Subject code - R21CS101

20.9.23

Problem Solving and C Programming.

Unit - I - Introduction to C Programming

Basic Organization of Computer.

ALU - Arithmetic Logical Unit

CU - control Unit

CPU

I/P → ALU + CU + Memory → O/P

Memory has two types of memory - Primary memory and Secondary memory.

Primary Memory:

* Primary memory is also called temperory memory or Volatile memory (Changeble memory).
* Primary memory is very expensive and it has limited storge capacity.
* It will lost the data when the system is shut down.

- Primary memory has two types they are
  * RAM - Random Access memory.
  * ROM - Read Only memory.

RAM → It is used to Read the data and write data.
* It has two types Static RAM and Dynamic RAM

ROM - It helps to Read the data

Types of Rom.

PROM - Programming Readl only Memory.
EPROM - Erasable Programming Read Only Memory.
EEPROM - Electrically Erasable Programming Read only

Secondary memory:

* Secondary memory is also called Permanent and non-volatile memory.
* It is cheaper as comparing to the primary memory.
* It has Unlimited capacity to Store.

O/P

* Out put unit will converts the machine language to the User understable language.

Problem formulation:

Logical Thinking and Analysis:

* ~~Requirement~~
1. Problem analysis
2. Requirement gathering.

We need to use Proble Solving Tools after Requirement

Problem Solving Tools:

(i) Algorithm
(ii) Flow chat
(iii) Pseudocode.

3. Development alogrithm
4. Draw Flowchat
5. write Pseudocode
6. write the code
7. Compilation and Execution
8. Debugg (if compilation of error).
9. Testing.
10.

Algorithm:

Step by step representation of the programme

Flow chart:

Pictorial representation of the programme

21.9.23

Characteristics of Algorithm

* Finiteness - countable
* Definiteness - It should be define clearly
* Effectiveness - It should be clear and effectively
* Generality - Algorithm should be written in General

Flowchart:

Symbols to draw or to represent the flow chart.

1. ⬭ -> Terminal => used for start and stop
2. ▱ -> Input & output => read the input of the user
3. ▭ -> Process => For Processing the programme and used to initilize a value of a variable
4. ◇ -> Decision -> To take a decission
5. ○ -> ON Page Connector -> To connect the pages
6. ⬠ -> OFF Page Connector -> When we count finish the flow chart in one Page it is used to divide
7. → -> Flow line -> Connection between the symbols and used for true or false

Pseudocode:

Compact representation of the programme.

We have some rules to write Pseudocode

* Always capitalize and then initial words.
* Make only one statement per line.
* Indent to show hierarching of the programme and to improve readability.
* While writing Pseudocode we want use the keywords. READ, WRITE, DISPLAY, PLAY, CALCULATE (END, BEGIN, ENDIF, ENDWHILE END FOR).

Control Structures:

1. Sequence → The instruction should be in sequential manner or one by one manner.
2. Selection
3. Iteration

22.9.23

Write Algorithm, Flowchart and pseudocode to add two numbers.

Algorithm

Step 1: Start

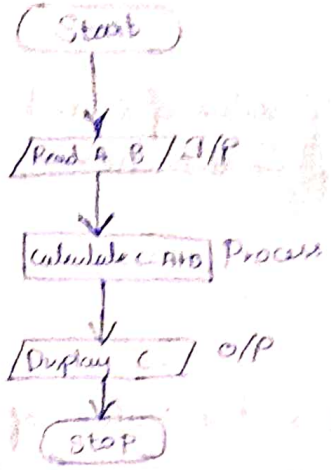
Step 2: Read the values A and B and these from the user.

Step 3: Add/Calculate the two values A and B. and store the result in C

Step 4: Display/print the result stored in C.

Step 5: Stop

Flow chart:



Pseudocode:

```
BEGIN
DISPLAY "Enter the values A and B"
READ A
READ B
CALCULATE C = A+B
PRINT C
END.
```

23.9.23

Example

Write an Algorithm and Pseudocode and Draw a flowchart for area of circle.

Algorithm

Step 1 : Start

Step 2 : Read the value of R and Initialies the Value of PI = 3.14 from the user

Algorithm :

Step 1 : Start

Step 2 : Initiates the value of PI = 3.14.

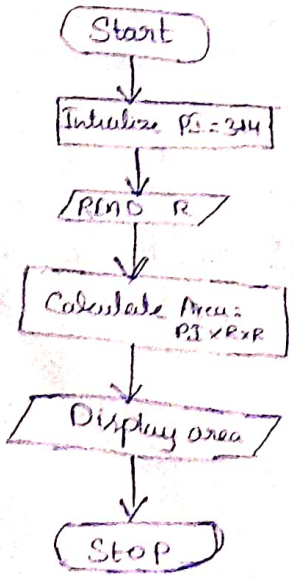
Step 3 : Read the value of R from the user.

Step 4 : Calculates the area of circle is area = PI*R*R

Step 5 : Display the calculated area in the O/P.

Step 6 : Stop.

Flowchart :



Pseudocode :

BEGIN

INITIALIZE PI = 3.14

DISPLAY "Enter the values of R."

READ R.

CALCULATE area = PI * R * R

PRINT area

END.

24.9.23

Write and Algorithm, Flowchart and Pseudocode for Calculate the Simple interest. $SI = \frac{P \times n \times r}{100}$

Algorithm

Step 1 : Start

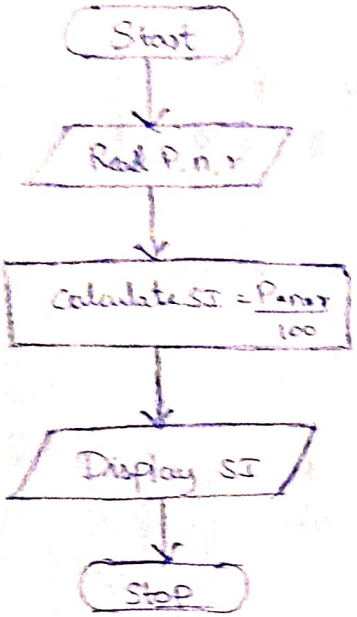
Step 2 : Read the values of P, n and r from the user.

Step 3 : Calculate the Simple Interest $SI = \frac{P * n * r}{100}$

Step 4 : Display the calculated simple interest in the O/P

Step 5 : Stop.

Flowchart



Pseudocode:

```
BEGIN
DISPLAY "Enter the value of P, n, r"
READ P
READ n
READ r
CALCULATE SI = P*n*r / 100
PRINT SI
END
```

Selection Control Structures

Write an Algorithm, Pseudocode and Flowchart to find the biggest of two numbers

Algorithm:

Step 1 : Start

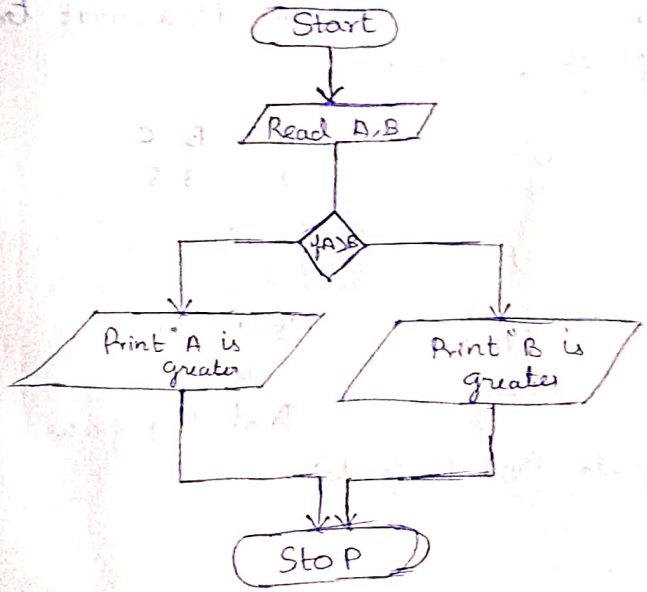
Step 2 : Read the values of A and B from the user

Step 3 : Check the condition if (A > B)

Step 4 : If Step 3 is true Print, "A is greater" Else print "B is greater".

