UNIT 1: Introduction to C language

History of C

The ALGOL is a root of all languages. It was introduced in 1960. It was the first computer language. The reason behind wide use is the concept of Structured programming.
In 1967, the language BCPL (Basic Programming Language) was developed by Martin Richards for writing System Software.
In 1970, the language B was developed by Ken Thompson. It has a many features of BCPL .The early versions of UNIX developed by this language in Bell Laboratories.
In 1972 the language C was developed by Dennis Ritchie at AT & T's (American Telecommunication & Telegraph) Bell Laboratories in USA.

1960	**'ALGOL 60'**	**International Group**
1967	**'BCPL'**	**Martin Richards**
1970	**'B'**	**Ken Thompson**
1972	**'C' Programming**	**Dennis Ritchie**

Features & Characteristics of 'C'

1. **Structured Programming –**
   It contains functions, Modules, Blocks, Selection, looping control constructs which helps to write structured program.
2. **Portable –**
   It is portable because it runs program on different machines and operating system.
3. **Rich Set of built-in-functions –**
   Its strength is variety of built-in-functions provided by C library which helps to develop any complex program easily.
4. **Extendibility –**
   It supports built-in-functions as well as user defined functions. Programmer can add continuously functions.
5. **General Purpose Language-**

It helps to develop most of the applications like Mathematical, Business, Scientific and System Software.

6. **Debugging and Testing –**
   Due to structured programming it makes program easy to Debugging, Testing and Maintenance**.**
7. **Variety of Keywords, Data types and Operators**
   It has a variety of Data types, Arithmetic Operators and 32 Keywords , which helps to develop program easily.

1.2 Basic structure of 'C' programming

Documentation Section	/* Program developed by.......*/

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

Definition Section	#define PI 3.14
Global Definition Section	void message();

main() function section
{
 	
Declaration Section
--------- 	
--------- 	
Executable Section
--------- 	
 	
--------- 	
}
	void main()
	{
	int r;
	float area;
	printf("Enter the radius of circle");
	scanf("%d",&r);
	area=PI*r*r;
	printf ("Area of Circle=%f",area);
	message();
	getch();
	}

Sub Function Section
Function1
Function2
--------- 	
--------- 	
Function n
(User Defined function)
	void message()
	{
	printf("Have a nice day!!!");
	}

1. **Documentation Section-**
   Consist of set of comments helps to understanding program.
   (e.g. Author or Other details)
2. **Link Section-**
   It provides instructions to compiler to link function from the system library.
3. **Definition Section-**
   It defines Symbolic Constants.

4. **Global Declaration Section-**

   To declare global variables that is used in more than one functions.

5. **main() function section-**

   The program must have main () function . It contains two parts

   a) **Declaration Part-**

   To declare all the variables which are used in executable part.

   b) **Executable Part-**

   It contains the executable statements and different function calls. The program execution begin with the opening brace { and ends with closing brace }

6. **Function Section-**

   It contains all the user defined functions that are called in main() function section.

Language Fundamental

Character Set and Tokens

Character Set

Character set consist of Alphabets, Digits and Special Symbols. It helps to represent information. The C character set is a Upper and Lowercase Alphabets, Digits , Special Characters and Alphanumeric character i.e. combination of Alphabets and Digits

1	**Alphabets**	**A,B,C,................,Z** **a,b,c,..................,z**
2	**Digits**	**0,1,2,3,4,5,6,7,8,9**
3	**Special Character**	**! @ # $ % ^ & * . () [] {} _ - + = ; : ” ’ , < > ? / \ | ,**

C Tokens

The smallest individual units in C program are called as Token. C has following Tokens

C Tokens	
1	Keywords
2	Identifiers
3	Variables
4	Data Types
5	Operators

Keywords and Identifiers

Keywords

Keywords are reserved words. They have fixed and predefined meaning and these meaning cannot be changed. The keywords cannot be used as variable name because which is not allowed in C. Keywords help in building blocks of program. There are only 32 keywords available in C.

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

Identifiers

Identifiers are user defined words and are used to give names to variables, arrays, functions, structures etc.

Rules for Identifiers
1. First character must be an alphabet or underscore.
2. It consists of only Letters, Digits and Underscore.
3. Only 31 characters are significant.
4. Keyword cannot be used as variable
5. No commas or Blank space allowed within a variable name.

Variables and Data types

Variables

a. Variable is a data name.
b. It may be used to store data value.
c. It may take different values at different time times during execution of program.
d. Variable name can be chosen by programmer in a way that it reflects its nature in program.
e. A data type is associated with each variable decides the types of value it will store.
f. The rules for naming the variables are same as that for naming identifiers.

Example
roll_no
grade
amount
avg_weight

Declaration of Variable

Variable must be declared before it is used in program. Its specifies its name and data type.

Example
int roll_no;
float amount;
char grade

Initialisation of Variable

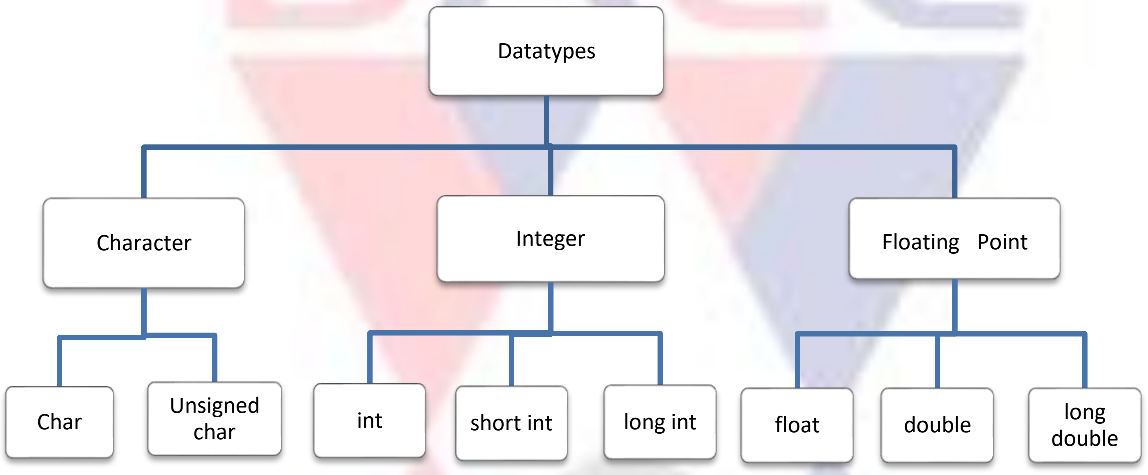
To assign the value to variable at the time of declaration is called as variable initialisation. Otherwise in only declaration it will takes garbage value.

Example

int count=1;
float x=4.2;
char ans=’y’;

Data Types in C

Every Program works on Data. Programming language provides way to store data with its type. When variable is declared you have to specify that what type of data it can contain.
The Data types specify that size and which type of value stored in that variable.
C language is rich in its Data type.



