**Department of Computer Engineering**

**LAB MANUAL**

**Course name: C programming**

(Subject Code: FEL204)

**Year: First Year (REV- 2019‘C’ Scheme)**

**Semester: II**

**Academic Year: 2020-21**



****

**CONTENT**

|  |  |
| --- | --- |
| **.No.** | **Topic** |
| **1.** | **Vision and Mission of college and department** |
| **2.** | **Examination Scheme and University Syllabus** |
| **3.** | **Course Objectives** |
| **4.** | **PSO & PO** |
| **5.** | **CO & PO Mapping and CO & PSO Mapping** |
| **6.** | **List of Experiments** |

**MGM’s College of Engineering and Technology**

**Vision:**

To become one of the outstanding Engineering Institute in India by providing a conductive and vibrant environment to achieve excellence in the field of Technology.

**Mission:**

To empower the aspiring professional students to be prudent enough to explore the world of technology and mould them to be proficient to reach the pinnacle of success in the competitive global economy.

**Department Of Computer Engineering**

**Vision:**

To produce world class engineers by providing a stimulating educational environment to address the challenges of the millennium and to inculcate social responsibility in them.

**Mission:**

M1: To provide excellent academic environment by adopting an innovative teaching techniques through well-developed curriculum

M2: To foster a self-learning atmosphere for students to provide ethical solutions for societal challenges

M3: To establish Centre of Excellence in various domains of Computer Engineering and promote active research and development.

M4: To enhance the competency of the faculty in the latest technology through continuous development programs.

M5: To foster networking with alumni and industries for generating new growth opportunities

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Code** | **Course Name** | **Teaching Scheme (Contact Hours)** | | | | | | **Credits Assigned** | | | | | | |
| **Theory** | | **Pract.** | | **Tut.** | | **Theory** | | **Tut.** | **Pract.** | | **Total** | |
| FEL204 | C programming | -- | | 2 | | -- | | -- | | -- | 1 | | 1 | |
| **Course Code** | **Course Name** | **Examination Scheme** | | | | | | | | | | | | |
| **Theory** | | | | | | | **Term Work** | | | **Pract.**  **/oral** | | **Total** |
| **Internal Assessment** | | | | | **End Sem. Exam.** | **Exam. Duration (in Hrs)** |
| **Test1** | **Test 2** | | **Avg** | |
| FEL204 | C programming | -- | -- | | -- | | -- | -- | 25 | | | 25 | | 50 |

**Outcomes:** Learner will be able to…

1. Translate given algorithms to a program.
2. Correct syntax and logical errors.
3. Write iterative as well as recursive programs.
4. Represent data in arrays, strings and structures and manipulate them through a program.
5. Declare pointers and demonstrate call by reference concept.

**Lab Description:**

Weekly 2 hours of laboratory Programming Assignments on the following topics:

|  |  |  |
| --- | --- | --- |
| 1. | Basic data types and I/O operations |  |
| 2. | Branching Statements |  |
| 3. | Loop Statements |  |
| 4. | Arrays |  |
| 5. | Strings |  |
| 6. | Functions |  |
| 7. | Recursion |  |
| 8. | Structure and Union |  |
| 9. | Pointers |  |
| **Term Work:**  **Experiments** | **(20 Programs) and Assignments (2 Assignments) should be completed** | **by** |