Step 5 : Stop.

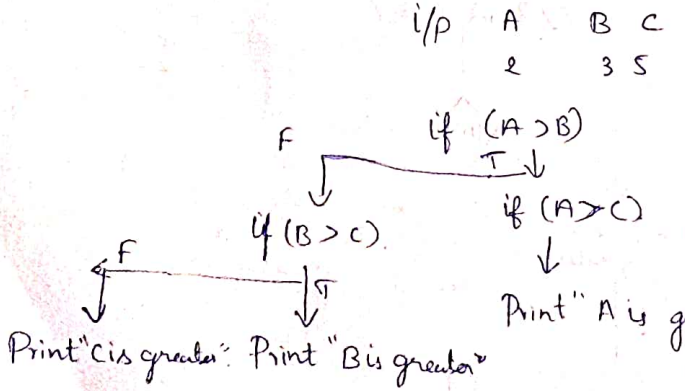
Flowchart:



Pseudocode.

```
BEGIN.
DISPLAY " Enter the value of A and B"
READ A
READ B.
IF (A>B)
    TRUE, PRINT "A is greater"
ELSE
    PRINT " B is greater"
ENDIF
END.
```

Write an algorithm, Pseudocode and Flowchart to find the greatest of Three numbers.



Algorithm :

Step 1 : Start.

Step 2 : Read the values of A, B and C from the user.

Step 3 : Check the condition if (A>B) and (A>C)

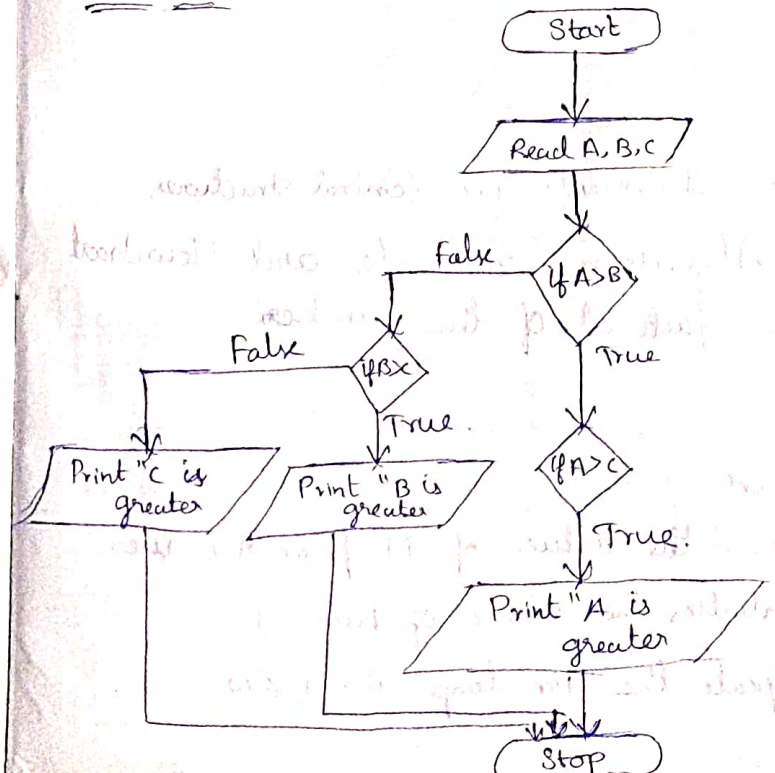
Step 4 : If step 3 is true Print "A is greater"

Step 5 : If (A>B) is false, check the condition if (B>C).

Step 6 : If Step 5 is True Print "B is greatest" Else, Print "C is greater."

Step 7 : Stop.

Flowchart :



Pseudocode:

```
BEGIN
DISPLAY " Enter the values of A,B,C"
READ A
READ B.
READ C.
If (A>B)
   If (A>C)
     TRUE , PRINT "A is greater"
ELSE
   If (B>C)
     TRUE , PRINT " B is greater".
ELSE
     PRINT "C is greater"
ENDIF
END.
```

26923

Iterative Statements (or) Control structures.

Write an Algorithm, Pseudocode and Flowchart to find the factorial of the number.

Algorithm:

Step 1 : Start

Step 2 : Read the value of N from the user.

Step 3 : Initialies the value of Fact = 1.

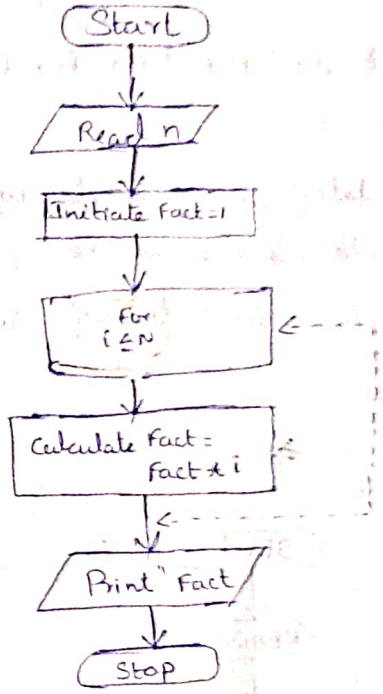
Step 4 : Repeate the For loop For i ≤ N.

Step 5 : Calculate the ~~valu~~ factorial value with the formula Fact = Fact * i

Step 6 : Display the value of Fact

Step 7 : Stop

Flow chart



Pseudocode:

```
BEGIN
DISPLAY " Enter the value of N"
READ N.
INITIALIES FACT = 1
REPEAT FOR i ≤ N.
     CALCULATE FACT = Fact * i
END FOR.
RRINT FACT
END.
```

Write an Algorithm, Pseudocode and Flowchart to find the Sum of n Natural numbers.

Algorithm:

Step 1 : Start

Step 2 : Read the value of N from the user.

Step 3 : Initialize the value of Sum=0 and i=1.

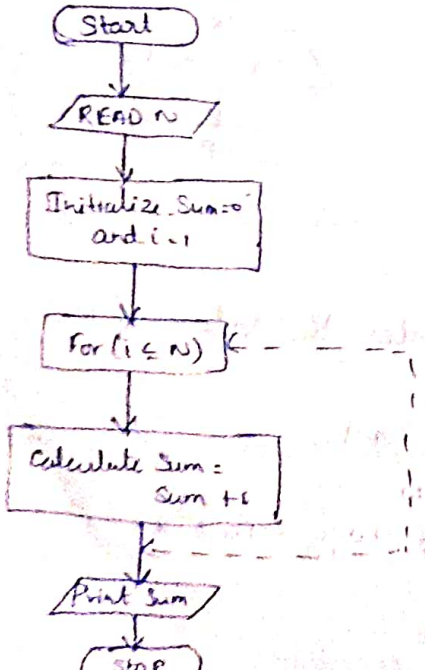
Step 4 : Repeat the For loop For i is less than (or) equal to N.