A. Character Data Type (char)

It stores a single character. A single character can be defined as char type data. Characters are usually stored in one byte of internal storage. The qualifier signed or unsigned may be explicitly applied to char.

Data type	Range	Byte	Format
Char	-128 to +127	1	%c
unsigned char	0 to 255	1	%c

B. Integer Data type (int)

It stores integer or whole numbers. To provide some control over the range of numbers and storage space, C has different integer storage namely int, short int and long int in both signed and unsigned forms.

Data type	Range	Byte	Format
Int	-32768 to 32767	2	%d
unsigned int	0 to65535	2	%u
short int	-32768 to 32767	2	%d
unsigned short int	0 to65535	2	%u
long int	-2147483648 to 2147483647	4	%ld
unsigned long int	O to 4294967295	4	%lu

C. Float Data type (float)

It stores floating point or real numbers with 6 digit precision when the accuracy provided by float is not sufficient then the type double and long double can be used.

Data type	Range	Byte	Format
Float	3.4e-38 to 3.4e+38	4	%f
Double	1.7e-308 to 1.7e+308	8	%lf
long double	3.4e-4932 to 1.1e+4932	10	%ld

Void Data type

The void data type has no values. It is usually to specify the type of function. The type of function is void means it doesn't return any value to calling function.

D. Enumerated Data type

A user defined data type enumerated data type is along with its set of identifiers can be created by following declaration

Example

```
enum datatype name(const1,const2,........,constn);
enum datatype daysofweek(sun,mon,tue,wed,thu,fri,sat);
```

Operators

An operator is a symbol that represents an operation. Which instruct compiler to perform some action? Operators can be of

A. **Unary Operator** –It operates on only one operand (e.g a++,b--)
B. **Binary Operator-** It operates on two operands (e.g. a>b&&a>c)
C. **Ternary Operator-** It operates on three operands (e.g conditional operator (? :))

1.4.1 Types of Operators

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increment and Decrement Operators
6. Conditional Operators
7. Bitwise Operators
8. Other Operators

1. Arithmetic Operators

It is used to perform arithmetic operations. It is called as binary operators because it operates on two operands.

Operator	**Meaning**	**Example**
+	Addition	7+5=12
-	Subtraction	5-2=3
*	Multiplication	4*2=8
/	Division	6/2=3
%	Modulo Division	5%2=1

2. Relational Operators

It is used to compare expressions. An expression containing relational operators evaluates to either True (1) or False (0). Any non zero value is considered True in C and zero is False. Thus even negative values are True.

Operator	Meaning	Example
<	Less than	a < b
<=	Less than equal to	a <= b
>	Greater than	a > b
>=	Greater than equal to	a >= b
==	Equal to	a == b
!=	Not equal to	a != b

3. Logical Operators

It can be used to test more than one conditions and make decision. It used to combine two or more expressions (relational). The entire expression is called logical expression which evaluates to True (1) or False (0).

Operator	Meaning	Example
&&	AND	(marks<=70&&marks>=60)
||	OR	(a<7||b>10)
!	NOT	!(a>b)

4. Assignment operators

It is denoted by (=). It is used to assign the value of an expression to variable.

Example

```
sum=a+b;
a = b;
```

Shorthand Assignment Operator

These are obtained by combining certain operators with the = operator.

Example

```
X+=y   i.e. x=x+y;
x-=y   i.e x=x-y;
k/=3   i.e. k=k/3;
j*=5   i.e. j=j*5;
```

5. Increment or Decrement Operator

C Provides the two unary operators i.e. Increment (++) or Decrement Operators (--)
++ Increments the value of operands by one
-- Decrements the value of operands by one

A. Prefix

In prefix the Increment or Decrement operators written before operand.
The Increment or Decrement done before the value of operand used in an expression (e.g. ++a ,--b)

B. Postfix

In postfix the Increment or Decrement operators written after operand.
The Increment or Decrement done after the value of operand used in an expression (e.g. a++ , b--)

6. Bitwise operator

A. C Programming provides us six operators for manipulation of data at bit level
B. These operators operate on an Integer and character but not on float or double.
C. Using Bitwise operators we can manipulate individual bit.
D. The Bitwise operators is for testing the bits or shifting them to the right or left.

Operator	**Name of Operator**
~	One's Compliment
>>	Right Shift
<<	Left Shift
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR

Bitwise Operators AND, OR, XOR –

Input Bits		**AND**	**OR**	**XOR**
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Rules for Table

1. If any one of the bit is zero then resultant AND is Zero.
2. If both bits are One then resultant AND is One.
3. If both bits are Zero then resultant OR is Zero.
4. If both bits are same then resultant XOR is Zero.
5. If both bits are different then XOR is One.

7. Other Operators

Assignment Operator

The Assignment Operator (=) is used to assign the value an expression to variable.

Syntax

```
variable = expression;
```

Example

```
sum=a+10;
a=5;
```

Shorthand Assignment Operator

These are obtained by combining certain operators with the = operator

Syntax –

```
variable operator=Expression;
```

Example –

```
x += y  means   x=x+y
x -= y
x /= y
x *= y
```

Type Conversions in expression

C permits mixing of constants and variables of different types in an expression. In these types of operations data type of one operand is converted into data types of another operand. This is known as implicit type conversion. These conversions are done by the C compiler according to some predefined rules of C language.

1. Implicit Type Conversion

C automatically converts any intermediate values to the proper type so that the expression can be evaluated without loosing any significance. This automatic conversion is known as implicit type of conversion.
The rule is that if the operands are of different types, the lower type is automatically converted to the higher type before the operation proceeds.
If one operand is float and other is integer, the integer is converted into float and result is float.
If one operand is long double then the other will be converted to long double, and the result will be long double.
If one operand is double then the other will be converted to double, and the result will be double.

Example

int a;
float b;
a=9.7; When 9.7 is assigned to a it gets converted into 9
b=20; When 20 is assigned to b it gets converted into 20.000000

2. Explicit Type Conversions

'C' provides a unary operator for explicit type conversion called cast operator.

Syntax

(Type_name) expression;

The expression is converted to the specified data type locally only for the purpose of evaluation of the expression.

Example

a=(int)9.4;	Here 9.4 is converted to integer by conversion.
c=(int)a+b;	Here a is converted to integer and then added to b.
c=(int)(a+b);	Here the result of a+b is converted to integer.

UNIT 2: Managing I/O Operations

Console based I/O and related built-in I/O functions

'C' doesn't have any built in Input/output statement. Therefore all Input/output operations are carried out with the help of functions like printf() and scanf(). Console Input/output functions are used to accept input from keyboard and display it on output screen. C Language provides readymade functions for this process. Console I/O functions can be classified into two categories unformatted and formatted I/O functions. The most of the I/O functions are defined in <stdio.h> the header file . So while using these functions in our program we have to include the header file like #include<stdio.h>

1. **Character Input/Output**
2. **String Input/Output**

1. Character Input/Output-

Character Input/Output functions are useful for reading single character from the keyboard and writing single character to the monitor.