## students on the given time duration

Experiments: 15 Marks

Assignment: 05 Marks

Attendance: 05 Marks

## Total: 25 Marks

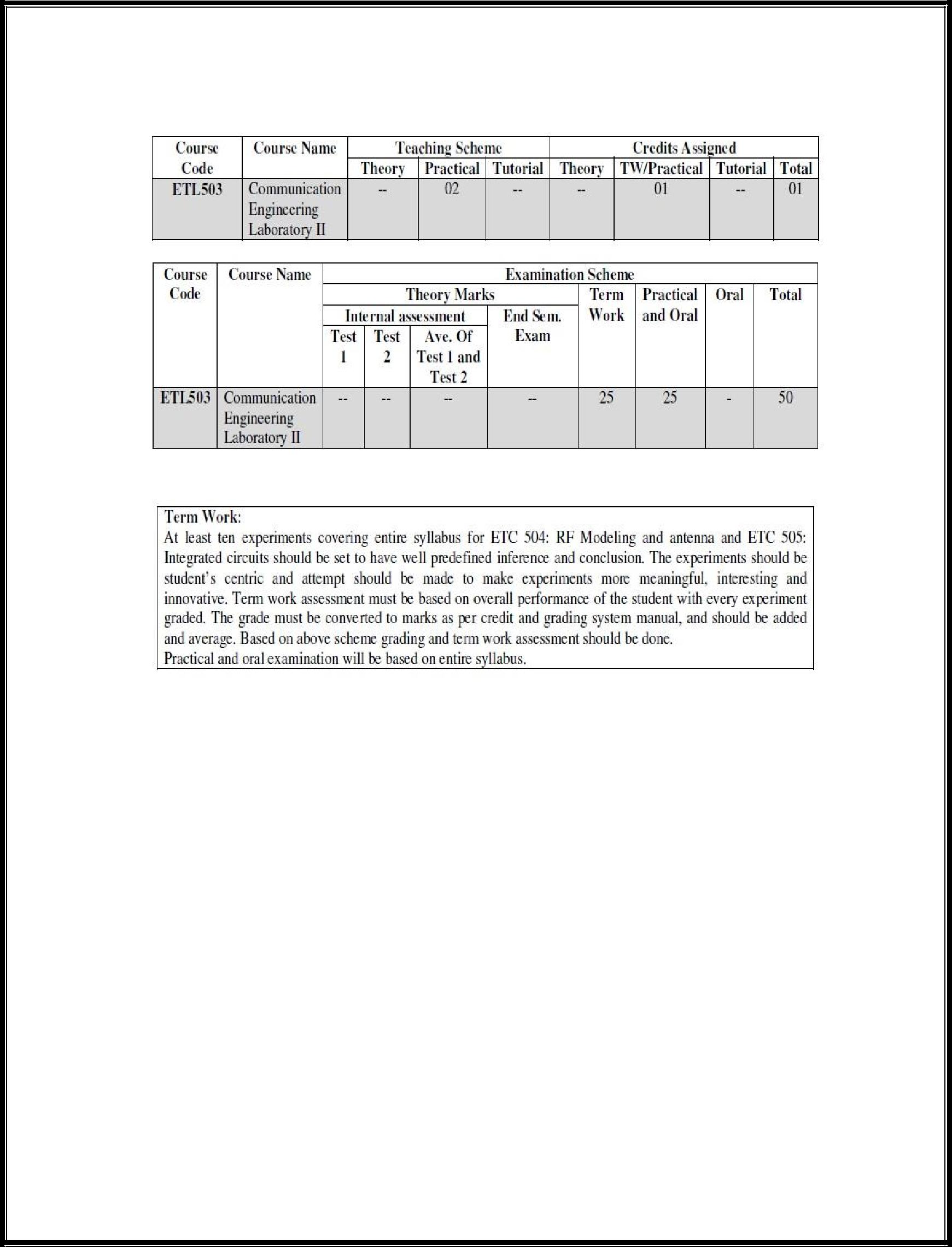
The final certification and acceptance of TW ensures the satisfactory performance of laboratory work and minimum passing in the TW.

**Practical and Oral :**

Practical and oral Exam should be conducted for the Lab, on Computer Programming in C subject for given list of experiments.

Implementation: 15 Marks

Oral : 10 Marks

**Course Objective:**

To provide exposure to problem-solving by developing an algorithm, flowchart and implement the logic using C programming language.

