        scanf("%d",&a);
        printf("%d",a);
        getch();
}
```

Note:
Here scanf() get the value of variable a from user.

Q-11 Explain getch(), getche() and getchar() with example.

Answer:

getch() gets a character from console but does not display it on the screen.

This function does not take any argument but it reads a single character from the console and returns that character.

Syntax:

int getch(void);

Example:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        clrscr();
        printf("\n enter the character");
        ch=getch();
        printf("\n %c",ch);
        getch();
}
```

getche():

getche() reads a character from console and echoes it to the screen.

This function does not take any argument but it reads a single character from the keyboard and returns the character read from the standard input device like keyboard.

Syntax:

int getche(void);

Example:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        clrscr();
        printf("\n enter the character");
        ch=getche();
        printf("\n %c",ch);
        getch();
}
```

getchar():

getchar() read a single character from the standard input terminal.

This function is used to read more than one character until newline character is entered.

We can also store these characters to an array.

This function does not take any argument and returns the character read from the input device.

Syntax:

```
int getchar(void);
```

Example:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int c;
        clrscr();
        while((c=getchar())!='\n')
                printf("\n %c",c);
        getch();
}
```

getch()	getche()	getchar()
Normally getch() is used at the end of main(). It just accepts a key stroke and never displays it and proceeds further.	getche() will accept a character from keyboard and display immediately, it does not wait for Enter key to be pressed for proceeding.	getchar() will accept a character from keyboard and display immediately and need enter key to press for proceeding.

Q-11 Explain putchar() with example.

Answer:

putchar() is a single character output function.
putchar function is used to write characters one at a time.

Syntax:
putchar(<variable name>);

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch='Y';
        putchar(ch);
        getch();
}
```