Inorder to read single character we use getch(),getche(),getchar() functions.These functions returns the character that has been most recently typed in.

a) **getch()**

This function gets one character input from keyboard but doesn't display on the screen .

Syntax

```
variable name=getch();
```

b) **getche()**

This function gets one character input from keyboard and display on the screen.

Syntax

```
variable name=getche();
```

c) **getchar()**

This function gets one character input from keyboard but doesn't display on the screen but it require the user hit the enter key after the character typed in.

Syntax

```
variable name=getchar();
```

We make use of putch() and putchar() functions for writting single character on terminal

a) **putch()**

This function prints a single character on screen.

Syntax

variable name=putch();

b) **putchar()**

This function prints a single character on screen.

Syntax

variable name=putchar();

There are number of character functions supported by C. These are contained in file<ctype.h>

Functions	**Meaning**
Isalpha()	To check is alphabet
Isalnum()	To check is alphanumeric
Isdigit()	To check is digit
Islower()	To check is lowercase character
Isupper()	To check is uppercase character
tolower()	To check is alphabet
toupper()	To convert character in uppercase
tolower()	To convert character in lowercase

2. String Input/Output

String Input/Output functions are useful for reading string from the keyboard and writing string to the monitor.

Inorder to read string we use gets() function and for writing string we use puts()function.

a) **gets()**

This function read the string character by characters contineously from input device until the enter key is pressed.

Syntax

gets(variable name);

b) **puts()**

This function writes the string on screen.

Syntax

```
puts(variable name);
```

Formatted Input/Output Functions

C Language provides functions for reading and writing formatted data from Input and Output device are printf() and scanf() functions.

a) scanf()

This function reads the character from standard input device. Interprets them according to the format specifiers and store them in corresponding argument

Syntax

```
scanf("control String",&arg1,&arg2,-------,&argn);
```

Example

```
scanf("%d %f",&x,&y);
scanf (" %2d",&y);
scanf (" %d%d",&n1,&n2);
scanf (" %15s",city);
scanf (" %s",name);
```

b) printf()

This function prints the captions and values on screen. It produces output easy to use and understandable format.The printf() function provides features to control alignment and spacing output.

Syntax

```
printf("control String",arg1,arg2,-------,argn);
```

Example

```
printf("Welcome to C Programming");
printf("%d %f",x,y);
printf("sum=%d",s);
printf(“ %.2f”,y);
printf(“ %7.2f”,x);
printf(“ %6d”,y);
printf(“ %-6d”,y);
printf(“ %s”,name);
printf(“ \n\n”);
printf(“  ”);
```

Q. Write a C Program to display message “Welcome to C”.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("Welcone to C Programming”);
getch();
}
```

Q. Write a C Program to calculate Addition of two numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter two numbers");
scanf("%d %d",&a,&b);
c=a+b;
printf("Addition=%d",c);
getch();
}
```

Q. Write a C Program to calculate Addition, Subtraction, Multiplication and Division of two numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,add,sub,mul;
float div;
clrscr();
printf("Enter two numbers");
scanf("%d %d",&a,&b);
add=a+b;
sub=a-b;
mul=a*b;
div=a/b;
printf("Addition=%d",add);
printf("Substraction=%d",sub);
printf("Multiplication=%d",mul);
printf("Division=%f",div);
getch();
}
```

Q. Write a C Program to calculate Area of Circle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r;
float area;
const float pi=3.14;
clrscr();
printf("Enter the radius of circle");
scanf("%d",&r);
area=pi*r*r;
printf("Area of Circle=%f",area);
getch();
}
```

Q. Write a C Program for Swapping of two numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,t;
clrscr();
printf("Enter two numbers");
scanf("%d%d",&a,&b);
t=a;
a=b;
b=t;
printf("\n After Swaping a=%d",a);
printf("\n After Swapping b=%d",b);
getch();
}
```

UNIT 3: Decision Making and Looping

Introduction

C program is a set of statements which are normally executed sequentially sometimes in programs there is need to test some condition at some point and selecting the alternative paths depending upon the result of condition or repeat a group of statements until certain specified conditions are met. C language processes such decision making capabilities by supporting the following statements.

Decision Making Structure

- Simple If statement
- if...else statement
- Nested if...else statement
- else if ladder
- switch statement

if statement

In simple if statement it test or evaluates the condition first if it is true then the Statement Block will get executed and if the condition is false then the Statement Block is skipped and statement-a is executed

Syntax

```
If(test condition)
{
  statement block;
 }
Statement-a;
```

if...else Statement

In if...else statement it test or evaluates the condition first if it is true then the Statement Block1 (called if block) is executed and if the condition is false then the Statement Block2(called the else block) is executed.

Syntax

```
If(test condition)
{
   statement Block1;
}
else
{
   Statement block2;
}
```

Q. Write a Program to find larger number between two numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
clrscr();
printf("enter two numbers");
scanf("%d%d",&a,&b);
if(a>b)
{
        printf("larger no=%d",a);
}
else
{
        printf("larger no=%d",b);
}
getch();
}
```

Q. Write a Program to find number is even or odd

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no;
clrscr();
printf("enter any no");
scanf("%d=",&no);

if(no%2==0)
{
        printf("enter no is even");
}
else
{
        printf("enter no is odd");
}

getch();
}
```

Nested if...else Statement

It is possible to nest if...else statement i.e. we can make use of more than one if...else statement by nesting them. There is no limit to the number of if...else nesting.

The logic of execution is like If the test condition-1 is false, the statement -3 will be executed otherwise it continuous to perform the second test. If the test condition-2 is true, the statement-1 will be evaluated; otherwise the statement-2 will be evaluated and then control is transferred to the statement-a.

Synatax

```
If(test condition-1)
{
        If(test condition-2)
        {
                statement block1;
        }
        else
        {
                 statement block2;
         }
 }
else
{
        statement block3;
}
 statement-a;
```

Q. Write a Program to enter any three numbers and display larger number between them.

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

int a,b,c;
clrscr();
printf("enter three numbers");
scanf("%d %d %d",&a,&b&c);

if(a>b)
{
if(a>c)
{
        printf("larger Number=%d",a);
}
else
{
        printf(" larger Number=%d",c);
}
else if(c>b)
{
        printf("larger Number=%d",c);
}
else
{
        printf("larger Number=%d",b);
}
}
getch();
}
```

else if ladder

When multiple decisions or conditions are involved we make use of else...if ladder. The else...if ladder is a chain of if where each else has an associated if.

In else...if ladder the conditions are evaluated from the top to downwards. As soon as the true condition is found, the statement associated it is executed and the control is transferred to the statement-a by skipping the rest of statements. When all n conditions becomes false then the final else containing the default statement will be executed.