**Course Outcomes (CO):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of the course** | **CO code** | **Course outcome {CO}** |  |
|  |  |  |  |
|  | FEL204.1 | Translate given algorithms to a program and Correct syntax and logical errors. |  |
|  |  |  |  |
|  |  |  |  |
|  | FEL204.2 | Analyze and Design programs on branching and conditional statements. |  |
|  |  |  |  |
|  |  |  |  |
|  | FEL204.3 | Analyze and Design programs on iteration statements, functions as well as on recursion. |  |
| **C programming** |  |  |  |
|  |  |  |
| FEL204.4 | Represent data in arrays, strings manipulate them through a program. |  |
|  |  |
|  |  |  |  |
|  |  |  |  |
|  | FEL204.5 | Represent data in structures and manipulate them through a program. |  |
|  |  |  |  |
|  |  |  |  |
|  | FEL204.6 | Declare pointers and demonstrate call by reference concept. |  |
|  |  |  |  |

**Program Specific Outcomes (PSO)**

The Computer Engineering graduates will be able to

(i) Acquire skills to design, analyse and develop algorithms and implement them using high-level programming languages

(ii) Contribute their engineering skills in computing and information engineering domains like network design and administration, database design and knowledge engineering.

(iii) Develop strong skills in systematic planning, developing, testing implementing and providing IT solutions for different domains which helps in the betterment of life.

**Program Outcomes (PO)**

**Engineering Graduates will be able to:**

**PO1) Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2) Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3) Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4) Conduct investigations of complex problems:** Use research-based knowledge and research method including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5) Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6) The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7) Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8) Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9) Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10) Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11) Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12) Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**CO & PO Mapping:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name of the course** | **CO code** | **Program Outcomes (PO)** | | | | | | | | | | | | |
| **PO**  **1** | **PO**  **2** | **PO**  **3** | **PO**  **4** | **PO**  **5** | **PO**  **6** | **PO**  **7** | **PO**  **8** | **PO**  **9** | **PO**  **10** | **PO**  **11** | **PO**  **12** | | |
| **C programming** | 1 | 3 | 3 | 3  3  3 | 2 |  |  |  |  |  |  |  | | 1 | |
| 2 | 3 | 3 | 3 | 2 |  |  |  |  |  |  |  | | 1 | |
| 3 | 3 | 3 | 3 | 2 |  |  |  |  |  |  |  | | 1 | |
| 4 | 3 | 3 | 3 | 3 |  |  |  |  |  |  |  | | 1 | |
| 5 | 3 | 3 | 3 | 3 |  |  |  |  |  |  |  | | 1 | |
| 6 | 3 | 3 | 3 | 3 |  |  |  |  |  |  |  | | 1 | |

**CO & PSO Mapping:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of the course** | **CO Code** | **PSO1** | **PSO2** | **PSO3** |
| **C programming** | CO1 | 3 | 3 | 2 |
| CO2 | 3 | 1 | 2 |
| CO3 | 3 | 1 | 2 |
| CO4 | 3 | 3 | 3 |
| CO5 | 3 | 3 | 3 |
| CO6 | 3 | 3 | 3 |

**LIST OF EXPERIMENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name of Experiment** | **CO** | **PO** | **PSO** |
| 1 | Program on Basic Data types and Arithmetic Operators | 1 | 1,2,3,4,12 | 1,2,3 |
| 2 | Program on C Control Structure - Selection | 2 | 1,2,3,4,12 | 1,2,3 |
| 3 | Program on C control Structure - Multi-way Selection | 2 | 1,2,3,4,12 | 1,2,3 |
| 4 | Program on Ternary Operator/Conditional Operator. | 2 | 1,2,3,4,12 | 1,2,3 |
| 5 | Program on C Control Structure-Iteration (while loop) | 3 | 1,2,3,4,12 | 1,2,3 |
| 6 | Program on C Control Structure-Iteration (do-while loop) | 3 | 1,2,3,4,12 | 1,2,3 |
| 7 | Program on C Control Structure-Iteration (for loop) | 3 | 1,2,3,4,12 | 1,2,3 |
| 8 | Program on C Sum Of Digits of a given number | 3 | 1,2,3,4,12 | 1,2,3 |
| 9 | Program on Functions | 3 | 1,2,3,4,12 | 1,2,3 |
| 10 | Program on Pattern Design | 3 | 1,2,3,4,12 | 1,2,3 |
| 11 | Program in C for Fibonacci series | 3 | 1,2,3,4,12 | 1,2,3 |
| 12 | Program in C using Arrays | 4 | 1,2,3,4,12 | 1,2,3 |
| 13 | Program in C for String Operation | 4 | 1,2,3,4,12 | 1,2,3 |
| 14 | Program in C by using String Library Functions | 4 | 1,2,3,4,12 | 1,2,3 |
| 15 | Program in C by using Multidimensional Arrays | 4 | 1,2,3,4,12 | 1,2,3 |
| 16 | Program on Recursive Function | 3 | 1,2,3,4,12 | 1,2,3 |
| 17 | Program to calculate the Sum of the series in C | 3 | 1,2,3,4,12 | 1,2,3 |
| 18 | Program by using an Array of Structures | 5 | 1,2,3,4,12 | 1,2,3 |
| 19 | Program on Pointer Concept | 6 | 1,2,3,4,12 | 1,2,3 |
| 20 | Program to swap two numbers using call by reference | 6 | 1,2,3,4,12 | 1,2,3 |

