

Problem Solving and C Programming.

Unit - I - Introduction to C Programming

Basic Organization of Computer: ALU - Arithmetic Logical Unit

CPU

CU - Control Unit

I/P → ALU

O/P

+ CU

* Memory

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

Primary Memory:

* Primary memory is also called temporary memory or volatile memory (changeable memory).

* Primary memory is very expensive and it has limited storage 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 Read Only Memory.

EPROM - Erasable Programming Read Only Memory.

EEPROM - Electrically Erasable Programming Read only Memory.

Secondary memory:

and non-volatile memory.

- * It is cheaper as compare to the primary memory
- * It has unlimited capacity to store.

Q.P:

- * At first unit will convert the machine language to the user understandable language.

Problem formulation:

Logical thinking and Analysis:

Requirement

1. Problem analysis
 2. Requirement gathering
- we need to use problem solving tools after requirement

Problem Solving Tools:

- i) Algorithm
- ii) flowchart
- iii) Pseudocode

3. Development algorithm

- i) Draw flowchart
- ii. write Pseudocode
- iii. write the code.
- iv. Compilation and Execution
- v. Debug (if compilation of error).
- vi. Testing.
- vii. Maintenance
- viii. User Manual
- ix. Documentation

Algorithm: Step by step representation of the programme

Flowchart: pictorial representation of the programme

21/9/23

Characteristics of Algorithm:

- * Finiteness: countable
- * Definitions: It should be define clearly
- * Effectiveness: It should be clear and effectively
- * Generality: Algorithm should be written in general

Flowchart:

Symbol to draw or to represent the flow

what.

1. → Terminal → used for start and stop

2. → Input & Output → used for input of the

data and output of the data and used to initialize value of a variable

3. → Process → for processing the programme

and used to update the value of a variable

4. → Decision → To take a decision

5. → On Page Connector → To connect the page

6. → Off Page Connector → when we want finish the flowchart 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

* We always capitalise and then initial words.

* Write only one statement per line.

* Indent to show hierarchy of the programme

and to improve readability.

* While writing Pseudocode we want use the

Keywords - READ, WRITE, DISPLAY, PRINT, CALCULATE

(END, BEGIN, ENDIF, ENDWHILE,

ENDFOR).

Control Structures:

1. Sequence → The instruction should be in sequence 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 store them in C.

Step 3 : Add calculate the two values A and B.

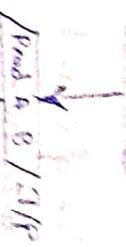
and store the result in C.

Step 4 : Display print the result stored in C.

Step 5 : Stop

Flow chart

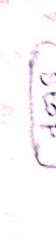
Start



[Initialize A & B] Process



Output C / STOP



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 on Algorithm and Pseudocode and draw a

Flowchart for sum of odd numbers.

Algorithm:

Step 1 : Start.

Step 2 : Read the value of R and Initiates the Value
of $\pi = 3.14$ from the user.

Algorithm:

Step 1 : Start.

Step 2 : Initiates the value of $\pi = 3.14$.

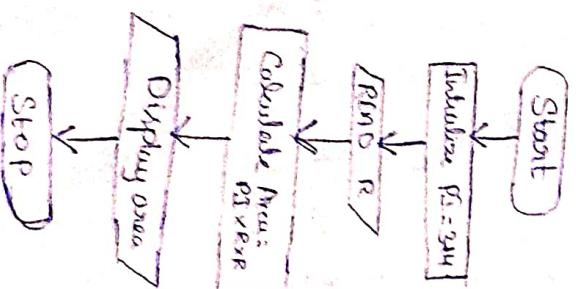
Step 3 : Read the value of R from the user.

Step 4 : calculate the area of circle is $\text{area} = \pi * R * R$

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.

249.23
Write and Algorithm , flowchart and Pseudocode for
calculate the Simple interest. $SI = \frac{P * n * r}{100}$

Algorithm:

Step 1 : Start

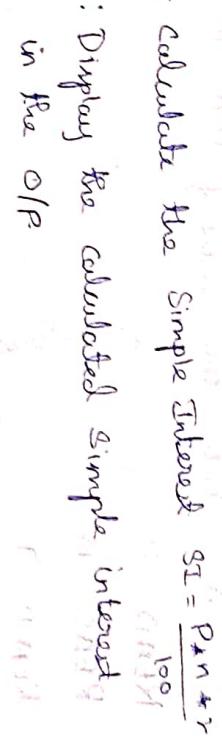
Step 2 : Read the Values of P,n and r from
the user.

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

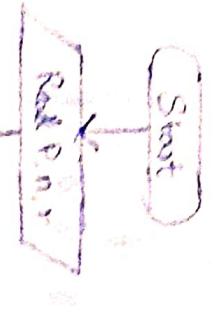
Step 4 : Display the calculated simple interest
in the O/P

Step 5 : Stop.

Flowchart:



Flowchart



Step 1 : Start

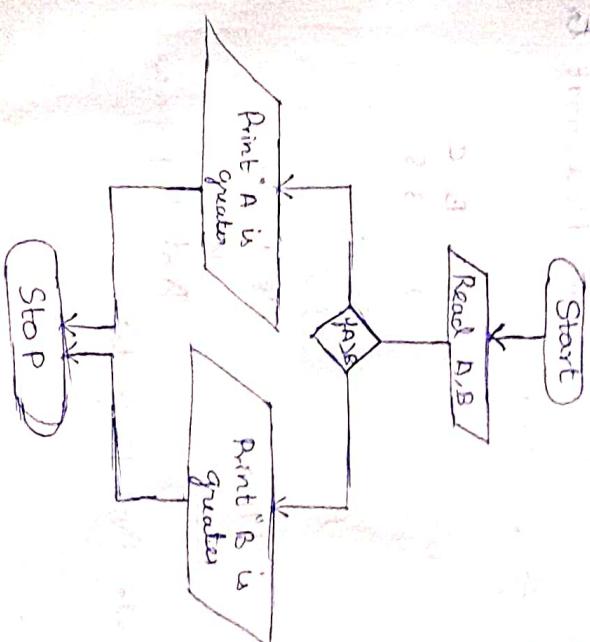
Step 2 : Read the values of A and B from the user.