Syntax –

```
If(test condition1)
  {
      statement block1;
  }
else If(test condition2)
  {
      statement block2;
  }
else If(test condition n)
  {
      statement block n;
  }
else
{
   default-statement;
}
statement-a;
```

Q. Write a Program to enter any number upto 5 digit and find number of digit in that number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no;
float per;
clrscr();
printf("enter a number");
scanf("%d",&no);

if(no>=9)
{
        printf("Number is One Digit");
}
else if (no>=99)
{
        printf("Number is Two Digit");
}
else if (no>=999)
{
        printf("Number is Three Digit");
}
else if(no>=9999)
{
        printf("Number is Four Digit");
}
        else if(no>=99999)
{
        printf("Number is Five Digit");
}
else
{
        printf("enter number upto five digit");
}
getch();
}
```

The switch statement

C provides a switch statement to avoid the use of series of if. C has a built-in multi way decision statement known as switch. The switch statement test the value of a given variable against a list of case values and when a match is found a block of statement associated with that case is executed. When all test values becomes false then the final default case block will be executed.

Syntax -

```
switch(expression)
 {
     case 1:
     statement block1;
     break;
     case 2:
     statement block2;
     break;
     case 3:
     statement block3;
     break;
     ..........
    ..........
    ..........
    default :
    default statement block;
    break;
 }
 Statement-a;
```

Q. Write a program to enter any character and find character is vowel or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch;
clrscr();
printf("enter character");
scanf("%c”,&ch);

switch(ch)
{
case ‘A’:
case ‘a’:
printf("Character is Vowel");
break;

case ‘E’:
case ‘e’:
printf("Character is Vowel");
break;

case ‘I’:
case ‘i’:
printf("Character is Vowel");
break;

case ‘O’:
case ‘o’:
printf("Character is Vowel");
break;

case ‘U’:
case ‘u’:
printf("Character is Vowel");
break;

default:
printf("Character is Vowel");
break;
}
getch();
}
```

Conditional Operator

It is called as ternary operator in C. It requires three operands. The operator pair (**? and :**) is used to construct conditional expression of the form.

expression1? expression2: expression3

Here expression1 is evaluated first. If it is true (non zero) then expression1 is evaluated and becomes the value of conditional expression and if it is false (zero) then the value of entire expression is that of expression3.

For example consider the following example.

```
Suppose a=5 and b=10
        Max = a > b ? a : b;
```

Here first expression a>b is evaluated, if it is false so the value of b becomes the value of conditional expression and it is assigned to variable max.

Loop Control Structure

- while loop
- do-while loop
- for loop
- Nested for loop

while loop

Loops are used when we want to execute a part of program or block of statements several times. Like a if statement here we have single or block of statement it is known as body of loop. In while loop first condition is tested or evaluated if it is true then statements in the body of loop are executed. While loop would keep on getting executed till the condition being tested remains true when the condition becomes false the body of loop is skipped and control is transferred to out of the loop i.e. statement-a.

Syntax

```
while(condition)
{
        Block of statements;
}
Statement-a;
```

Q. Write a Program to display name 10 times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();

while(i<=10)
{
        printf("Aditya\n");
        i++;
}
getch();
}
```

Q. Write a Program to display 1 to 10 numbers and sum of the digits of numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1,sum=0;
clrscr();

while(i<=10)
{
        printf("\n %d",i);
        sum=sum+i;
        i++;
}
printf("sum of the digits=%d",sum);
getch();
}
```

Q. Write a Program to enter any number and print in reverse order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,rem,rev=0;
clrscr();
printf("\n enter the no");
scanf("%d=",&no);

while(no>0)
{
        rem=no%10;
        rev=rem+(rev*10);
        no=no/10;
}
printf("\n Revers no=%d",rev);
getch();
}
```

do- while loop

The do-while statement is also used for looping . The body of loop may contain a single statement or block of statements. In do-while the statement inside loop body is executed and then the condition is tested or evaluated if it is true then again statements in the body of loop are executed. This process continuous until the condition becomes false, when the condition becomes false the body of loop is skipped and control is transferred to out of the loop i.e. statement-a.

Syntax

```
do
{
        block of statements;
} While(condition);

statement-a;
```

Q. Write a Program to print name 20 times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();

do
{
        printf("Adti \n");
        i++;
} while(i<20);
getch();
}
```

Q. Write a Program to print Multiplication Table of number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1,n,prod;
clrscr();
printf("enter any number");
scanf("%d"&n);

do
{
        prod=n*i;
        printf("\n%d",prod);
        i++;
} while(i<10);
getch();
}
```

Difference between while and do-while loop

	while loop	**Do-while loop**
1	It is a entry controlled loop structure	It is a exit controlled loop structure
2	The condition is checked before the statements block execution	In do- while the condition is checked after the statements block execution.
3	In while If the condition is initially false, the statement block will not be executed even once	In do-while Statement block will executed at least once even at first condition becomes false
4	The loop variable has to be initialized before the while loop begins	The loop variable need not be initialized

5	**Syntax-**
	
	while(condition)
	{
	
	Block of statements;
	}
	Statement-a;
		Syntax-
		
		do
		{
		
		Block of statements;
		
		} While(condition);
		
		statement-a;

For loop

For loop is the most frequently use loop construct. For statement provides initialisation of counter. Test condition and counter Increment/Decrement all in single line. For is entry controlled loop construct. It first checks the test condition and if it is true only then it executes the loop body.

We can initialise more than one value in for loop. The test condition may have compound relation and the testing need not be limited only to the loop control variable. Both the initialization and increment sections are omitted in for statement. However the semicolon separating the section must be remain.

Syntax

```
for(initialisation; condition; Increment)
{

  block of statements;

}
statement-a;
```

Q. Write a Program to find factorial of number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,fact=1;
clrscr();
printf("\n enter a number");
scanf("%d",&n);

for(i=1;i<=n;i++)
{
        fact=fact*i;
}
printf(" \n factorial=%d",fact);
getch();
}
```

Q. Write a Program to find numbers which is divisible by 5 from 1 to 100.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();

for(i=1;i<=100;i++)
{
if(i%5==0)
{
        printf("\n%d",i);
}
}
getch();
}
```

Nested for loop

When a loop is written inside the body of another loop, then it is known as nesting of loops. The for loop also can be used nesting type i.e. one for statement within another for statement is allowed in C.

Syntax –

```
for(initialisation; condition; Increment)
{
for(initialisation; condition; Increment)
{
        block of statements;
}
}
statement-a;
```

Q. Write a Program to print following pattern.

```
****
****
****
****
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(i=1;i<=4;i++)
{
for(j=1;j<=4;j++)
{
        printf("*");
}
printf("\n");
}
getch();
}
```

Q. Write a Program to print following pattern.