**Experiment No: 1**

**Aim:- Write an algorithm, draw flowchart and C program to calculate simple interest after accepting principal sum, number of years and rate of interest**

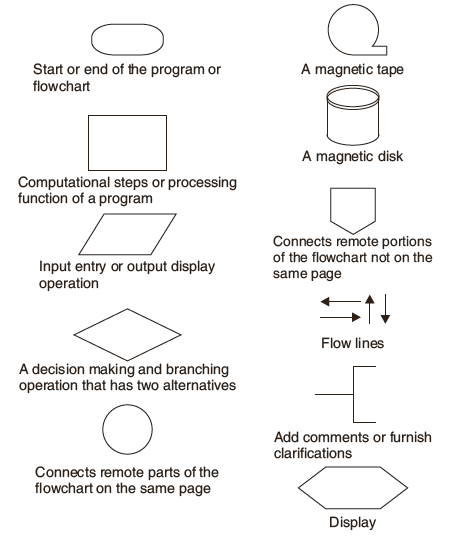
**Theory:-**

A computer program is a collection of [instructions](https://en.wikipedia.org/wiki/Instruction_set) that performs a specific task when [executed](https://en.wikipedia.org/wiki/Execution_(computing)) by a [computer](https://en.wikipedia.org/wiki/Computer). A computer requires programs to function.

A computer program is usually written by a [computer programmer](https://en.wikipedia.org/wiki/Computer_programmer) in a [programming language](https://en.wikipedia.org/wiki/Programming_language). From the program in its human-readable form of [source code](https://en.wikipedia.org/wiki/Source_code), a [compiler](https://en.wikipedia.org/wiki/Compiler) can derive [machine code](https://en.wikipedia.org/wiki/Machine_code)—a form consisting of instructions that the computer can directly execute.

Flowchart are the graphical representation of a program.

**Flow Charts: Symbolic Representation:-**

****

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output. Eficiency of an algorithm can be measured based on usage of different resources. Time complexity of an algorithm represents the amount of time required by the algorithm to run to completion. Space complexity of an algorithm represents the amount of memory space required by the algorithm in its life cycle. Algorithm design is a specific method to create a mathematical process in problem solving processes.

Steps to develop algorithm:

An algorithm is a step by step method of solving a problem.

Step 1: Obtain a description of the problem. This step is much more difficult than it appears

step 2: Analyze the problem. ...

Step 3: Develop a high-level algorithm. ...

Step 4: Refine the algorithm by adding more detail. ...

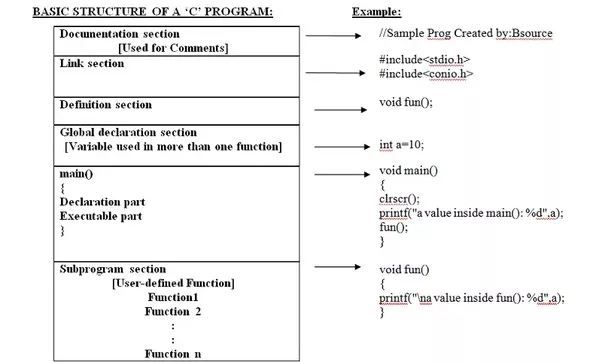
Step 5: Review the algorithm.