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 P, n, r"
READ P
READ n
READ r
READ T
CALCULATE ST = P+n+r
PRINT ST
100
END
  
```

* Selection Control Structures
 will be on Algorithm , Pseudo code and flowchart
 to find the biggest of two numbers

Pseudocode.

BEGIN.

DISPLAY "Enter the value of A and B"

READ A

READ B

IF ($A > B$)

TRUE, PRINT "A is greater"

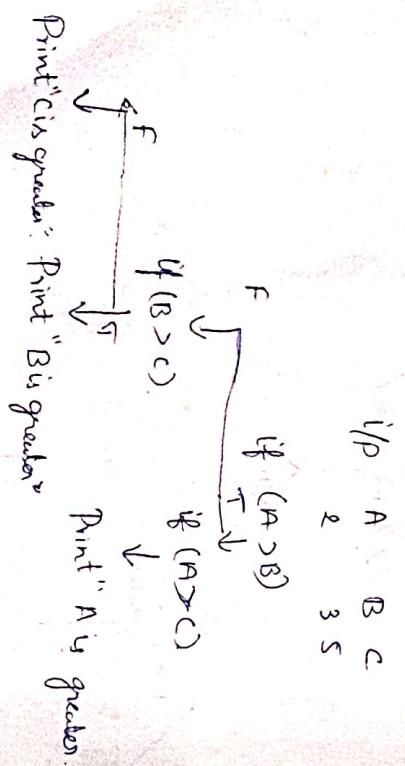
ELSE

PRINT "B is greater"

ENDIF

END.

Write an algorithm, Pseudocode and flowchart to find the greatest of three numbers.



Print "C is greater" Print "B is greater"

Print "A is greater".

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".

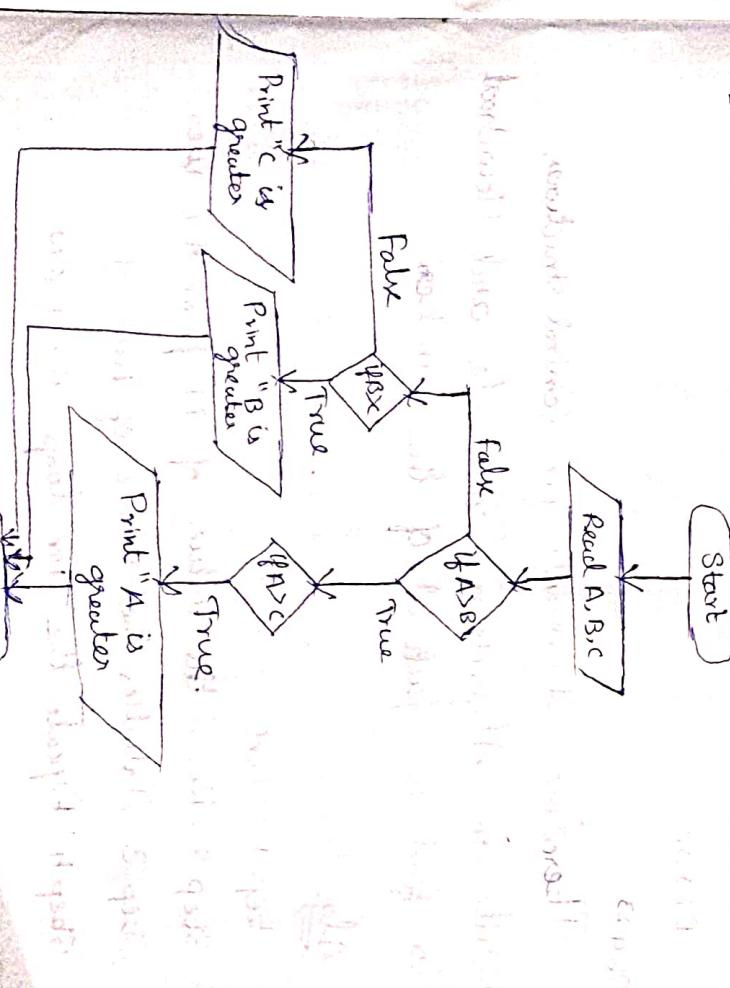
Step 5 : If ($A > B$) is false , check the condition if ($B > C$).

Step 6 : If step 5 is true Print "B is greater".

Else , Print "C is greater".

Step 7 : Stop.

Flowchart:



Pseudo code:

DISPLAY "Enter the values of A,B,C"

READ A

READ B

READ C

IF (A>B)

 true

 ELSE

 true

 ENDIF

 true

 ELSE

 PRINT "C is greater"

 ENDIF

END.

26923
Iterative statements (or) control structures.

Write an Algorithm, Pseudo code and Flowchart to find the factorial of the Numbers.

Algorithm:

Step 1 : Start

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

Step 3 : Initialize the value of Fact = 1.

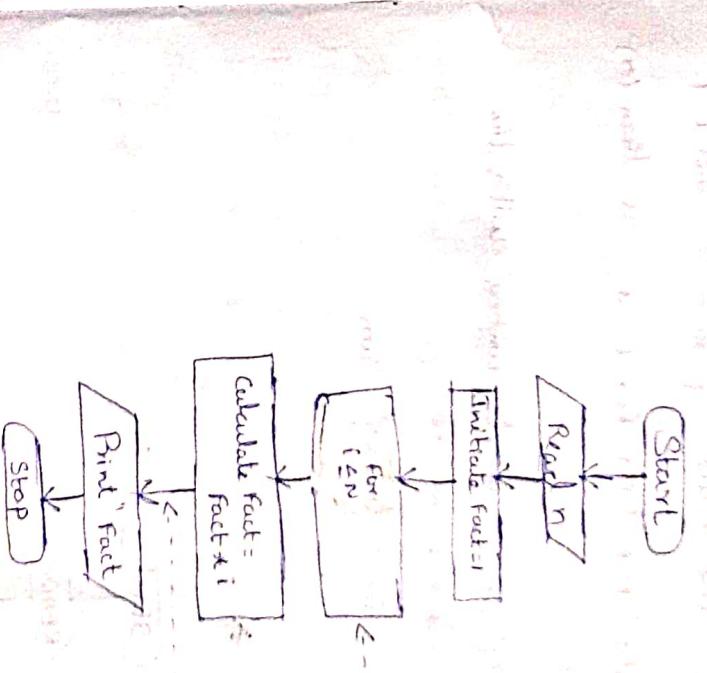
Step 4 : Repete the for loop for $i \leq N$.

Step 5 : Calculate the new factorial value with the formula $Fact = Fact * i$

Step 6 : Display the value of fact

Step 7 : Stop

Flow chart



Pseudo code:

BEGIN
DISPLAY "Enter the value of N"
READ N.

INITIALES fact = 1

REPEAT FOR $i \leq N$

 CALCULATE fact = fact * i

END FOR.