```
*
**
***
****
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,line;
clrscr();
printf("How many lines pattern");
scanf("%d",line);
for(i=1;i<=line;i++)
{
for(j=1;j<=i;j++)
{
        printf("*");
}
printf("\n");
}
getch();
}
```

Jump Statements

- break statement
- continue statement
- goto statement
- exit

break statement

Break statement is used to unconditionally exit from the loop. It is useful in a situation where we want to jump out of loop immediately without waiting to get back to the conditional test. When the loops are nested, the break would only exit from the loop containing it. That is, the break will exit only a single loop.

Syntax

while(condition)
{
……….
………
If(condition)
break;
………..
………..
}
Statement-a;
	do
	{
	……….
	………
	If(condition)
	break;
	………..
	………..
	
	} While(condition);
	
	statement-a;
		for(initialisation; condition; Increment)
		{
		……….
		………
		If(condition)
		break;
		………..
		………..
		}
		statement-a;

Q. Write a Program to demonstrate break statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();

for(i=1;i<=10;i++)
{
        printf("%d",i);
if(i==5)break;
        printf("$$$$$");
}
getch();
}
```

continue statement

Continue command is used to skip the rest of the commands of the loop for the current occurrence and move the program pointer to the top of the loop. When the keyword ‘continue’ is encountered inside any C loop, control automatically passes to the beginning of the loop.

Q. Write a Program to demonstrate continue statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,line;
clrscr();

for(i=1;i<=10;i++)
{
        printf("%d",i);
if(i==5)continue;
        printf("$$$$$");
}
getch();
}
```

goto statement

C supports goto statement for unconditional branching from one point in the program to another . The goto statements requires label. The label identifies the place in the program where the program control is to be transferred when the goto is encountered.

Syntax

```
goto label name
.......
.......
.......
label name:
Statement 1;
        Statement 1;
```

Q. Write a Program to demonstrate goto statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
start:
{
        int a=5;
        printf("%d",a);
        a++;
}
if(a<15)
        goto start;
getch();
}
```

exit

We can jump out of the loop using break statement or goto statement. In similar way, we can jump out of the program by using the library function exit(). We can break out of the program and return to the operating system, we can use exit() function.

```
..........
.........
If(condition)
exit(0);
...........
...........
```

The exit function takes an interger value as its argument. Normally zero is used to indicate normal termination and a nonzero value to indicate termination due to some error or abnormal condition.
The use of exit() functions requires to inclusion of header file <stdlib.h>.

Compound Statements

A compound statement consists of several statements enclosed within pair of curly braces {}.
Compound statement is also known as block os statements . Note that there is no semicolon after the closing brace.

Example

```
{
        int l=10,b=6,h=8;
        int volume;
        volume=l*b*h;
}
```

The variables that are declared inside a block can be used only inside that block.

NULL Statements

The semicolon on a line is a null statement and this doesn't perform any action.

Example

```
main()
{
        int x=2;
        ;       //null statement
}
```

UNIT 4: Programs through Conditional and Looping Statements

Introduction

C program is a set of statements which are normally executed sequentially sometimes in programs there is need to test some condition at some point and selecting the alternative paths depending upon the result of condition or repeat a group of statements until certain specified conditions are met. C language processes such decision making capabilities by supporting the following statements.

Decision Making Structure

- Simple If statement
- if...else statement
- Nested if...else statement
- else if ladder
- switch statement

Program of if...else Statement

Q. Write a Program to find larger number between two numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
clrscr();
printf("enter two numbers");
scanf("%d%d",&a,&b);
if(a>b)
{
        printf("larger no=%d",a);
}
else
{
        printf("larger no=%d",b);
}
getch();
}
```

Q. Write a Program to find number is even or odd

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no;
clrscr();
printf("enter any no");
scanf("%d=",&no);

if(no%2==0)
{
        printf("enter no is even");
}
else
{
        printf("enter no is odd");
}

getch();
}
```

Programs for Practice	
1	Q. Write a Program to find number is positive or negative
2	Q. Write a Program to find year is leap year or not leap year
3	Q. Write a Program to find person is eligible for voting or not eligible

Program of Nested if...else Statement

Q. Write a Program to enter any three numbers and display larger number between them.

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

int a,b,c;
clrscr();
printf("enter three numbers");
scanf("%d %d %d",&a,&b&c);

if(a>b)
{
if(a>c)
{
        printf("larger Number=%d”,a);
}
else
{
        printf(" larger Number=%d”,c);
}
else if(c>b)
{
        printf("larger Number=%d”,c);
}
else
{
        printf("larger Number=%d”,b);
}
}
getch();
}
```

Program of else if ladder

Q. Write a Program to enter any number upto 5 digit and find number of digit in that number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no;
float per;
clrscr();
printf("enter a number");
scanf("%d",&no);

if(no>=9)
{
        printf("Number is One Digit");
}
else if (no>=99)
{
        printf("Number is Two Digit");
}
else if (no>=999)
{
        printf("Number is Three Digit");
}
else if(no>=9999)
{
        printf("Number is Four Digit");
}
        else if(no>=99999)
{
        printf("Number is Five Digit");
}
else
{
        printf("enter number upto five digit");
}
getch();
}
```

Q. Write a simple program to enter a marks of five subject and display Total, Percentage and Class.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int sub1,sub2,sub3,sub4,sub5,total;
float per;
clrscr();
printf("enter the marks of five subject");
scanf("%d %d %d %d %d",&sub1,&sub2,&sub3,&sub4,&sub5);
total=sub1+sub2+sub3+sub4+sub5;
per=total/5;
printf("Total Marks =%d",total);
printf("Percentage=%f",per);

if(per>=70)
{
        printf("Distinction");
}
else if(per>=60)
{
        printf("First Class");
}
else if(per>=50)
{
        printf("Second Classs");
}
else if(per>=40)
{
        printf("Pass");
}
else
{
        printf("Fail");
}
getch();
}
```

Program of switch statement

Q. Write a program to enter any character and find character is vowel or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch;
clrscr();
printf("enter character");
scanf("%c”,&ch);

switch(ch)
{
case ‘A’:
case ‘a’:
printf("Character is Vowel");
break;

case ‘E’:
case ‘e’:
printf("Character is Vowel");
break;

case ‘I’:
case ‘i’:
printf("Character is Vowel");
break;

case ‘O’:
case ‘o’:
printf("Character is Vowel");
break;

case ‘U’:
case ‘u’:
printf("Character is Vowel");
break;

default:
printf("Character is Vowel");
break;
}
getch();
}
```

Programs for Practice
Write a Program to enter month number and find name of the month.
Write a menu driven program to calculate area of Rectangle, area of Square, area of Triangle, area of Circle as per user choice
Write a Program to find person is eligible for voting or not eligible

Program on Loop Control Structure

- while loop
- do-while loop
- for loop
- Nested for loop

while loop

Q. Write a Program to display name 10 times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();

while(i<=10)
{
        printf("Aditya\n");
        i++;
}
getch();
}
```

Q. Write a Program to display 1 to 10 numbers and sum of the digits of numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1,sum=0;
clrscr();

while(i<=10)
{
        printf("\n %d",i);
        sum=sum+i;
        i++;
}
printf(“sum of the digits=%d”,sum);
getch();
}
```