**C Programming Language:**

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie. A C program can vary from 3 lines to millions of lines and it should be written into one or more text files with extension ".c"; for example, *hello.c*.

The source code written in source file is the human readable source for your program. It needs to be "compiled", into machine language so that your CPU can actually execute the program as per the instructions given. The compiler compiles the source codes into final executable programs.

**Basic Structure of a C Program:**

* Documentations (Documentation Section)
* Preprocessor Statements (Link Section)
* Global Declarations (Definition Section)
* The main() function
* Local Declarations
* Program Statements & Expressions
* User Defined Functions
* 

# **C – data types:**

|  |  |
| --- | --- |
| **Types** | **Data Types** |
| Basic data types | int, char, float, double |
| Enumeration data type | enum |
| Derived data types | pointer, array, structure, union |
| Void data type | void |

Below table gives the detail about the storage size of each C basic data type in 16 bit processor. Please keep in mind that storage size and range for int and float datatype will vary depend on the CPU processor (8,16, 32 and 64 bit)

|  |  |
| --- | --- |
| **C Data types / storage Size** | **Range** |
| char / 1 | –127 to 127 |
| int / 2 | –32,767 to 32,767 |
| float / 4 | 1E–37 to 1E+37 with six digits of precision |
| double / 8 | 1E–37 to 1E+37 with ten digits of precision |
| long double / 10 | 1E–37 to 1E+37 with ten digits of precision |
| long int / 4 | –2,147,483,647 to 2,147,483,647 |
| short int / 2 | –32,767 to 32,767 |
| unsigned short int / 2 | 0 to 65,535 |
| signed short int / 2 | –32,767 to 32,767 |
| long long int / 8 | –(2power(63) –1) to 2(power)63 –1 |
| signed long int / 4 | –2,147,483,647 to 2,147,483,647 |
| unsigned long int / 4 | 0 to 4,294,967,295 |
| unsigned long long int / 8 | 2(power)64 –1 |

**Hello World Example**

A C program basically consists of the following parts −

* Preprocessor Commands
* Functions
* Variables
* Statements & Expressions
* Comments

**C - Header Files**. A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. There are two types of header files: the files that the programmer writes and the files that comes with your compiler.

C programming has several [in-built library functions](https://www.programiz.com/c-programming/library-function) to perform input and output tasks.

Two commonly used functions for I/O (Input/Output) are printf() and scanf().

The scanf() function reads formatted input from standard input (keyboard) whereas the printf() function sends formatted output to the standard output (screen).

Let us look at a simple code that would print the words "Hello World" −

#include <stdio.h>

#include<conio.h>

int main()

{/\* my first program in C \*/

clrscr();

printf("Hello, World!");

getch();

return 0;

}

Let us take a look at the various parts of the above program −

* The first line of the program *#include <stdio.h>* is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
* The next line *int main()* is the main function where the program execution begins.
* The next line /\*...\*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
* The next line *printf(...)* is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
* The next line return 0; terminates the main() function and returns the value 0.

**Accepting Input and Displaying output to screen:**

* printf() and scanf() functions are inbuilt library functions in C programming language which are available in C library by default. These functions are declared and related macros are defined in “stdio.h” which is a header file in C language.
* We have to include “stdio.h” file as shown in below C program to make use of these printf() and scanf() library functions in C language.
* **printf()** function is used to print onto the output screen.

Ex. printf("Hello, World! \n");

* **scanf()** function is used to data from keyboard.