PRINT 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.

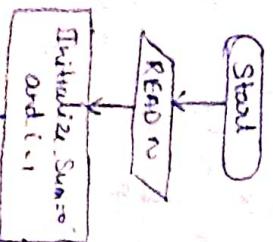
Step 4 : Repeat the For loop For i is less than (or) equal to n.

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

Step 6 : Display the value of sum

Step 7 : Stop.

Flowchart:



Pseudocode:

BEGIN

DISPLAY "Enter the value of n"

READ n

INITIALIZE sum=0 and i=1

REPEAT FOR i<n
CONCERNTE 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.

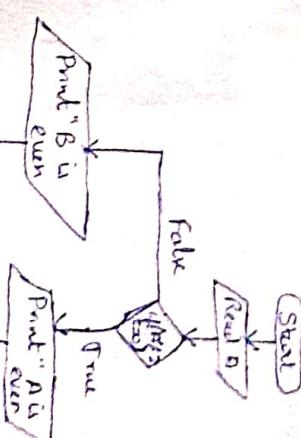
Step 3 : Check the condition if ($A \% 2 == 0$)

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

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

Step 6 : Stop.

Flow chart:



Pseudo code

BEGIN

DISPLAY "Enter the value of n"

READ n

IF (n >= 0)

 TRUE, PRINT "n is even"

ELSE

 PRINT "n is odd"

END IF

END.

Note : 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 : Increment the value of i

Step 7 : Stop.

Flowchart

Start

Read n

Initialize
 $i = 1$

Print i

Increment
 $i = i + 1$

Stop

Pseudocode

BEGIN

DISPLAY "Enter the value of n"

READ n

INITIALIZE i = 1

REPEAT WHILE $i \leq n$

 PRINT i

 INCREMENT i = i + 1

END WHILE

END.

Write an Algorithm, Pseudocode and flowchart 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 $(B=0)$
- 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 step 6 is true, Print "The roots are real and equal".
- Step-8 Else, 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 step 10 is false, check if else.
- Print "The roots are not real and unequal".
- Step 15 - calculate $\text{real} = -b/2+a$, $d = -d$ and $\text{imag} = \sqrt{|d|}/2+a$.
- Step 16 - Print r_1 and r_2 .
- Step 17 - Stop.

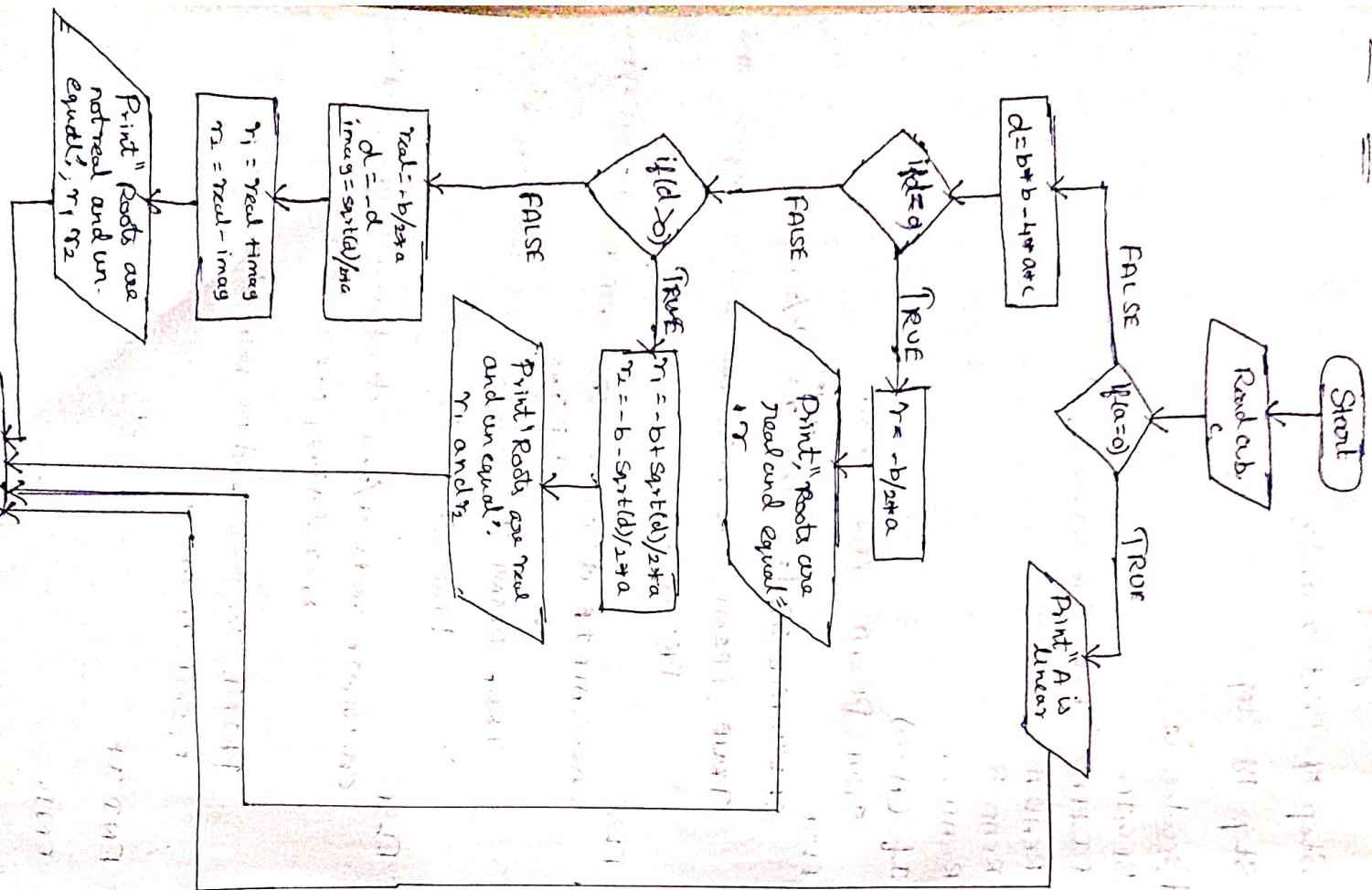
Pseudocode:

```

Step 1. 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 r = -b/2+a.
        TRUE, PRINT "The roots are real and equal" AND
        PRINT r
        . . . 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 - used with mainframe computers
BCPL - Basic computer programming language

B

were the programming language which is used.

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

In 1972/3 Dennis Ritchie finished the

development of a programme language.

1. C Programme is Portable. It doesn't require any specific operating systems like Linux.

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 always old data in memory address to change them. They are.~~

1. Numeric Constant - 1. Integer 2. Float.
2. character constant - 1. character 2. String

Numerical constants

Numerical Constant has two types of constant.

they are

+ Ramanic Sutras

四
卷之三

Intergen Compendium

All the triangle numbers are single.

Float content: float content = $\frac{\text{volume of float}}{\text{volume of concrete}}$

Constant - Eq. 2.43.1

Character constant: (char)

2. A single word represented in single quote

in valley *inuncta* constant.

卷之三

Character constant has two types of

constant tiny add.

String.

String constant

A group of words represented in double quotes

is called string constant.

Fig. 1. Schematic diagram of the experimental setup.

Variable in Engineering

Syndrome d'Angioedème

Deltotis ventralis var.

datatype - data type is given by the user it should be stored in datatype. It has many types like integer (int), float (float) or character (char) etc.

Rules to write the variable name

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 something
the variable name. Key words like 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.

Eg: $c = a + b // \text{Addition}$
of two
no.

If it is single line it is denoted by $//$, If it is multiple line $/* */$.

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-function.

Basic 'c' Programmes.

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

int main () { Parameters / arguments

{ }

Printf ("Sethu Institute of Technology");

return 0;

}

Write a c programme to display your bio-data such as name, DOB, Year and Department.

#include <std io.h>
#include <stdio.h>

int main ()

{ printf ("Name : K.P.Vignesh\n");

printf ("DOB : 17-12-2004\n");

printf ("Year : Ist Year\n");

printf ("Department : ECE\n");
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, f;

float g;

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;
```

Output shown below

```
area = 113.040000
c = 37.680000
```

Output shown below

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

```
float PI=3.14 , n=1 , r=8, SI ;
SI = PI * n * r * r / 100;
```

```
printf("Simple Interest is : %f", SI);
```

```
getch();
```

```
return 0;
```

```
Output shown below
```

```
The Simple Interest is 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

4 Write a C Programme to find the simple interest

Output shown below

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

```
int main()
```

```
float P=50 , n=1 , r=8, SI ;
```

```
SI = P * n * r / 100;
```

```
printf("Simple Interest is : %f", SI);
```

```
getch();
```

```
return 0;
```

```
Output shown below
```

```
The Simple Interest is 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

```
Output shown below
```

```
Interest = 3.360000
```

Write a C programme for Bio-Datal 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 : ");

Scant ("%s", &name);

Printf ("Enter the DOB : ");

Scant ("%d", &DOB);

Printf ("Enter the Dept : ");

Scant ("%s", &Dept);

Printf ("Name is : %s\n", name);

Printf ("Date of birth is : %d-%d-%d", DOB);

Printf ("Department is : %s", Dept);

getch ();

return 0;

Q.

Output:

Enter the Name : Vignesh K.P.

Enter the DOB : 17-12-2004.

Enter the Dept : ECE

Concole output window

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 values from the user.

include <stdio.h>

include <conio.h>

int main ()

{

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 is %f", circumference);

getch ();

return 0;

Q. : Is it possible to do this without using PI?

Output:

Enter the surface radius : 6.

Area of concrete : 113.84 m²

~~circumference of circle = 37.68000~~

Write a C programme to add, subtract, multiply and divide two numbers.

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

1. *Leucosia* *leucosia* Linné

وَمِنْهُمْ مَنْ يَرْجُوا
كَلَّا إِنَّمَا يَرْجُوا
الْحَيَاةَ الدُّنْيَا

Printf prints the value of a : ");

Scanf ("%.d", &obj);
scanf ("%c", &key);

printf ("enter the value of b: ");

$$C = \text{atb}; \quad \text{atb} = 15000$$

$$d = a - b; \quad$$

$f = a/b$, where a and b are integers.

Point 7 (the sum is 1.4. d m, c);

Printf ("The sub is : %d\n", d); // the result

```
Printf ("The multiplication is : %.d \n", e);
```

```
printf("The division is : %f", f);
```

return 0;

~~✓ J. 1619~~

Enter the value of a
Enter the value of b

The sum is = 23
The sub is = 7
The multiplication is = 160
The division is = 1.87500
write a programme to find the simple interest

```
#include <stdio.h>
#include <conio.h>
int main()
{
    clrscr();
    cout<<"Hello World";
}
```

float p~~as~~, n, r, si
do -
n - value of standard

```
printf("Enter the value of p : ");
```

```
printf("Enter the value of n:");
```

```
scanf("%f", &r);
```

Scorff (Mörf, Schorr) ist so gewandt

Rinty ("The simple interest is 5%", SI).

Output

2. Put char ('); → To display the character in ${}^{\text{37}}$

Enter the value of $P = 6$

enter the value of $n = 11$

The Simple Interest is \$226.00 - \$180.00 =

Operations

1. Arithmetic operators $\Rightarrow +, -, \times, /, \div$
 2. Relational operator $\rightarrow <, >, \leq, \geq, !=$
 3. Logical operators $\wedge \& \rightarrow$ Logical AND

1

Assignment operator
(var).
Equal to operator

S. Bitwise operator - & - AND

operator - & - AND
 || - OR \oplus (XOR) exclusive OR
 ^ - Exclusive OR (XOR)
 ~ - complement (NOT)
 << - Left shift (L.) bits
 >> - Right shift (R.) bits

Managing Input and Output

1. `getchar();` - used to get a single character or group of characters from the screen.

14

variable name = getchar();
eg - char ch;
ch = getchar();

the output

getchar rec.
ch = *getchar*;
Putchar(ch);

Unit-2. Decision Making and Looping Statement.

Decision Making:

1. If - Syntax.

```
If (Test expression);
```

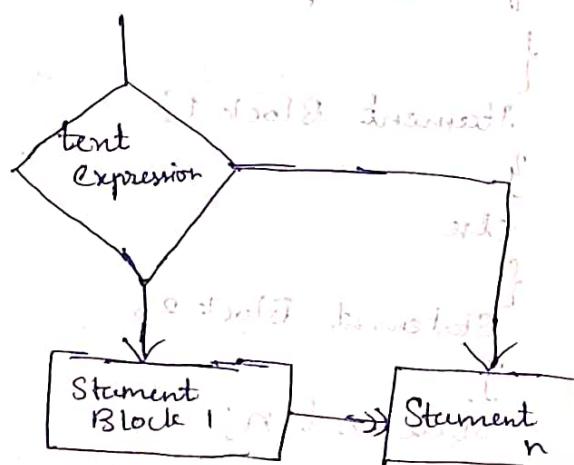
```
{
```

```
Statement Block 1;
```

```
}
```

```
Statement n;
```

Flowchart:



I. 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;
}
```


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.