Q. Write a Program to enter any number and print in reverse order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,rem,rev=0;
clrscr();
printf("\n enter the no");
scanf("%d=",&no);

while(no>0)
{
        rem=no%10;
        rev=rem+(rev*10);
        no=no/10;
}
printf("\n Revers no=%d",rev);
getch();
}
```

Programs for Practice	
1	Write a Program to find no is palindrome or not palindrome.
2	Write a program to print Fibonacci series upto 20 terms.
3	Write a Program to find no is Armstrong number or not Armstrong number.
4	Write a Program to enter number in digit and display in words (245- Two Four Five)
5	Write a Program to find the L.C.M. and G.C.D. of a number

Program on do- while loop

Q. Write a Program to print name 20 times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();

do
{
        printf("Adti \n");
        i++;
} while(i<20);
getch();
}
```

Q. Write a Program to print Multiplication Table of number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1,n,prod;
clrscr();
printf("enter any number");
scanf("%d"&n);

do
{
        prod=n*i;
        printf("\n%d",prod);
        i++;
} while(i<10);
getch();
}
```

Program on For loop

Q. Write a Program to find factorial of number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,fact=1;
clrscr();
printf("\n enter a number");
scanf("%d",&n);

for(i=1;i<=n;i++)
{
        fact=fact*i;
}
printf(" \n factorial=%d",fact);
getch();
}
```

Q. Write a Program to find numbers which is divisible by 5 from 1 to 100.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();

for(i=1;i<=100;i++)
{
if(i%5==0)
{
        printf("\n%d",i);
}
}
getch();
}
```

Q. Write a Program to find number prime or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,prime=1;
clrscr();
printf("\n enter any no");
scanf("%d",&n);

for(i=2;i<n;i++)
{
if(n%i==0)
{
        prime=0;
        break;
}
}
if(prime==0)
{
        printf("  no is not prime ");
}
else
{
        printf("no is prime");
}
getch();
}
```

Programs for Practice	
1	Write a Program to calculate **x^y** .
2	Write a program to find sum of 50 natural/odd/even numbers.
3	Write a Program to find Armstrong numbers between 1 to 500.
4	Write a Program to find Prime numbers between 1 to 100.
5	Write a Program to print 1/1!+ 2/2!+ 3/3!+.............N.
6	Write a Program to print $1+x^2/y^2+x^4/y^4+$.............N.

Program on Nested for loop

Q. Write a Program to print following pattern.

```
****
****
****
****
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(i=1;i<=4;i++)
{
for(j=1;j<=4;j++)
{
        printf("*");
}
printf("\n");
}
getch();
}
```

Q. Write a Program to print following pattern.

```
*
**
***
****
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,line;
clrscr();
printf("How many lines pattern");
scanf("%d",line);
for(i=1;i<=line;i++)
{
for(j=1;j<=i;j++)
{
        printf("*");
}
printf("\n");
}
getch();
}
```

Jump Statements

- break statement
- continue statement
- goto statement
- exit

break statement

Q. Write a Program to demonstrate break statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();

for(i=1;i<=10;i++)
{
        printf("%d",i);
if(i==5)break;
        printf("$$$$$");
}
getch();
}
```

continue statement

Q. Write a Program to demonstrate continue statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,line;
clrscr();

for(i=1;i<=10;i++)
{
        printf("%d",i);
if(i==5)continue;
        printf("$$$$$");
}
getch();
}
```

goto statement

Q. Write a Program to demonstrate goto statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
start:
{
        int a=5;
        printf("%d",a);
        a++;
}
if(a<15)
        goto start;
getch();
}
```

UNIT 5: Arrays and Strings

Introduction

The fundamental data types namely int, float, char etc. are very useful but variable of these data type can be stored only one value at a time. So they can handle limited amount of data. In many applications we need to handle large volume of data for that we have to need powerful data types that would facilitate efficient storing, accessing and manipulation of data items.

C supports derived data type known as Array.

Definition

An array is collection of data items of the same data type
An array is fixed-size sequenced collection of elements of the same data type.
An array is also called as Subscripted Variables

Features of Array

1) An array is a collection of similar elements.
2) The location of array is the location of its first element.
3) The first element in array is numbered zero so the last element is less than the size of array.
4) The length of array is the number of its elements in array.
5) The type of an array is the data type of its element.
6) An array is known as subscripted variable.
7) Before using array it's type and dimension must be declared.

Introduction to One- Dimensional Array

Definition

A list of items can be given one variable name using only one subscript and such a variable is called a single subscripted or single Dimension Array.

Syntax

data type arrayname[size];

Example

int rollno[3];
float marks[5];
char name[30];

0	1	2	
100	101	102	
1340	1342	1344	----- Memory Address

Array elements are always stored in contiguous memory locations and since the data type int occupies 2 bytes of memory, each element will be allocated 2 bytes.

Declaration

We can declare array in the same way the ordinary variable.

Syntax

data type arrayname[size];

Example

int rollno[5];
int rollno[ ];
float num[5];

Initialization

We can initialise elements of array in the same way the ordinary variable.

Syntax

data type arrayname[size]={list of values};

Example

int rollno[5]={1,2,3,4,5};
int rollno[ ]={1,2,3,4,5};
float num[5]={2.5,7.2,9.2,6.2,3.3};

Accessing and displaying array elements

Program to enter elements in array and display.

```
include<stdio.h>
#include<conio.h>
void main()
{
int i,a[50],n;
clrscr();
printf("\n  How many elements want to enter:");
scanf("%d",&n;);
        printf("\n  Enter elements in array:");
for(i=0;i<5;i++)
{
        scanf("%d",&a[i]);
}
for(i=0;i<5;i++)
{
        printf("\n %d",a[i]);
}
        getch()
}
```

Introduction to Two-Dimensional Array

Definition

An array whose elements are specified by more than one subscript is known as multi dimension array (also called Matrix)

Declaration

Syntax

data type arrayname[row size][column size];

Example

int student[5][2];

	C0	C1
R0	1	67
R1	2	73
R2	3	82
R3	4	90
R4	5	58

char name [4][10];

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
R0	S	W	A	P	N	I	L	\0			
R1	S	A	N	T	O	S	H	\0			
R2	J	A	Y	D	I	P	\0				
R3	S	A	N	D	I	P	\0				

Initialisation Array

int number[3][4]={8,12,25,37,42,52,68,79,81,92,100,102};

char city[5][10]={“Mumbai”,”Punei”,”Satara”,”Kolhapur”,”Sangli”};

Accessing and displaying Array elements

Program to print Addition of two Matrix

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10][10],b[10][10],s[10][10],r,c,i,j;
clrscr();
printf("Enter the no. row's & column's");
scanf("\n%d%d",&r,&c);
printf("\n\nFirst matrix:\n");
for(i=0;i<r;i++)
{
```

```
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("\n\nSecond matrix:\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
s[i][j]=a[i][j]+b[i][j];
}
}
printf("\n\nAddition of matrix:\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("\t%d",s[i][j]);
}
printf("\n");
}
getch();
}
```

Introduction to Strings

Definition

In C character string simply treats as array character. The size in a character string represents the maximum number of characters that the string can hold.