Consider below example where user enters a number. This value is assigned to the variable “ch” and then displayed.

int number;

printf("Enter any number");

   scanf("%d", &number);

**Formula**

simpleInterest=amount\*noOfYears\*rateOfInterest / 100;

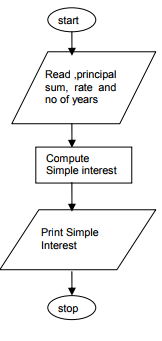
**Algorithm:**

1.Start   
2.Input the value principal sum, number of years and rate of interest  
4. Calculate

simpleInterest=amount\*noOfYears\*rateOfInterest / 100;

6. Print Simple interest   
7. End

**Flowchart :**

****

**Result**:………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications :**

Applications of given experiment are in software development for following areas,

* + Banking
  + Finance
  + Loans
  + Investments

**Conclusion:**

**Questionnaire**

1. Define the term computer program?
2. What is flowchart?
3. Define an Algorithm?
4. How Efficiency of Algorithm is calculated?
5. State time complexity?
6. Explain space complexity?
7. State and explain common symbols of flow chart?
8. What are the steps to develop the algorithm?
9. Who is the Inventor of C programming?
10. Enlist the function is used to display output in C program?

**Experiment No: 2**

**Aim :- Write an algorithm, draw flowchart and C program to check whether entered year is leap year or not using if else construct**

**Theory:-**

**Decision making in C**

Decision making is about deciding the order of execution of statements based on certain conditions or repeat a group of statements until certain specified conditions are met. C language handles decision-making by supporting the following statements,

* **if statement**
* **switch statement**

This is very useful when we need to evaluate multiple conditions. The switch block defines an expression (or condition) and case has a block of statements, based on the result of expression, corresponding case block gets executed. A switch can have any number of cases, however there should be only one default handler.

* **conditional operator statement (? : operator)**

Conditional operators return one value if condition is true and returns another value is condition is false.

This operator is also called as ternary operator.

Syntax     :        (Condition? true\_value: false\_value);

* **goto statement**

A goto statement in C programming provides an unconditional jump from

the 'goto' to a labeled statement in the same function.

goto label;

..

….

label: statement;

Here label can be any plain text except C keyword and it can be set anywhere in the C program above or below to goto statement.

**Decision making with if statement**

The if statement may be implemented in different forms depending on the complexity of conditions to be tested. The different forms are,

* Simple if statement
* if....else statement
* Nested if....else statement
* Using else if statement

The if-else statement is used to express decisions. In decision control statements (if-else and nested if), group of statements are executed when condition is true.  If condition is false, then else part statements are executed.

|  |  |
| --- | --- |
| **Decision control statements** | **Syntax/Description** |
| if | Syntax: if (condition)  { Statements; }  Description: In these type of statements, if condition is true, then respective block of code is executed. |
| if…else | Syntax: if (condition)  { Statement1; Statement2; }  else  { Statement3; Statement4; }  Description: In these type of statements, group of statements are executed when condition is true.  If condition is false, then else part statements are executed. |
| nested if else | Syntax: if (condition1){ Statement1; } else\_if(condition2)  { Statement2; }  else Statement 3;  Description: If condition 1 is false, then condition 2 is checked and statements are executed if it is true. If condition 2 also gets failure, then else part is executed. |

**Algorithm:**

Step 1: Begin

Step 2: [Input Year]

Read year

Step 3: [check for leap year]

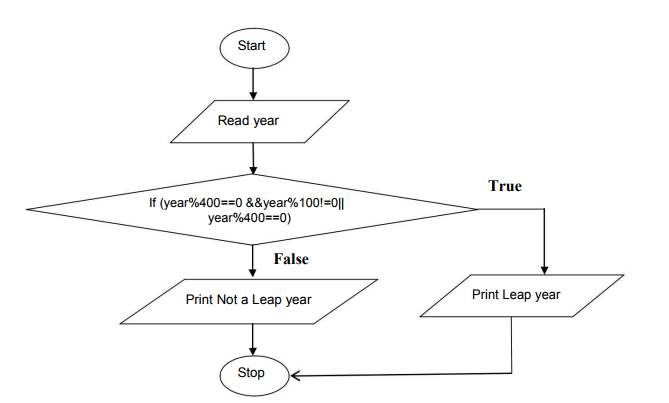
If ((year%4 ==0 && year%100! =0) || year%400==0)

Print the given year is leap year

Else Print the given year is not leap year

Step 4: End

**Flowchart:**

****

**Result**:……………………………………………………………………………………………………………………………………………………………………………

**Industrial Applications:**

Applications of if else are in software development for following areas,

* + Pattern matching
  + Predictions
  + It is used in almost all softwares where a decision is to be taken based on inputs.