{ (text expression)

} Statement Block 1;

3. else if (text expression2);

} Statement Block 2;

} else if (text expression3);

} Statement Block 3;

} else {

} Statement 4;

} Statement 5;

} Statement 6;

} Statement 7;

} Statement 8;

} Statement 9;

} Statement 10;

} Statement 11;

} Statement 12;

} Statement 13;

} Statement 14;

} Statement 15;

} Statement 16;

} Statement 17;

} Statement 18;

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 equal 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

Thankyou

Output 3:

Enter a number : 0

The number is equal to zero

Thank you

4. Nested if

Syntax - if (test expression)

{ Statement Block 1 }

else { Statement Block 2 }

if (test expression 2)

{ Statement Block 3 }

else { Statement Block 4 }

if (test expression 3)

{ Statement Block 5 }

else { Statement Block 6 }

else { Statement Block 7 }

else { Statement Block 8 }

else { Statement Block 9 }

else { Statement Block 10 }

else { Statement Block 11 }

else { Statement Block 12 }

else { Statement Block 13 }

else { Statement Block 14 }

else { Statement Block 15 }

else { Statement Block 16 }

else { Statement Block 17 }

else { Statement Block 18 }

else { Statement Block 19 }

else { Statement Block 20 }

else { Statement Block 21 }

else { Statement Block 22 }

else { Statement Block 23 }

else { Statement Block 24 }

else { Statement Block 25 }

else { Statement Block 26 }

else { Statement Block 27 }

else { Statement Block 28 }

else { Statement Block 29 }

else { Statement Block 30 }

else { Statement Block 31 }

else { Statement Block 32 }

else { Statement Block 33 }

else { Statement Block 34 }

else { Statement Block 35 }

else { Statement Block 36 }

else { Statement Block 37 }

else { Statement Block 38 }

else { Statement Block 39 }

else { Statement Block 40 }

else { Statement Block 41 }

else { Statement Block 42 }

else { Statement Block 43 }

else { Statement Block 44 }

else { Statement Block 45 }

else { Statement Block 46 }

else { Statement Block 47 }

else { Statement Block 48 }

else { Statement Block 49 }

else { Statement Block 50 }

else { Statement Block 51 }

else { Statement Block 52 }

else { Statement Block 53 }

else { Statement Block 54 }

else { Statement Block 55 }

else { Statement Block 56 }

else { Statement Block 57 }

else { Statement Block 58 }

else { Statement Block 59 }

else { Statement Block 60 }

else { Statement Block 61 }

else { Statement Block 62 }

else { Statement Block 63 }

else { Statement Block 64 }

else { Statement Block 65 }

else { Statement Block 66 }

else { Statement Block 67 }

else { Statement Block 68 }

else { Statement Block 69 }

else { Statement Block 70 }

else { Statement Block 71 }

else { Statement Block 72 }

else { Statement Block 73 }

else { Statement Block 74 }

else { Statement Block 75 }

else { Statement Block 76 }

else { Statement Block 77 }

else { Statement Block 78 }

else { Statement Block 79 }

else { Statement Block 80 }

else { Statement Block 81 }

else { Statement Block 82 }

else { Statement Block 83 }

else { Statement Block 84 }

else { Statement Block 85 }

else { Statement Block 86 }

else { Statement Block 87 }

else { Statement Block 88 }

else { Statement Block 89 }

else { Statement Block 90 }

else { Statement Block 91 }

else { Statement Block 92 }

else { Statement Block 93 }

else { Statement Block 94 }

else { Statement Block 95 }

else { Statement Block 96 }

else { Statement Block 97 }

else { Statement Block 98 }

else { Statement Block 99 }

else { Statement Block 100 }

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

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a,b,c;
    clrscr();
    printf("Enter the value of a,b,c");
    scanf("%d%d%d",&a,&b,&c);
    if (a>b)
        if (a>c)
            printf("a is greater");
        else
            if (b>c)
                printf("b is greater");
            else
                printf("c is greater");
    else
        if (c>b)
            printf("c is greater");
        else
            printf("b is greater");
}
```

Output

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

A is greater.

Thank you.

5. Switch Statement.

Syntax ⇒ Switch (Expression)

```
{ case value1:  
    Statement1;  
    break;  
case value2:  
    Statement2;  
    break;  
default:
```

```
Statementm;  
} Statement n;
```

```
case value1:  
    Statement1;  
    break;
```

```
case value2:  
    Statement2;  
    break;
```

```
case value1:  
    Statement1;  
    break;
```

```
case value2:  
    Statement2;  
    break;
```

```
case value1:  
    Statement1;  
    break;
```

```
case value2:  
    Statement2;  
    break;
```

```
case value1:  
    Statement1;  
    break;
```

```
case value2:  
    Statement2;  
    break;
```

```
case value1:  
    Statement1;  
    break;
```

```
case value2:  
    Statement2;  
    break;
```

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/%f", &a, &b);  
    switch (operator)  
    {  
        case '+':  
            printf("Addition is : %f\n", a+b);  
            break;  
        case '-':  
            printf(" Subtraction 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");  
            }  
    }  
}
```

Default:

```
    break  
    printf("Invalid input");  
}  
  
printf("Thank You");  
getch();  
return 0;
```

3.

Output.

Enter the operator (+, -, *, /) in the prompt:

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

Addition is 6.

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

```
#include <conio.h>  
#include <stdio.h>  
int main().  
{  
    int a;  
    printf("Enter the Value of a");  
    scanf("%d", &a);  
    switch(a){  
        case "1":  
            printf("Monday");  
        case "2":  
            printf("Tuesday");  
        case "3":  
            printf("Wednesday");  
        case "4":  
            printf("Thursday");  
        case "5":  
            printf("Friday");  
        case "6":  
            printf("Saturday");  
        case "7":  
            printf("Sunday");  
    }  
}
```

6. goto Statement

Syntax :

 goto Label;

Statement 1:] forward jump.