Declaration

char name[10];

It declares the name as a character array (string) variable that can hold a maximum 10 characters. Each character of the string is treated as an element of the array name and is stored in the memory as follows.

‘W’	‘E’	‘L’	‘ ’	‘D’	‘O’	‘N’	‘E’	‘\0’

A Character string terminates with an additional null character. Thus the element in name[10] holds the null character ‘\0’. When declaring character arrays, we must allow one extra element space for the null terminator.

Initialization

We can initialise string in array the same way as the ordinary variable.

Syntax

data type arrayname[size]={list of values};

Example

```
char name[5]={‘s’,’w’,’a’,’p’,’n’,’i’,’l’,’\0’};
char name[]=”siddhi”;
char name[5]={‘P’};
char  *colour[]={“Red”,”Green”,”Blue”,”Yellow”};
```

Standard Library Function

The C library support large number of string manipulation functions in the header files <string.h> that can be used to carry out many string manipulation functions.

strlwr()	This function converts the given string into lowercase character.
Syntax	strlwr(string);

Example	#include<stdio.h>
	#include<conio.h>
	#include<string.h>
	void main()
	{
	char string[30];
	clrscr();
	printf(“Enter String”);
	gets(string);
	strlwr(string);
	printf(“Converted String=%s”,string);
	getch();
	}

strupr()	This function converts the given string into uppercase character.
Syntax	strlupr(string);

strlen()	This function counts and returns the number of characters in a string. Excluding null character.
Syntax	strlen(string);

strrev()	This function reverse the given string contents and store it again in same string variable.
Syntax	strrev(string);

strcat()	This function concatenates (joins) two strings means it appends the second string at the end of first string.
Syntax	strcat(string1,string2);

strcpy()	This function copy the contents of string2 to string1 and return sring1.the original content of string1 get lost.
Syntax	strcpy(string1,string2);

strcmp()	This function compares two strings to check if they are equal or not .If both strings are equal then it returns '0'.If they are not equal it returns value nonzero
Syntax	strcmp(string1,string2);

UNIT 6: Functions

Introduction

Function is a group of statement which are used to performing specific task.
Every 'C' program is the collection of function. We can avoid read and write same code over and over by using function.

Purpose of Function

1. Modular or Structured programming can be done by use of function.
2. Troubleshooting and debugging become easier in structured programming.
3. Individual functions can be easily built and tested.
4. Program development becomes easy.
5. It is easier to understand the program logic.
6. A repetitive task can be put into a function that can be called whenever required. This reduces the size of program.

- **Function Declaration**
- **Function Definition**
- **Calling Function**

Function Definition

It is self contained block of statement.

Syntax

```
return data type function name(argument list)
        {
                Block of Code;
        }
```

Function Declaration

To declare a function following three things required.

1. Name of function
2. Return Data type(Optional/Default int)
3. The number and types and arguments(parameters) that will be passed to functions.

Syntax

return data type function name(argument list)

Example

int sum(int a,int b,int c);
sum(int,int,int);

Here sum is function accepting three integers and returning as integer.

Function call

Any function can be called simply using it's name and arguments in statement.

Syntax

function name(argument list)

Example

message();
sum(int,int);

Types of Functions

There are two types of functions in C

1. Library functions/Predefined functions
2. User Defined functions

1. Library / Predefined functions

The Library / Predefined functions are prewritten, compiled and placed in libraries they come along with the compiler.
To use a library functions in a program it's corresponding header file must be included in program.

Library Header File	Library Functions
<stdio.h>	printf(),scanf(),getchar(),putchar()
<conio.h>	Clrscr()
<math.h>	Sqrt(),power(),sin()
<string.h>	Strcpy(),strcat(),strcmp(),strlen()

2. User defined functions

User defined functions are written by user and the user has freedom to choose the name, arguments (number and type) and return data type of function.
While creating user defined type function three factors are important

Program- Function with no parameter passing no return value

```
#include<stdio.h>
#include<conio.h>
void message();
void main()
{
        clrscr();
        message();
        getch();
}
void message()
{
        printf("Welcome to C Functions");
}
```

Program- Function with parameter passing no return value

```
#include<stdio.h>
#include<conio.h>
void addition(int x,int y);
void main()
{
        int a,b;
        clrscr();
        printf("Enter any two numbers");
        scanf("%d %d",&a,&b);
        addition(a,b);
        getch();
}
void addition(int x,int y)
{
        int ans;
        ans=x+y;
        printf("Addition=%d",ans);
}
```

Call by value & Call by Reference

There are two ways to pass arguments(parameters) to a function:

1. Passing arguments by value(Call by value)
2. Passing arguments by address or pointes(Call by reference)

1. Call By Value

- In this method, the contents of the arguments in the calling functions are not changed, even if they are changed in the called function.
- The contents of the actual parameters get copied into the corresponding formal parameters.

Program- Function with call by value for swapping of two numbers

```
#include<stdio.h>
#include<conio.h>
void addition(int x,int y);
void main()
{
        int a,b;
        clrscr();
        printf("Enter any two numbers");
        scanf("%d %d",&a,&b);
        addition(a,b);
        getch();
}
void addition(int x,int y)
{
        int z;
        z=x;
        x=y;
        y=z;
        printf("\n After Swapping First Number=%d",x);
        printf("\n After Swapping Second Number=%d",y);
}
```

2. Call By Reference

- In this method, the contents of the arguments in the calling functions get changed i.e. original values are changed.
- Instead of passing the value of variable, we can pass the memory address of the variable to the function. This is called as Call by reference.

Program- Function with call by reference for swapping of two numbers

```
#include<stdio.h>
#include<conio.h>
void addition(int *x,int *y);
void main()
{
        int a,b;
        clrscr();
        printf("Enter any two numbers");
        scanf("%d %d",&a,&b);
        addition(&a,&b);
        printf("\n After Swapping First Number=%d",a);
        printf("\n After Swapping Second Number=%d",b);
        getch();
}
void addition(int *x,int *y)
{
        int z;
        z=*x;
        *x=*y;
        *y=z;
}
```

UNIT 7: Introduction to Pointer

Introduction to Pointer

Definition

Pointers are the variables which holds the addresses of other variable within the memory.

A pointer is a variable that represents the location of a variable or an array element.

Pointer variables are denoted by ‘ * ptr ’ . Pointers are the derived data types.

A pointer is used for creating data structure such as linked list trees, graphs etc.

Two pointer variable can not be added. Pointer variable can not be multiplied by constant. The value can not be assigned to an arbitrary address &p=10 is illegal.