**Conclusion:**

**Questionnaire**

1. Top of Form
2. What is a decision making statement? Explain.
3. What are the various decision making statements available in C?
4. List the standard functions in C?
5. How if statement works?
6. Explain working of if-else statement.
7. What is the use of header file?
8. What is extension of C program?
9. Explain how switch statement works?
10. State the syntax of conditional operator?
11. Explain syntax of goto statement.

**Experiment No : 3**

**Aim : Write an algorithm and draw flowchart and C program that simulates a simple calculator using switch case construct**

**Theory:**

The **switch** and **case** statements help control complex conditional and branching operations. The **switch** statement transfers control to a statement within its body.

[Syntax](javascript:void(0))

*selection-statement*:

**switch (***expression***)***statement*

*labeled-statement*:

**case***constant-expression***:***statement*

**default :***statement*

Control passes to the statement whose **case** *constant-expression* matches the value of **switch (** *expression* **)**. The **switch**statement can include any number of **case** instances, but no two case constants within the same **switch** statement can have the same value. Execution of the statement body begins at the selected statement and proceeds until the end of the body or until a **break** statement transfers control out of the body.

Use of the **switch** statement usually looks something like this:

**switch** ( *expression* )

**{**

*declarations*

   .

   .

   .

**case** *constant-expression* **:**

*statements executed if the expression equals the*

*value of this constant-expression*

      .

      .

      .

**break;**

**default :**

*statements executed if expression does not equal*

*any case constant-expression*

**}**

You can use the **break** statement to end processing of a particular case within the **switch** statement and to branch to the end of the **switch** statement. Without **break**, the program continues to the next case, executing the statements until a **break** or the end of the statement is reached. In some situations, this continuation may be desirable.

An expression is a combination of one or more operands, zero or more operators, and zero or more pairs of parentheses. There are three kinds of expressions: An arithmetic expression evaluates to a single arithmetic value.

The **default** statement is executed if no **case** *constant-expression* is equal to the value of **switch (** *expression* **)**. If the **default** statement is omitted, and no **case** match is found, none of the statements in the **switch** body are executed. There can be at most one **default** statement. The **default** statement need not come at the end; it can appear anywhere in the body of the **switch** statement. A **case** or **default** label can only appear inside a **switch** statement.

The type of **switch** *expression* and **case** *constant-expression* must be integral. The value of each **case** *constant-expression* must be unique within the statement body.

The **case** and **default** labels of the **switch** statement body are significant only in the initial test that determines where execution starts in the statement body. Switch statements can be nested. Any static variables are initialized before executing into any **switch** statements.

**If-esle**

* statement will be executed depend upon the output of the expression inside if statement.
* It using multiple statement for multiple choices.
* if statement evaluates integer, character, pointer or floating-point type or boolean type.

**switch**

* statement will be executed is decided by user.
* It using single expression for multiple choices.
* Switch statement evaluates only character or integer value.

A C instruction consists of a variable name on the left hand side of = and names and constants on the right hand side of = are connected by arithmetic operators like +, - , \* and /

## Arithmetic Operators:

|  |  |
| --- | --- |
| Operators | Description |
| \*, /, % | Multiplication, Division, Modular Division |
| +, - | Addition, Subtraction |
| ++, -- | Increment, Decrement |

## Relational Operators:

|  |  |
| --- | --- |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| == | Equal to |
| != | Not Equal to |

## Logical Operators:

|  |  |
| --- | --- |
| && | And |
| | | | Or |

Relational and Logical operators are used with the if statement.