Step 5 : Calculate the natural numbers with the formula Sum = Sum+i

Step 6 : Display the value of Sum

Step 7 : Stop

Flow chart:



Pseudocode:

```
BEGIN
DISPLAY "Enter the value of N"
READ N
INITIALIZE Sum=0 and i=1.
REPEAT FOR i≤N
        CALCULATE Sum = Sum+i
END FOR
PRINT Sum
END.
```

Write an Algorithm, Pseudocode and Flowchart to find whether the given number is odd or even.

Algorithm:

Step 1 : Start

Step 2 : Read the value of A from the user.

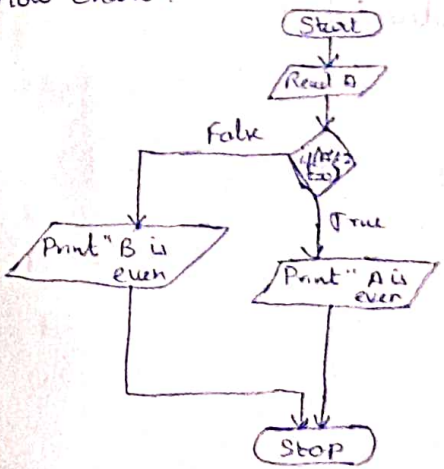
Step 3 : Check the condition if (A % 2 == 0)

Step 4 : If Step 3 is true, Print "A is even"

Step 5 : Else, Print "A is ~~even~~ odd".

Step 6 : Stop

Flow chart:



Pseudocode:

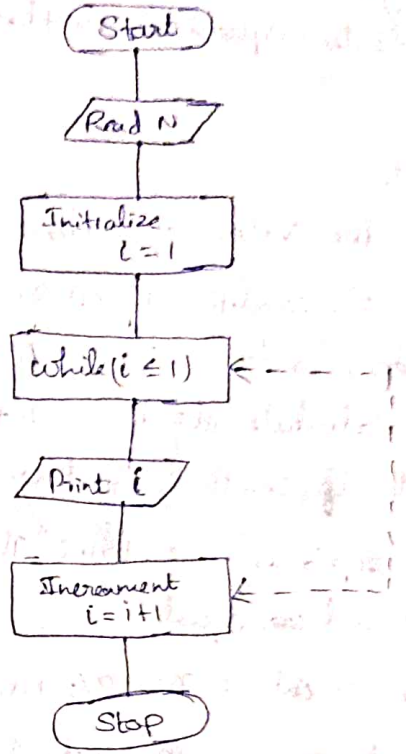
```
BEGIN
DISPLAY "Enter the value of A"
READ A
If (A % 2==0)
    TRUE, PRINT "A is even"
ELSE
    PRINT "A is odd"
ENDIF
END.
```

Write an Algorithm, Pseudocode, and Flowchart to Read 'N' and print the numbers From 1 to N.

Algorithm:

Step 1 : Start.
Step 2 : Read the value of 'N' from the user.
Step 3 : Initialize the value of i = 1
Step 4 : Repeat the while loop when i is less than (or) Equal to 'N'
Step 5 : Print the value of i
Step 6 : Increament the value of i
Step 7 : Stop.

Flowchart:



Pseudocode:

```
BEGIN
DISPLAY "Enter the value of N"
READ N.
INITIALIZE i = 1
REPEAT WHILE i ≤ 1
    PRINT i
    INCREAMENT i = i+1
ENDWHILE
END.
```

Write an Algorithm, Pseudocode and Flow chart of to Solve Quadratic equation $ax^2+bx+c=0$.

Algorithm:

Step-1 Start.

Step-2 Read the value of a, b, c, from the user.

Step-3. Check the condition If (A=0)

Step-4 If Step 3 is true, Print "A is linear"

Step-5 Else, Calculate d = b*b - 4*a*c.

Step-6. Check if whether If d=0.

Step-7. If step6 is true, Print "The roots are real and equal".

Step-8 , calculate r = -b/2*a.

Step-9. Print r in the output.

Step-10. If Step 6 is false, check if d>0

Step-11. If Step 10 is true, Print "The roots are real and unequal".

Step-12 Calculate $r_1 = -b+Sqrt(d)/2*a$ and
$r_2 = -b-Sqrt(d)/2*a$

Step-13 - Print r_1 and r_2 in the output.

Step 14 - If Step10 is false, check if d<0.
Print "The roots are not real and unequal".

Step 15 - Calculate real = -b/2*a, d=-d and
imaginary = Sqrt(d)/2*a.