Label : ←

Statement 2:

Label : ←
Statement 2:

Label : ←
Statement 2:
Statement 1:] backward jump

Looping Statements

1. Entry controlled loop
2. Exit controlled loop

Entry controlled Loop:

1. while :

Syntax \Rightarrow while (condition)
 {
 Statements;
 }

Next statement;

Write a 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

2. do while:

Syntax \Rightarrow do
 {
 Statements;
 }

while (condition);
Next statements;

1. do while

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

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

int main ()
{
    int i = 1;
    do
    {
        printf ("values are %d\n", i);
        i++;
    } while (i <= 5);
    getch ();
    return 0;
}
```

Output:

1

2

3

4

5

write a C programme to print even numbers 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);
        i += 2;
    }
    getch ();
    return 0;
}
```

Output:

2

4

6

8

10

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;
    int sum = 0;
    while (i <= 5)
    {
        sum = sum + i;
        i++;
    }
    getch ();
    return 0;
}
```

Output:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

For loop.

Syntax \Rightarrow For (initialization, condition, iteration)

g.

Output:

The value is : 2
The value is : 4.

The value is : 6.
The value is : 8.

The value is : 10.

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;
}
```

Continue:

Syntax - Continue:
Once a continue statement is encountered by a compiler, it will jump up to the beginning of the loop.

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

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

int main ()
{
    int i;
    for (i=1 ; i<=10 ; i+=2)
    {
        if (i%2 == 0)
            continue;
        else
            printf ("The values is : %d\n",i);
    }
    getch();
    return 0;
}
```

Write a C programme using for loop to print the even numbers 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;
}
```

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 array size is [3] it starts with a[0], a[1], a[2] and ends in a[1].

Initialisation of Array:

int a[2] = {10, 12};
(or)
int a[] = {10, 12};

Access an element in Array:

1. int a [5] = {10, 12, 13, 14, 15};
printf("%d", a[2]); // a[2] is the third element
Output : 13

2. To print out

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

a[2] = 17; // change the
// second element to 17
printf("%d", a[2]); // prints 17

Loop is an array

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

```
int main() { int i, j; for (i = 0; i < n; i++) for (j = 0; j < m; j++) cout << a[i][j]; }
```

```
int a[3] = {1, 2, 3};  
int i;
```

```
for (i=0 ; i<2 ; i++)
```

```
printf("%d\n", a[i]);
```

get ch c)

return 0;

ج

$\omega_N -$

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 columns.

Initialization of Two Dimensional Array

$$\text{int } \alpha [\beta] \beta = \{ 123, 234, 326, 567, 721, 821 \}$$

Write a C program to indicate three rows and two columns. and to print those elements in the output screen.

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

```
int main()
{
    int a[3][2] = {{102, 113}, {912, 263}, {121, 567}};
    cout << a[1][0];
}
```

```
For(int i=0 ; i<3 { i++})
```

```
for (int j=0 ; j<2 ; j++)
```

```
Printf("Element of [r:d] [r:d]", i,j);
```

Gleich drin;

T 3

ଓন্দৰা পত্ৰ ১৮৩

Second array size denotes the number of rows.

卷之三

String :

Syntax :

Char string - name [size];

Eg - char a[10] = "sethu";
(or)
char a[10];

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[5] = {'s', 'e', 't', 'h', 'u', '\0'};

While writing string function in the program,
#include <string.h> header file should be included in the program.

Operation :

1. String length - It defines how many character does the string have.
2. Function - strlen () ;

Program: char a[] = "sethu";
printf("%d", strlen(a)); Output - 5

Program: char a[] = "sethu";
printf("%d", sizeof(a)); Output - 6

3. Concatenate string : Joining of two strings

function - strcat ();

Program - char a[] = "Hello";

char b[] = "world";

printf("%s", strcat(a, b));

Output - Helloworld.

→ A string copy is used to copy a string to another string.

function - strcpy();

Program: char a[] = "Hello";

char b[];

strcpy(b, a);

printf("%s", b);

Output - Hello.

String functions and operation :

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

#include <stdio.h>
#include <conio.h>
#include <string.h>

int main()

```
{
```

char a[5];

printf("Enter the character :");

scanf("%c", a);

printf("%c", 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 strings.

- (a) If number double. If the values are same meant i.e. on the ASCII value of display 0. If not, it displays a - number in pure.

Function- strcmp()

Program 1: char

String compare

String

- (b) If number double. If the values are same meant i.e. on the ASCII value of display 0. If not, it displays a - number in pure.

UNIT-4 - FUNCTIONS

function:

writing a program into small modules

of the program.

- * Function Declaration.
- * Function call
- * Function Definition.

Function Declaration:

A function is declared before the int main

and after the header files.

Syntax - return-type function-name (argument-list);

Function call:

It is declared within the main function

Syntax - function-name (argument-list);

Function Definition:

It is written separately after the main.

Syntax - return-type function-name (argument-list)

{

 Function body;

}

write a program using function to add two numbers.

```
#include <stdio.h>
#include <conio.h>
int add(int a, int b)
{
    int c;
    c = a + b;
    printf("The sum is %d\n", c);
    getch();
}
```

3

```
int add (int a,int b)
```

```
{
```

```
    return a+b;
```

}

Output :

Enter the value of a & b: 2 3

The sum is 5

Press any key to continue

Types of function:

1. Library function
2. User Defined function

Function

Function Prototype / ways of function call

2. function without argument & with return value

Q. Function without arguments & without return value

Write a C program using function to add two numbers without argument & return value.

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

void add ();
void main ()
{
    clrscr();
    add();
}

void add ()
{
    int a,b,c;
    printf (" Enter two numbers : ");
    Scanf ("%d %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
```

3. Function with arguments and with return value

Write a C program using function to add two numbers with arguments & with return value

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

int num (int a,int b);

void main()
```

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

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

int sum ();
void main ()
{
    int result;
    result = sum();
    printf ("%d", result);
}
```

```
int sum()
{
    int a,b,c;
    printf (" Enter the two numbers : ");
    Scanf ("%d %d", &a, &b);
    c = a+b;
    return c;
}
```

3. Function with arguments and with return value

4. Function with argument and without return value

```
int a,b, result;
printf("Enter two numbers : ");
scanf("%d %d", &a, &b);
result = sum(a,b);
printf("The sum is : %d", result);
```

```
int num(int a, int b)
{
    int sum;
    sum = a+b;
    return sum;
}
```

```
int a,b;
main()
{
    printf("Enter two numbers : ");
    scanf("%d %d", &a, &b);
    result = num(a,b);
    printf("The sum is : %d", result);
}
```