Declaration

Pointer are declared in the same way as any other variable but it is preceded with ‘*’ in declaration.

```
int *p;

float *p;

char *p;
```

Initialization

The process of assigning the address of a variable to a pointer variable is known as initialisation of pointer . Use of addressof (&) operator as a prefix to the variable name assigns its address to the pointer.

```
int *p;

P=&a;
```

We can also initialise pointer with value null or zero

```
int *p = NULL;

int *p=0;
```

Example

```
void main()
{
int i , *j;
i =  4 ;
j  =  & i ;
cout<<“The value of i is  \t”<<i<<endl;
cout<<“The value of *j is \t”<<*j;
}
```

```
Output : :
The value of i is 4
The value of *j is 4
```

Dynamic Memory Allocation

The process of allocating memory at run time is known as dynamic memory allocation. There are four library functions known as Memory management function that can be used for allocating and freeing memory during program execution.

1. malloc ()

It allocates request size of memory bytes and returns a pointer to the first byte of the allocated space.

Syntax

```
ptr=(cast type*)malloc(byte size);
ptr=(char*)malloc(10);
```

Allocates 10 bytes of memory to the pointer ptr of type char.

2. calloc ()

It allocates memory for an array element. The main difference in between calloc() and malloc() is that memory allocated by malloc() contains garbage value while calloc() contains all zero.

Syntax

```
ptr=(cast type*)calloc(n,element size);
```

3. realloc ()

It changes the size of previously dynamically allocated memory . realloc() takes two arguments . The first is the pointer referencing the memory. The second is the total number of bytes that are to be reallocated.

Syntax

```
    ptr=realloc(ptr,new size);
```

4. free ()

It frees previously allocated memory.

Syntax

```
     free(ptr);
```

UNIT 8: Structure

Introduction to Structure

Structure are derived data types. We make use of structure to represent a collection of data items of different types. Structure are useful for handling logically related data items.

Example

The personal data of people like name, address, phoneno. etc.
The data of students like rollno ,names, marks and grade etc.

Definition

A structure contains a number of data types grouped together. The data type can be smaller or of different types.

The general form of a structure definition is

```
struct  structname
   {
       Datatype member1;
       Datatype member2;
       ------------
       ------------
       ------------
   };
```

Example

The structure to represent the student information is

```
struct student
   {
       int rollno;
       charname[40];
       float percentage;
   };
```

Here the keyword struct declares the structure with the name student.

The structure students declared to consist of three fields the rollno, name and percentage. These fields are called as structure members or structure elements. once you have defined this structure data type you can declare one or more variables to be of that type.

Declaration

```
        struct student
           {
                int rollno;
                charname[40];
                float percentage;
           }s1,s2,s3;
```

OR

```
        struct student s1,s2,s3;
```

Initializing Structure

Structure variables can be initialised at the time of declaration as follows.

```
        struct student
           {
                int rollno;
                charname[40];
                float percentage;
           };
        struct student s1={101,"Jaydip",87.25};
        struct student s2={102,"Aditya",90.57};
```

Accessing Structure Members

We make use of dot (.) operator to access structure element.

```
        for e.g.
                s1.name;
                s2.percentage;
```

Structure Operations

Write a program to create structure employee and show employee details.

```
#include<stdio.h>
#include<conio.h>
struct employee
{
int eid;
char name[30],designation[20];
int salary;
};
void main()
{
struct employee x;
clrscr();
printf(" enter employee id");
scanf("%d",&x.eid);
printf("Enter the employee name");
gets(x.name);
printf("Enter  employee designation");
gets(x.designation);
printf(" enter salary");
scanf("%d",&x.salary);

printf(" Employee Details \n");
printf("Employee ID=%d",x.eid);
printf("Employee Name=%s",x.name);
printf("Employee Designation=%s",x.designation);
printf("Employee Salary=%d",x.salary);
getch();}
```

Array of Structure

When a variable of structure is declared as array type then it is called as array of structure. It is used to store same piece of information for more than one time. While accessing any member first we have to write structure variable name followed by index number followed by dot
(.) followed by member name.

Synatax

```
 struct  structname
    {
          Datatype member1;
          Datatype member2;
          ------------
          ------------
          ------------
    };
struct  structurename variable[number];
```

Example

```
struct student
    {
          int rollno;
          charname[40];
          float percentage;
    }s1[50],s2[50];
```

OR

```
struct  student  s1[50],s2[50];
```

Here we declare a structure as a array type of size 50. So we get 50 student information.

Nested Structure

One structure can be nested in another structure . This means that you can have a structure within a structure. Then it is called nested structure.

Syntax

```
 struct  structure1
    {
          Datatype member1;
          Datatype member2;
          ------------
    struct  structure2
    {
          Datatype member1;
          Datatype member2;
          ------------
    }s1;
}s2;
```

Example

```
struct  student
    {
          int rollno;
          charname[40];
    struct  marks
    {
          int sub1;
          int sub2;
    }s2;
}s1;
```

Introduction to Union

Definition

Union which are very similar to structure. It is used to group number of different variable elements. Unions are user defined data type like structure. The members of union share the same memory, only one member can be active at a time .When we store another member values, the first member values are removed from the memory and at this place, new member's values are stored. It can useful as they efficiently use computer's memory.
The size of union is equal to the maximum size occupied by its member.
It is used when we have to process each member sequentially .To declare a union keyword Union is used.

Declaration

```
union  unionname
    {
          Datatype member1;
          Datatype member2;
          ------------
          ------------
          ------------
    };
```

Example

The union to represent the student information is

```
union student
    {
          int rollno;
          charname[40];
          float percentage;
    };
```

Here the keyword union declares the union with the name student.

Write a program for enter patient details and show using Union.

```
#include<stdio.h>
#include<conio.h>
union patient
{
int pid;
char name[25];
int amount;
};
void main()
{
union patient x;
clrscr();
printf("\nenter patient id");
scanf("%d",&x.pid);
printf("\nPatient ID=%d",x.pid);
printf("\nnter the patient name");
scanf("%s",&x.name);
printf("\nPatient Name=%s",x.name);
printf("\nenter bill amount");
scanf("%d",&x.amount);
printf("\nBill Amount=%d",x.amount);
getch();
}
```

Differentiate between Structure and Union

Sr. No	Structure	Union
1	The keyword struct is used to define a Structure	The keyword union is used to define a Union.
2	When a variable associated with a Structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members	When a variable associated with a Union, the compiler allocates the memory by considering the size of the largest memory, so size of union is equal to the size of largest member.
3	Each member within a Structure is assigned unique storage area of location.	Memory allocated is shared by individual member of Union
4	Individual member can be accessed at a time.	Only one member can be accessed at a time.
5	Several members of a Structure can initialize at once.	Only the first member of Union can be initialized.

6	**Syntax**
	struct structurename
	{
	Member1;
	Member2;
	------------- 	
	};
		Syntax
		union unionname
		{
		Member1;
		Member2;
		------------- 	
		};