Step 16 - Calculate r_1 = real + imaginary and r_2 = real-imaginary

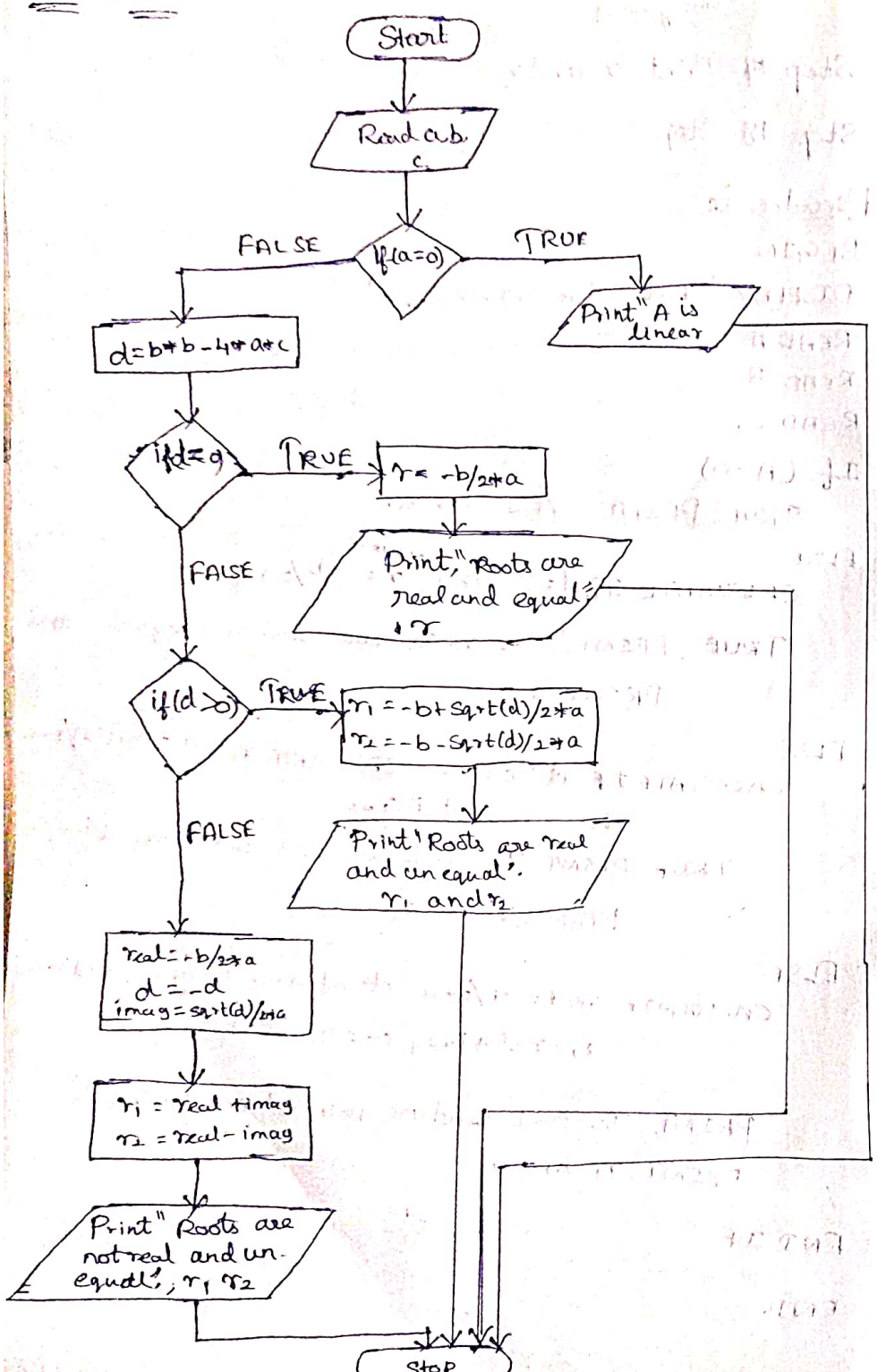
Step 17 - Print r_1 and r_2

Step 18 - Stop.

Pseudocode:

```
BEGIN
DISPLAY "Enter the value of a, b, c".
READ A
READ B
READ C.
If (A=0)
   TRUE, PRINT "A is linear".
ELSE
   CHECK If d=0 and CALCULATE r = -b/2*a.
   TRUE, PRINT "The roots are real and equal" and
         PRINT r.
ELSE
   CHECK IF d>0. and CALCULATE r1 = -b+Sqrt(d)/2*a
         r2 = -b-Sqrt(d)/2*a.
   TRUE, PRINT "The roots are real and unequal"
         PRINT r1 and r2
ELSE
   CALCULATE real = -b/2*a, d=-d and imag = Sqrt(d)/2*a
         r1 = real+imag and r2 = real-imag.
   PRINT, "The roots are not real and unequal"
   PRINT, r1 and r2
END IF
END.
```

Flow chart:



C Programming:

ALGOL.

BCPL - Basic computer programming language.

B.

Were the programing languages which is used before the C programme language.

In 1970's C programme language is started to develop in Bell lab.

In 1972's Dennis Ritchie finished the development of C Programme language.

1. C Programme is Portable. It doesn't require any specific operating Systems Like Linux.

2. C. Programme is structured and it is modularity

Modularity:

Modularity helps us to identify the errors in simplest way.

We can write the 30 lines of the programme in the modules.

3. Built in library files

4. Faster Execution.

Constant in C Programming:

We have two types of constant in C Programme They are.

1. Numeric Constant - 1. Integer
   2. Float.
2. Character constant - 1. Characaeter
   2. String

Numeric Constant:

Numeric constant has two types of constant they are

* Numeric Integer
* Float

Integer Constant: (int)

All the whole numbers are Integer constant Eg. 0, 1, 7, 8

Float constant: (Float)

Decimal numbers are the Float Constant. Eg. 2.1, 3.1

Character Constant: (char)

A single word represented in single Quote is called character constant.

Eg. 's', 'I', 'T'

1. Character constant has two types of Constant they are.
   * character
   * String.

String Constant:

A group of words represented in double Quote is called String constant.

Eg. "SIT".

Variable in C Programme.

Variable Names:

Syntax:

datatype Variable name.

Datatype:

If the data is given by the user it should be stored in datatype. It has many types of datatype. Example - Integer (int), Float (Float), Character (char) etc.

Rules to write the variable Name:

We have some certain rules to write variable name.

1. Variable name should start in underscore or character. For example. - a, _Sum
2. Should not give any space between the two variable names, instead of space we can use the underscore. Eg - Totalsum, Total_sum
3. Should not use any keyword before the variable name. Key words if, else, while

Structure of the C Programme

1. Documentation section - Details of the programme. It does not compile in the compilation. It is also known as Comment lines. If it is single line it is denoted by //, If it is multiple line /* ... */.

   Eg: c = a+b // Addition of two no.

2. Link Section - Header files is comes under the Link section. Header files link with the c-library.

3. Definition Section - To identify the definition Section it will starts with #define and after the Header file.

4. Global Declaration Section - We can access the variable anywhere from the programme it comes above the int main() It is called Global Declaration Section.

5. Main function - In main function the function or variable can be access within the main function. It is also called local Declaration section.

6. Sub-Function - The part of the function or Programme in the main function is called Sub-fuction.

3.10.23.

Basic 'c' Programmes.

```
# include <std io.h>
# include <conio.h>
```
} header files.

```
int main ( )
{
  Printf ("Sethu Institute of Technology");
  return 0;
}
```
→ Parameters / arguements

Write a C Programme to display you bio-data such as name, DOB, Year and Department.

```
#include <std io.h>
#include <con io.h>
int main ( )
{
  Printf ("Name : K.P.Vignesh \n");
  Printf ("DOB : 17-12-2004 \n");
  Printf ("Year : I st Year \n");
  Printf ("Department : ECE ");
  getch ( );
  return 0;
}
```

Output :

Name : K.P. Vignesh
DOB : 17-12-2004
Year : Ist Year
Department : ECE

2. Write a C Programme to add, sub, multiply and divide two numbers.

```
# include <stdio.h>
# include <conio.h>
int main ( );
{
  int a=50, b=20, c, d, e;
  float f;
  c = a+b;
  d = a-b;
  e = a*b;
  f = a/b;
  Printf ("The sum is : %d \n", c);
  Printf ("The sub is : %d \n", d);
  Printf ("The multiplication is : %d \n", e);
  Printf ("The division is : %f \n", f);
  getch ( );
  return 0;
}
```

Output

The Sum is : 70
The Sub is : 30
The multiplication is : 1000
The division is : 2.000

3 Write a C programme to find the Area of circle and Circumference of the circle.

```
#include <stdio.h>
#include <conio.h>
int main()
{
float PI = 3.14, r = 6, area, c;
area = PI * r * r;
c = 2 * PI * r;
Printf("Areat of the circle is : %f \n", area);
Printf("circumference of circle is : %f \n", c);
getch();
return 0;
}
```

Out put:

Area of circle is : 113.040001

Circumference of circle is : 37.68000

4 Write a C Programme to find the Simple interest


```
#include <stdio.h>
#include <conio.h>
int main()
{
float P=6, n=7, r=8, SI;
SI = P*n*r/100;
Printf("Simple Interest is : %f", SI);
getch();
return 0;
}
```