```
#include <stdio.h>
int sum (int a,int b)
{
    int result;
    result = a+b;
    return result;
}
```

```
#include <conio.h>
int sum (int a,int b);
void main ()
{
    int result;
    result = sum (10,20);
    printf("The sum is : %d", result);
}
```

```
int sum(int a, int b)
{
    int result;
    result = a+b;
    return result;
}
```

```
#include <stdio.h>
int sum (int a,int b);
void main ()
{
    int result;
    result = sum (10,20);
    printf("The sum is : %d", result);
}
```

```
#include <conio.h>
int sum (int a,int b);
void main ()
{
    int result;
    result = sum (10,20);
    printf("The sum is : %d", result);
}
```

write a c program using function to add two numbers, with argument and without return value

#include <stdio.h>

Void sum (int a,int b);

```
Void sum (int a,int b)
{
    int result;
    result = a+b;
    printf("The sum is : %d", result);
}
```

```
Void sum (int a,int b)
{
    int result;
    result = a+b;
    printf("The sum is : %d", result);
}
```

```
Void sum (int a,int b)
{
    int result;
    result = a+b;
    printf("The sum is : %d", result);
}
```

```
Void sum (int a,int b)
{
    int result;
    result = a+b;
    printf("The sum is : %d", result);
}
```

```
Void sum (int a,int b)
{
    int result;
    result = a+b;
    printf("The sum is : %d", result);
}
```

```
Void sum (int a,int b)
{
    int result;
    result = a+b;
    printf("The sum is : %d", result);
}
```

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. Change in formal argument does not affect the actual argument.

Write a C program to Swap two numbers

#include <stdio.h>

#include <conio.h>

int Swap (int x, int y);

int main()

{

int x=5, y=3;

Swap(x, y); //Actual argument //

Print("The numbers before swapping : %d %d", x, y);

getchar(); //function exits after printing the output

return 0;

}

int Swap (int x, int y) //Formal argument//

{

int temp = x;

x = y;

y = temp;

Write a C program to display Hello world using function

#include <stdio.h>

#include <conio.h>

Void putc (

{

Print("Hello, World ! \n");

int main ()

{

getchar();

Output:

Hello , world !

Recursion / Recursive Function.

A function which calls itself again and again is called Recursion

write a C program to find Sum of range of natural numbers using recursion

#include <stdio.h>

int sum (int k);

int main()

{

int result = sum(10);

Print("%d", result);

return 0;

}

int sum (int k)

{

if (k>0)

{

return k + sum(k-1);

else

{

return 0;

}

JNTU-S . POINTERS, STRUCTURES AND UNIONS

Structures :

Declaration :

Syntax - Struct - Structure-name

```
struct car {
    char carname[20];
    int carmodel;
    int year;
};

struct car c1 = {"Tata", "A", 1980};
struct car c2 = {"Audi", "A", 2022};
struct car c3 = {"Maruti", "Suv", 1999};
```

Access the structure. Accessing dot operator (.)

Syntax - int main()

Syntax - Struct structure_name Struct - variable;

Struct variable . member;

```
Struct variable . member 2 = "SIT";
```

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

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

struct car {
    char carname[20];
    int carmodel;
    int year;
};

char carname[20];
int carmodel;
int year;
```

Write a C program using structure to call employee with name, employee name, basic pay, HRA and DA as members. Find the Total pay

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

Struct employee
```

```
char EmployeeName[30];
char EmployeeID[30];
int basicPay;
float HRA;
float DA;
int main()
```