Out put

The Simple Interest is 3.360000

5/10/23

Write a C Programme for Bio-Data of a user. get the details from the user and Display.

```
#include <stdio.h>
#include <conio.h>
int main ()
{
 Char Name [30];
 int DOB ;
 Char Dept [30 ];
 Printf ("enter the name : ");
 Scanf ("%s", &Name);
 Printf ("enter the DOB :");
 Scanf ("%d", & DOB);
 Printf ("enter the Dept :");
 Scanf ("%s", & Dept );
 Printf ("Name is : %s \n", Name);
 Printf ("Date of Birth is : %d \n", DOB);
 Printf ("Department is : %s", Dept);
 Getch ();
 return 0 ;
}
```

Output :

```
Enter the Name : Vignesh K.P.
Enter the DOB : 17-12-2004.
Enter the Dept : ECE

Name is : Vignesh K.P.
Date of birth is : 17-12-2004
Department is : ECE.
```

2. Write a C Programme to find Area of circle and Circumference of the circle. get the value from the user.

```
#include <stdio.h>
#include <conio.h>
int main ()
{
 int r;
 Float area, circumference, PI=3.14 ;
 Printf ("enter the radius : ");
 Scanf ("%d", & r);
 area = PI * r * r;
 Circumference = 2 * PI * r;
 Printf ("Area of circle : %f \n", area);
 Printf ("circumference of circle : %f ", circumference);
 getch ();
 return 0;
}
```

Output :

Enter the value radius : 6.

Area of circle : 113.040001

Circumference of circle = 37.68000

Write a C Programme to adel, Subbract, Mulliply and divide two numbers, Get the value of user.

```
# include <Stdio.h>
# include <conio.h>
int main ()
{
 int a, b, c, d, e, ;
 float f ;
 Printf ("enter the value of a :");
 Scanf ("%d", & a);
 Printf ("enter the value of b :");
 Scanf ("%d", & b);
 c = a+b;
 d = a-b.;
 e = a*b;
 f = a/b.;
Printf ("The sum is : %d \n", c);
Printf ("The Sub is : %d \n", d);
Printf ("The multiplication is : %d \n", e);
```



```
Printf ("The division is : %f ", f);
getch ();
return 0;
}
```

Output :

enter the Value of a 15
enter the value of b 8

The Sum is = 23
The Sub is = 7
The multiplication is = 120
The division is = 1.875000

Write a C Programme to find the simple interest.

```
# include <stdio.h>
# include <conio.h>
int main ()
{
float P, n, r, SI ;
Printf ("enter the value of P :");
Scanf ("%f", & P);
Printf ("enter the value of n :");
Scanf ("%f", & n);
Printf ("enter the value of r :");
Scanf ("%f", & r);
SI = P* n* r/100;
Printf ("The simple interest is: %f", SI);
getch ();
return 0;
}
```

Output.

Enter the value of P = 6

Enter the value of n = 4

Enter the value of r = 8

The Simple Interest is = 0.418000

Operators:

1. Arithmetic operators :) +, -, x, /, %
2. Relational operator → <, >, <=, >=, !=
3. Logical operators :- && → Logical AND
   || → Logical OR
   ! → Logical NOT
4. Assignment operator (or) Equal to operator → =, +=, -=, *=, /=
5. Bitwise operator – & - AND
   | - OR
   ^ - Exclusive OR (XOR).
   ~ - complement
   << - Left shift.
   >> - Right shift

Managing Input and Output.

1. get char ( ); - Used to read a single character or group of character from the user.

   Variable name = getchar ( );

   eg - Char ch;

   ch = getchar ( );

2. Putchar ( ); → To display the character in the output.

   Putchar (variable name);

   eg - Char ch;

   ch = getchar;

   Putchar (ch);

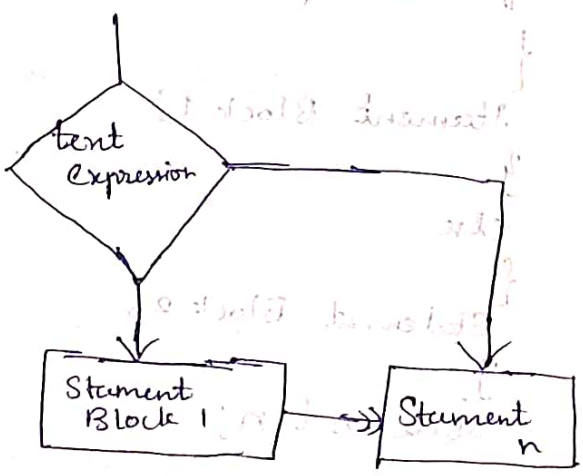
Unit-2. Decision Making and Looping Statement.

Decision Making:

1. If - Syntax.

```
If (Test expression);
{
 Statement Block 1;
}
 Statement n;
```

Flowchart:



1. Write a C Programme to check whether a number is Positive.

```
#include <stdio.h>
#include <conio.h>
#int main().
{
 int num;
 Printf ("enter the num");
 Scanf ("%d", &num);
 If (num>0)
 {
  Printf ("The number is positive");
 }
 Printf ("Thank you");
 getch();
 return 0;
}
```

Output 1:

Enter a number: 7.

The number is Positive.

Thank you

Output 2:

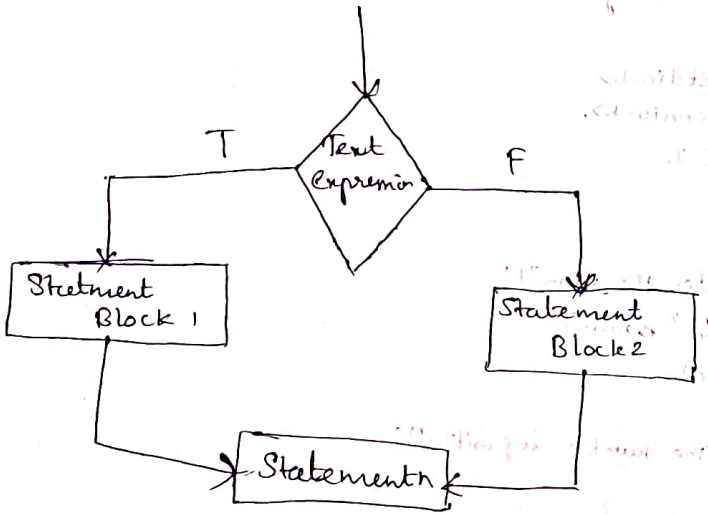
Enter a number : -6

Thank you.

2. If-else - Syntax.

```
if(test expression)
{
 Statement Block 1;
}
 else
{
 Statement Block 2;
 }
 Statement n;
```

Flow chart:



Write a C programme to check whether A number is positive or negative. ECPOSITI-C

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int num;
 Printf("Enter the number");
 Scanf("%d", &num);
 If(num>0)
 {
 Printf("The number is positive");
 }
 else
 {
 Printf("The number is negative");
 }
 Printf("Thank you");
 getch();
 return 0;
 }
```

Output 1:

Enter a number : 7.

The number is positive

Thank you.

Output 2:

Enter a number : -6

The number is negative

Thank you.

Write a C programme to check whether a number is odd or even.

```
#include <stdio.h>
# include <conio.h>
int main()
{
 int a;
 Printf ("Enter the value of a");
 Scanf ("%d", &a);
If (a%2==0)
{
 Printf ("The number is even");
}
else
{
 Printf ("The number is odd");
}
 Printf ("Thank you");
 getch ();
 return 0;
}
```

Output 1:

Enter the value of a = 2.
The number is even
Thank you

Output 2:

Enter the value of a = 3.
The number is odd
Thank you.

Write a C Programme to check whether a number is equal to zero.

```
#include <stdio.h>
# include <conio.h>
int main()
{
 int a;
 Printf ("Enter the value of a");
 Scanf ("%d", &a);
If (a==0)
{
 Printf ("The number is equal to zero");
}
else
{
 Printf ("The number is not equal to zero");
}
 Printf ("Thank you");
 getch ();
 return 0;
}
```

Output 1:

Enter the value of a = 0.
The number is equal to zero
Thank you.

Output 2:

Enter the value of a = 2
The number is not equal to zero
Thank you.

3. Else - If - Syntax.

```
if (test expression)
{
 Statment Block 1;
}
else if (test expression2)
{
 Statment Block 2;
}
else if (test expression 3)
{
 Statement Block 3;
}
else
{
 Statement 4;
}
Statement n;
```

Flow chart:

Write a C Programme to check whether a number is positive or negative or equal to zero.

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int a;
 Printf("Enter the no Number:");
 Scanf("%d", &a);
 if (a==0)
 {
  Printf("Number is equals to zero");
 }
 else if (a<0)
 {
  Printf("Number is negative");
 }
 else
 {
  Printf("Number is Positive");
 }
  Printf("Thank you");
  getch();
  return 0;
```

Output 1:

```
Enter a Number :5
The number is positive
Thank you
```

Output 2:

```
Enter a number :-5
The number is negative
Thank you
```

Output 3:

```
Enter a Number : 0
The number is
      equal to zero
Thank you.
```

4. Nested if -

Syntax -

```
if (test expression)
{
 if (test expression 2)
  {
   Statement Block 1;
  }
 else {
   {
   Statement Block 2;
   }
 }
else
 {
  if (test expression 3)
   {
   Statement Block 3;
   }
  else
   {
   Statement Block 4;
   }
 }
Statement n;
```

Write a C programme to find the largest of three Numbers.

```
# include <stdio.h>
# include <conio.h>
int main()
{
 int a, b, c;
 clrscr();
 Print ("Enter the value of a, b, c");
 Scanf ("%d \t %d \t %d", &a, &b, &c);
 if (a >= b)
  {
   if (a >= c)
    {
    Printf (" A is greater ");
    }
   else
    {
     Printf ("C is greater");
    }
  }
else
  {
   if (b >= c)
    {
     Printf ("b is greater");
    }
   else
    {
     Printf ("c is greater");
    }
  }
 Printf ("Thank You")
 getch();
```

Output 1:

Enter the value of a, b, c ; 7, 3, 1

A is greater.

Thank you.

5. Switch Statement.

Syntax ⇒ Switch (expression)

```
{
case value1:
    Statement 1;
    break;
case value 2:
    Statement 2;
    break;
default:
    Statement m;
}
Statement n;
```

write a C Programme to perform Arithmetic operation.

```
#include <stdio.h>
#include <conio.h>
int main()
{
Clrscr();
float a, b;
Char operator;
Printf("Enter the operator ; (+, -, *, /);");
Scanf("%s", &operator);
Printf("Enter the value of a and b");
Scanf("%f \t %f", &a, &b);
Switch (operator)
{
 Case '+':
    Printf("Addition is : %f \n", a+b);
    break;
 Case '-':
    Printf("Subraction is : %f \n", a-b);
    break;
 Case '*':
    Printf("Multiplication is : %f \n", a*b);
    break;

 Case '/':
    If (b != 0)
    {
     Printf("Division is : %f \n", a/b);
    }
    else
    {
     Printf("Error");
    }
```

```
        break;
Default:
    Printf("Invalid input");
}
Printf ("Thank You");
getch ();
return 0;
}
```

3.

Output.

Enter the operator (+, -, *, /): +.

Enter the value of a and b: 2, 4.

Addition is 6.

Write a C programme using switch case to display the days of weak.

```
#~~int main~~
#include <stdio.h>
#include <conio.h>
int main().
{
int a;
Printf("Enter the Value of a");
Scanf("%d", &a);
Switch (a)
{
Case '1':
    Printf("Monday");
```

6. Goto Statement

Syntax:

```
goto label;
    Statement 1;
label:
    Statement 2;
```

forward jump.

```
label:
    Statement 2;
goto label;
    Statement 1;
```

backward jump

Looping Statements

1. Entry Controlled loop
2. Exit controlled loop

Entry Controlled Loop:

1. While:

Syntax ⇒

```
while (condition)
{
Statements;
}
Next statement;
```

Write a C programme to print the numbers from 1 to 5 (while)

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i = 1;
while (i <=5)
{
Printf("Values are %d \n", i);
i++;
}
getch();
return 0;
}
```

Output:

```
Values are :1
Values are :2
Values are :3.
Values are :4.
Valuesare :5.
```

2. do while:

Syntax →

```
do
{
Statements;
}
while (condition);
Next statements;
```

Write a C programme to print the numbers from 1 to 5. (do while)

```
# include <stdio.h>
# include <conio.h>
int main ()
{
 int i = 1;
 do
 {
 Printf ("values are %d \n");
 i++;
 }
 while (i<=5);
 getch ();
 return 0;
}
```

Write a C programme to print sum of numbers from 1 to 5. using while

```
# include <Stdio.h>
# include <conio.h>
int main ()
{
 int i=1;
 Sum = 0;
 while (i<=5)
 {
 Sum = Sum+i;
 i++;
 }
 Print f("Sum of numbers from 1 to 5 = %d \n");
 getch ();
 return 0;
}
```

Write a C programme to print even number from 2 to 10 using do while.

```
# include <Stdio.h>
#include <conio.h>
int main ()
{
 int i=2;
 while (i<=10)
 {
 Printf ("The values are : %d \n";)
 i+=2;
 }
 getch ();
 return 0;
}
```

Output:

The values are : 2
The values are : 4
The values are : 6
The values are : 8
The values ar : 10.

19.10.23.

For loop.

Syntax => For (initialization, conditions, iteration)

Write a C programme using for loop to print the numbers from 1 to 5.

```
# include <stdio.h>
# include <conio.h>
int main ()
{
 int i;
 For (i=1 ; i<=5 ; i++)
  {
   Printf ("The value is : %d \n",i);
  }
  getch ();
  return 0;
}.
```

Write a C programme using for loop to print the even number from 1 to 10.