```
Struct Employee;
printf("Enter employee name :");
```

```
Scantf (" .s ", &Employee);
```

```
Printf (" Employee ID : ");
```

```
Scantf (" .d ", &Employee.employeID);
```

```
Printf (" Enter Basic Pay : ");
```

```
Scantf (" .f ", &Employee.basicPay);
```

```
Printf (" HRA : ");
```

```
Scantf (" .f ", &Employee.HRA);
```

```
Printf (" DA : ");
```

```
Scantf (" .f ", &Employee.DA);
```

```
float calculateTotalPay (struct Employee l)
```

```
{
```

```
    return employee1.basicPay + employee1.HRA +
```

```
    employee1.DA;
```

```
3
```

```
float totalPay = calculateTotalPay (employee1);
```

```
Printf (" Total Pay for %s (Employee ID %d) : 
```

```
%f\n", employee.name, employee.employeID,
```

```
totalPay);
```

```
getch();
```

```
return 0;
```

```
/* Output: Employee name is "Raj"
```

```
Employee ID is 1000
```

```
Basic Pay is 10000
```

```
HRA is 2000
```

```
DA is 1000
```

```
Total Pay is 13000
```

```
Press any key to continue . . .
```

Union Declaration :

Syntax - Union - Union name

{
datatype - member 1;
datatype - member 2;

};

datatype - member 3;

datatype - member 4;

datatype - member 5;

datatype - member 6;

datatype - member 7;

datatype - member 8;

datatype - member 9;

datatype - member 10;

datatype - member 11;

datatype - member 12;

datatype - member 13;

datatype - member 14;

datatype - member 15;

datatype - member 16;

datatype - member 17;

datatype - member 18;

datatype - member 19;

datatype - member 20;

datatype - member 21;

datatype - member 22;

datatype - member 23;

datatype - member 24;

datatype - member 25;

datatype - member 26;

datatype - member 27;

datatype - member 28;

datatype - member 29;

datatype - member 30;

datatype - member 31;

datatype - member 32;

datatype - member 33;

datatype - member 34;

datatype - member 35;

datatype - member 36;

datatype - member 37;

datatype - member 38;

datatype - member 39;

datatype - member 40;

datatype - member 41;

datatype - member 42;

datatype - member 43;

datatype - member 44;

datatype - member 45;

datatype - member 46;

datatype - member 47;

datatype - member 48;

datatype - member 49;

datatype - member 50;

datatype - member 51;

datatype - member 52;

datatype - member 53;

datatype - member 54;

datatype - member 55;

datatype - member 56;

datatype - member 57;

datatype - member 58;

datatype - member 59;

datatype - member 60;

datatype - member 61;

datatype - member 62;

datatype - member 63;

datatype - member 64;

datatype - member 65;

datatype - member 66;

datatype - member 67;

datatype - member 68;

datatype - member 69;

datatype - member 70;

datatype - member 71;

datatype - member 72;

datatype - member 73;

datatype - member 74;

datatype - member 75;

datatype - member 76;

datatype - member 77;

datatype - member 78;

datatype - member 79;

datatype - member 80;

datatype - member 81;

datatype - member 82;

datatype - member 83;

datatype - member 84;

datatype - member 85;

datatype - member 86;

datatype - member 87;

datatype - member 88;

datatype - member 89;

datatype - member 90;

datatype - member 91;

datatype - member 92;

datatype - member 93;

datatype - member 94;

datatype - member 95;

datatype - member 96;

datatype - member 97;

datatype - member 98;

datatype - member 99;

datatype - member 100;

datatype - member 101;

datatype - member 102;

datatype - member 103;

datatype - member 104;

datatype - member 105;

datatype - member 106;

datatype - member 107;

datatype - member 108;

datatype - member 109;

datatype - member 110;

datatype - member 111;

datatype - member 112;

datatype - member 113;

datatype - member 114;

datatype - member 115;

datatype - member 116;

datatype - member 117;

datatype - member 118;

datatype - member 119;

datatype - member 120;

datatype - member 121;

datatype - member 122;

datatype - member 123;

datatype - member 124;

datatype - member 125;

datatype - member 126;

datatype - member 127;

datatype - member 128;

datatype - member 129;

datatype - member 130;

datatype - member 131;

datatype - member 132;

datatype - member 133;

datatype - member 134;

datatype - member 135;

datatype - member 136;

datatype - member 137;

datatype - member 138;

datatype - member 139;

datatype - member 140;

datatype - member 141;

datatype - member 142;

datatype - member 143;

datatype - member 144;

datatype - member 145;

datatype - member 146;

datatype - member 147;

datatype - member 148;

datatype - member 149;

datatype - member 150;

datatype - member 151;

datatype - member 152;

datatype - member 153;

datatype - member 154;

datatype - member 155;

datatype - member 156;

datatype - member 157;

datatype - member 158;

datatype - member 159;

datatype - member 160;

datatype - member 161;

datatype - member 162;

datatype - member 163;

datatype - member 164;

datatype - member 165;

datatype - member 166;

datatype - member 167;

datatype - member 168;

datatype - member 169;

datatype - member 170;

datatype - member 171;

datatype - member 172;

datatype - member 173;

datatype - member 174;

datatype - member 175;

datatype - member 176;

datatype - member 177;

datatype - member 178;

datatype - member 179;

datatype - member 180;

datatype - member 181;

datatype - member 182;

datatype - member 183;

datatype - member 184;

datatype - member 185;

datatype - member 186;

datatype - member 187;

datatype - member 188;

datatype - member 189;

datatype - member 190;

datatype - member 191;

datatype - member 192;

datatype - member 193;

datatype - member 194;

datatype - member 195;

datatype - member 196;

datatype - member 197;

datatype - member 198;

datatype - member 199;

datatype - member 200;

datatype - member 201;

datatype - member 202;

datatype - member 203;

datatype - member 204;

datatype - member 205;

datatype - member 206;

datatype - member 207;

datatype - member 208;

datatype - member 209;

datatype - member 210;

datatype - member 211;

datatype - member 212;

datatype - member 213;

datatype - member 214;

datatype - member 215;

datatype - member 216;

datatype - member 217;

datatype - member 218;

datatype - member 219;

datatype - member 220;

datatype - member 221;

datatype - member 222;

datatype - member 223;

datatype - member 224;

datatype - member 225;

datatype - member 226;

datatype - member 227;

datatype - member 228;

datatype - member 229;

datatype - member 230;

datatype - member 231;

datatype - member 232;

datatype - member 233;

datatype - member 234;

datatype - member 235;

datatype - member 236;

datatype - member 237;

datatype - member 238;

datatype - member 239;

datatype - member 240;

datatype - member 241;

datatype - member 242;

datatype - member 243;

datatype - member 244;

datatype - member 245;

datatype - member 246;

datatype - member 247;

datatype - member 248;

datatype - member 249;

datatype - member 250;

datatype - member 251;

datatype - member 252;

datatype - member 253;

datatype - member 254;

datatype - member 255;

datatype - member 256;

datatype - member 257;

datatype - member 258;

datatype - member 259;

datatype - member 260;

datatype - member 261;

datatype - member 262;

datatype - member 263;

datatype - member 264;

datatype - member 265;

datatype - member 266;

datatype - member 267;

datatype - member 268;

datatype - member 269;

datatype - member 270;

datatype - member 271;

datatype - member 272;

datatype - member 273;

datatype - member 274;

datatype - member 275;

datatype - member 276;

datatype - member 277;

datatype -

Pointers:

Pointer is a Variable which stores the address of another Variable.

int van = 5; → Ordinary variable

int *a; → Pointer variable

Dereferencing operator.

a = & van;

Syntax → datatype * Pointer-Variable;

Wild pointer

If we not assign any value or value

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

with name, class, roll no, total mark 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 class name [50];

int roll no;

```
float total marks;
g;
char calculate grade (float marks)
{
    if (marks >= 80)
        return 'A';
    else if (marks >= 60)
        return 'B';
    else if (marks >= 50)
        return 'C';
    else
        return 'F';
}
```

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 = calculateGrade(Stu.rollno);

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 : %d\n", Stu.totalmarks);

Printf(" Grade: %c\n", grade);

getch();

return 0;

}

Pointers to pointer:

Pointer which stores the address of

the another pointer.

int *x, *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()

{

Output:
a=10 - b=5.

Pointers to function.

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

#include <stdio.h>

#include <conio.h>

voidswap(int *a, int *b);

int temp = *a;

* a = *b;

* b = temp;

3

int main()

int a=5, b=10;

swap(&a, &b);

Printf("a=%d , b=%d\n", a,b);

getch();

return 0;

3

int x=5, *P,**q; /* Declaring pointer variable
* P = &x; /* Assigning address of x to P
**q = &P; /* Assigning address of P to q
Printf("The values are : %d %d %d\n", x,*P,**q);

getch();

3 return 0;

Dynamic 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 holds 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);`

`n` - number of elements.

`element-size` - size of the elements.

Eg - `ptr = (float*) calloc(25, (float) 1);`

`realloc()`:

Re allocating the memory space without deleting the memory space.

Syntax `ptr = realloc(ptr, newsize);`

`free()` :
De allocating the memory space (or)
Freeing the memory space.
It should declare before the getch() and
return 0.