```
# include <stdio.h>
# include <conio.h>
int main ()
{
 int i;
 For (i=2 ; i<=10 ; i+=2)
  {
  Printf ("The values is : %d \n", i);
  }
  getch ();
  return 0;
}.
```

Output:

```
The value is : 2
The value is : 4.
The value is : 6.
The value is : 8.
The value is : 10.
```

Continue:

Syntax - continue;

Once a continue statement is encountered by a Compiler, control will jump up to the beginning of the loop.

Write a C programme to print the odd numbers from 1 to 10 using continue

```
# include <stdio.h>
# include <conio.h>
int main ()
{
 int i;
 For loop (i=1 ; i<=10 ; i+=2)
  {
   if (i%2 ==0)
    {
     continue;
    }
   else
    {
```

```
        Printf("The odd numbers is : %d \n", i);
    }
}
getch();
return 0;
}.
```

2.11.23.

UNIT - 3 - Arrays and Strings.

Arrays:

Collection of elements that belong to the same data type

General Syntax of Array:

datatype variable name [array size];

Eg - Float a [3];

Array size starts with zero and ends in n-1 value. For example the arrays size is [3] it starts with a[0], a[1], a[2] and ends in (n-1)

Initialisation of Array:

int a[2] = {10, 0};

(or)

inta[] = {10, 12};

Access an element in Array:

1. int a [5] = {10, 12, 13, 14, 15};

   Printf("%d", a[2]);

   O/P - 13.

2. To change an element.

   int a[5] = {10, 12, 13, 14, 15};

   a[2] = 17

   Printf("%d", a[2]);

   O/P - 17.

Loop in an array.

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int a[3] ={1, 2, 3};
 int i;
 for(i=0; i<=2; i++)
  {
  Printf("%d\n", a[i]);
  }
 getch();
 return 0;
}
```

Output :

```
1
2
3
```

Two Dimensional Array.

In the program two array size will be declared and it is called Two Dimensional Array.

Declaration :

```
int a[3][2];
```

First array size denotes the number of rows.
Second array size denotes the number of column.

Initialisation of Two Dimensional Array

```
int a[3][2] = {{123, 234, 326, 567, 721, 821}}
```

(or)

```
= {{123, 234}, {326, 567}, {721, 821}}
```

write a c program to initialites three rows and two columns. and to print three elements in the output screen.

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int a[3][2] = {{102,111},{912, 263},{121, 567}};
  For(int i=0; i<3; i++)
   {
    For(int j=0; j<2; j++)
     {
      Printf("Element of [%d][%d]", i, j);
      Printf("%d\n", a[i][j]);
     }
   }
   getch();
   return 0;
}
```

Output

```
Element of [0][0]   102
Element of [0][1]   111
Element of [1][0]   312
Element of [1][1]   263
Element of [2][0]   121
Element of [2][1]   567
```

String:

Syntax:

Char String_name[size];

Eg - Char a[10] = "Sethu";

(or)

Char a[ ];

Null datatype = '\0';

It defines the end of the string.

Declaration of string.

There are different ways to represent the string.

1. char a[10];
2. char a[10] = "sethu";
3. Char a[ ] = "Sethu";

Null character is not necessary while giving the group of character.

But it is necessary to give while giving single characters.

4. char a[50] = {'s','e','t','h','u','\0'};
5. Char a[ ] = {'s','e','t','h','u','\0'};

String functions and operation:

While writing string function in the program.

#include <string.h> header files should be included in the program.

Operation:

1. String length. - It defines How many character does the String has.

Function = Strlen ( );

Program = Char a[ ] = "Sethu";

Printf("%d", Strlen (a)); O/P - 5)

2. Function - Sizeof( ); - It will count the memory size of the function

Program: Char a[ ] = "Sethu";

Printf("%d", Sizeof(a)); O/p - 6.

3. Concatenate string: Joining of two string

function = Strcat ( );

Program - Char a[ ] = "Hello";

Char b[ ] = "world";

Printf("%s", Strcat (a, b));

O/p - Helloworld.

4. String copy: Used to a copy a string to another string.

function - Strcpy( );

Program:

```
Char a[ ] = "hello";
Char b[ ];
Strcpy (b,a);
Printf("%s", b);
```

O/p - hello.

Write a C program to find the length of a String.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
  Char a[];
  Printf("Enther the Character:");
  Scanf("%s", a);
  Printf("%s", a);
  Printf("The String length of the character :%d", Strlen(a));
  getch();
  return 0;
}
```

Output:

Enter the Character - Vignesh.

Vignesh

The string length of the character : 7

5. String compare - Comparing the two different String.

- (or) + number denote on the Ascii value of the String.

If the values are same meant, display 0.

If not, it display a + - number on Positive.

Function - Strcmp();

Program : char

20.11.23.

UNIT-4 - FUNCTIONS

Function:

Writting a program into small modules of the program.

* Function Decleration
* Function call
* Function Definition.

Function Decleration:

A function is ~~dot~~ declared before the int main and after the header files.

Syntax - return_type function_name (arguement_list);

Function call:

It is declared within the main function

Syntax - function_name (arguement_list);

Function Definition:

It is written separately after the main.

Syntax - return_type function_name (argument list)
{
Function body;
}

Write a C program using function to add two numbers.

```
#include <stdio.h>
#include <conio.h>
int add(int a, int b);
int main()
{
int a,b;
printf("Enter the values of a &b:");
scanf("%d%d",&a,&b);
Printf("The sum is %d",add(a,b));
getch();
return 0;
}
int add(int a,int b)
{
return a+b;
}
```

Output:

Enter the value of a &b: 2 3

The sum is : 5

Types of Function:

1. Library function
2. User Defined function

Function Prototype / ways of function call

1. Function without arguements & without return value

2. Write a c program using function to add two number without arguement & return value

```
#include <stdio.h>
#include <conio.h>
void add ( );
void main ( );
{
 add ( );
}
void add ( )
 {
  int a,b,c;
  Printf (" Enter two numbers :");
  Scanf ("%d \t %d", &a,&b);
  c = a+b.
  Printf ("The value of c is : ", &c);
 }
```

Output:

Enter two numbers : 2 3

The value of c is : 5

2. function without arguement & with return value

Write a c program using function to add two number without arguement & with return value

```
# include <stdio.h>
# include <conio.h>
int sum ( );
void
int main ( )
{
  int result;
  result = sum();
  Printf ("%d", result);
}
int sum ()
 {
  int a,b; c;
  Printf (" Enter the two number: ");
  Scanf ("%d \t %d", &a &b);
   c= a+b
  return c;
 }
```

3. Function with arguements and with return value

Write a c program using function to add two numbe with arguments & with return value

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

```
int a,b, result;
Printf ("Enter two numbers :");
Scanf ("%d %d", &a, &b);
result = sum(a,b)
Printf ("The sum is : %d", result);
}
int sum(int a, int b)
 {
  return a+b;
 }
```

(or)

```
#include <stdio.h>
# include <conio.h>
int sum (int a, int b);
void main ()
{
 int result;
 result = sum (a,b)
 Printf ("%d", result);
}
int sum(int a, int b)
 {
  int a,b;
  Printf ("Enter two numbers;");
  Scanf ("%d %d", &a, &b);
  return a+b;
 }
```

4. Function with argument and without return value

Write a C program using function to add two numbers, with argument and without return value

```
#include <stdio.h>
#include <conio.h>
 void sum (int a, int b);
 void main()
 {
  int result, a,b;
  Printf ("Enter two numbers: ");
  Scanf ("%d %d", &a, &b);
  sum(a,b);
 }
 Void sum (int a, int b)
  {
   Printf ("The sum is : %d", a+b);
  }
```

Parameter Passing Methods :

There are two ways of passing parameter of a function

* Call by Value
* Call by reference.

Call by Value :

1. Changes in actual argument does not affect the formal argument
2. Changes in Formal argument does not affect the Actual argument.

Write a c program to swap two numbers in function

```
#include <stdio.h>
#include <conio.h>
 int Swap (int x, int y);
 int main()
 {
  int x=5 , y=3;
  Swap(x, y); // Actual arguement //
  Printf("The numbers before swaping : %d\t%d", x, y);
  getch();
  return 0;
 }
 int Swap (int x, int y) // Formal arguement //
 {
  int temp = x ;
      x = y ;
      b = temp;
 }
```

Write a c program to display hellow world using function

```
#include <stdio.h>
#include <conio.h>
 Void greet()
 {
  Printf("Hello, world !\n");
 }
 {
  int main()
  {
  greet();
  getch();
  return 0;
 }
```

Output:

Hello, world !

Recursion / Recursive Function:

A function it call a function itself again and again is called Recursion

Write a c program to find Sum of range of natural numbers using recursion

```
#include <stdio.h>
#include <conio.h>
 int sum (int k);
 int main()
 {
  int result = Sum(10);
  Printf("%d", result)
  return 0;
 }
  int Sum (int k)
  {
   if (k>0)
   {
    return k + Sum(k-1);
    }
   else
    {
    return 0;
    }
```

29.11.23

UNIT-5 . POINTERS, STRUCTURES AND UNIONS

Structures :

Declaration :

Syntax - Struct - Structure name
```
{
datatype - member 1;
datatype member 2;
};
```

Access the structure. Accessed using dot operator (.)

Syntax - int main()
```
{
```
Syntax. - Struct Structure_name Struct_variable;
Struct variable . member 1 = 15;
Struct variable . member 2 = "S IT";
```
}
```

Write a C program using structure to display any three car name with its models and year of introduction.

```
#include <stdio.h>
#include <conio.h>
Struct car
{
 Char Car_name [20];
 Char Car_model [20];
 int year;
};
int main()
{
 Struct Car c1 = {"Tata", "Ace", 1980};
 Struct Car c2 = {"Audi", "A4", 2022};
 Struct Car c3 = {"Maruti", "Suv", 1999};
 Printf("Car1 is ;", c1.Carname, c1.Carmodel, c1.year);
 Printf("Car2 is ;", c2.Carname, c2.Carmodel, c2.year);
 Printf("car3 is ;", c3.Carname, c3.Carmodel, c3.year);
 getch();
 return 0;
}
```

Write a C program using Structure to call employee with stores, employee name, id, basic pay, HRA and DA as Members. Find the Totale pay

```
#include <stdio.h>
#include <conio.h>
Struct employee
{
 Char Employee_name[30];
 Char Employee_ID [30];
 int basic pay;
 float HRA;
 float DA;
};
int main()
{
 Struct Employee ;
 Printf("Enter employee name:");
```

```
Scanf ("%.s", & Employee);
Printf ("Employee ID : ");
Scanf ("%.s", & Employee1.employee_ID);
Printf ("Enter Basic Pay:");
Scanf ("%.d", & Employee1.Basic pay);
Printf (" HRA: ");
scanf ("%.f", & Employee1.HRA);
Printf ("DA :");
Scanf ("%f", & Employee1.DA);
float calculat Total pay (struct Employee 1)
 {
 return employee1.basic pay + employee1.HRA +
        employee1.DA;
 }
float Total pay = calculate totalpay (employee 1);
Printf (" Total pay for %.S (Employee ID:%8):
        %.f \n", employee.name, employee.employee_ID,
                 total pay);
getch();
return 0;
}
```

Union

Declaration:

Syntax - Union - Union name

```
{
datatype - member 1;
datatype_member 2:
};
```

Access the union

Syntax =

```
int main()
{
Union union_name Union_variable;
}
```

Write a program using union to display the number variable in the output

```
#include <stdio.h>
#include <conio.h>
Union. number.
 {
  int number;
 };
int main()
 {
  Union number a = 15;
  Printf ("Number is : %.d ; a.number);
  getch();
  return 0;
 }
```

Pointer:

Pointer is a variable which stores the addrss of another variable.

int var = 5; → ordinary variable

int *a ; → Pointer variable.

↓

Dereferencing operator.

a = &var;

Syntax → datatype * Pointer_Variable;

Wild pointer:

If we not assign any value or ~~valu~~ Variable to the pointer.

Null Pointer:

Always it is initialize with the value zero. *a=0;

Write a c program using union call student with nam, class, Roll no, total mark as as members Find the and the display the Grade of each student A < 80=A, < 60 = B, < 50 = C, > 50 = F

```
#include <stdio.h>
#include <conio.h>
union student
 {
  char name [50];
  char classname [20];
  int roll no;
  float total marks;
};
Char calculate Grade (float mark)
 {
  if (marks >=80)
   {
    return 'A';
   }
  else if (marks >=60)
    {
     return 'B';
    }
  else if (marks >=50)
    {
     return 'C';
    }
  else
    {
     return 'F';
    }
 }
int main()
 {
  union student_Stu;
  Printf (" Enter Student name: ");
  Scanf ("%s", &Stu.name);
  Printf (" Enter Class name : ");
  Scanf ("%s", &Stu.classname);
  Printf (" Enter roll number : ");
  Scanf ("%d", & Stu.rollno);
  Printf (" Enter total marks: ");
  Scanf ("%f", &Stu.totalmarks);
```

```
Char grade = calculate grade (Stu.total marks);
Printf ("Student Information: \n");
Printf ("Name: %s \n", Stu.name);
Printf ("class : %s \n", Stu.classname);
Printf ("Roll Number : %d \n", Stu.rollno);
Printf ("Total marks : %.f \n", Stu.total marks);
Printf ("Grade: %c \n", grade);
getch ();
return 0;
}
```

Pointer to pointer:

Pointer which stores the address of the another pointer.

```
int x=5, *P, **q
*P = &x
**q = &P
```

Write a C program using pointer to pointer concept to display the values of pointer variables.

```
#include <stdio.h>
#include <conio.h>
int main()
{
int x = 5, *P, **q;
   *P = &x;
  **q = &P;
Printf("The values are : %d \t %d \t %d \n", x, *P, **q);
getch ();
return 0;
}
```

Pointer to function.

Write a C program to swap two number using function by call by reference

```
#include <stdio.h>
#include <conio.h>
void swap (int *x, int *y);
{int temp = *x;
   *x = *y;
  *y = temp;
}
int main()
{
int a=5; b=10;
swap (&a, &b);
Printf ("a = %d , b = %d \n", a, b);
getch();
return 0;
}
```

Output:

a = 10 _ b = 5.

ynamic Memory Allocation.

Allocating the Memory space during the execution of the program

1. Malloc () → Memory Allocation
2. Calloc () → Contiguous Memory Allocation.
3. realloc () → Re-allocation.
4. Free () → De allocation

Malloc ()

Syntax Ptr = (type_cast*) malloc(byte-size);

Ptr → Pointer variable which hold the address of the first element.

Eg - Ptr = (int*) malloc (100 * sizeof (int));

Calloc ()

Multiple chunks of memory can be allocated.

Syntax Ptr = (type_cast*) calloc (n, element_size);

Eg n - number of elements.

element_size - Size of the elements

Eg - Ptr = (float*) calloc (25, (float*));

realloc ():

Re allocating the memory space without deleting the memory space

Syntax → Ptr = reallocate (Ptr, newsize);

free ():

De allocating the memory space (or) Freeing of the memory space.

It should declare before the getch () and return 0.