**Algorithm:**

1. Start

2. Input 2 numbers

3. Display menu for operations

4. Accept choice

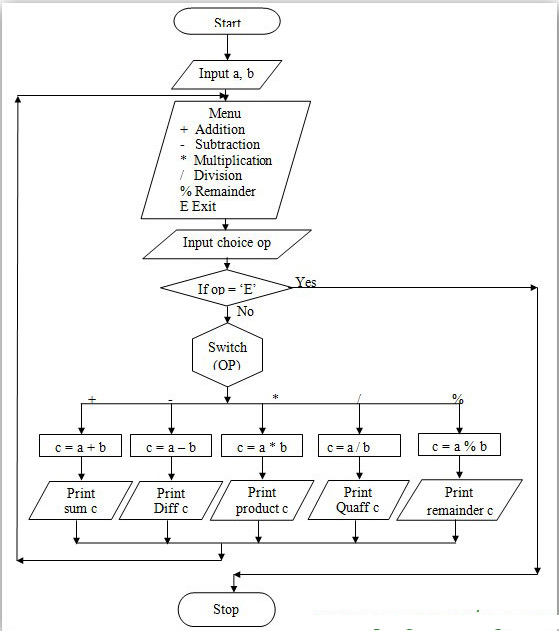
5. Perform operation based on choice

switch (char var)   
{   
case ‘ +’:   
//code to execute   
break;   
case ‘ -’:   
//code to execute   
break;   
case ‘ \*’:   
//code to execute   
break;   
case ‘ /’:   
//code to execute   
break;   
default:   
//code to execute   
break;   
}

5. Display result

6. End

**Flowchart:**



**Result**: ………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..

**Industrial Applications:**

Applications of switch case construct are in software development for following areas,

1. Hotel management
2. Bank Management
3. Switch case construct is used in almost all softwares where selection control mechanismis used to take different actions based on input.

**Conclusion:**

**Questionnaire**

1.Which of the following cannot be checked in a switch-case statement?

a.Character b. Integer

c.Float d. Enum

2. Enlist the various arithmetic operators?

3. Define an arithmetic expression?

4. Explain the various relational operators?

5. What are the various logical operators?

6. In mathematics and computer programming, which is the correct order of mathematical operators ?

a. Addition, Subtraction, Multiplication, Division

b. Division, Multiplication, Addition, Subtraction

c. Multiplication, Addition, Division, Subtraction

d. Addition, Division, Modulus, Subtraction

7.Compare switch and if-else.

8.What will be the output of following program ?

#include <stdio.h>

void main()

{

int a=10;

switch(a){

case 5+5:

printf("Hello\n");

default:

printf("OK\n");

}

}

1. Hello
2. OK
3. Hello

OK

1. Error

9. 'default' case is mandatory in a switch statement

a. True

b. False

10. Can we use a switch statement to switch on strings?

a. True

b. False

**Experiment No: 4**

**Aim:-Write an algorithm and draw flowchart and C program using ternary operator to find maximum number of given three numbers**

**Theory:-**

Ternary operator is an operator which can be used in place of an if else condition when both if and else part has only one line inside them. Lets look at the syntax of ternary operator in C language and understand ternary operators with example

**Syntax of Ternary Operators in C**  
  
Here is the syntax of ternary operator along with its if else equivalent code.

**(some condition to check) ? (if condition is true) : (if condition is false);**

/\*

if(some condition to check){

//if condition is true

}else{

//if condition is fasle

}

\*/

* Ternary operator has three operands (as the name ternary suggests).  
  First is the condition which needs to be checked for true or false.
* After the condition, there is a question mark ( ? ) and after the question mark comes the statement which is executed if the condition is true i.e the if part of if else.
* At last there is a colon ( : ) and after colon comes the statement which is executed if the condition is false i.e the else part of if else.

**Algorithm:**

1. Begin
2. Input a,b,c
3. big = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c) ;
4. output "big"
5. End

**Flowchart:**



Input a,b,c

Is a>b

Is b>c

Is a>c

Print largest

no b

Print largest

no c

Print largest

no c

Print largest

no a

Stop

read n

read n

**Result**:…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Industrial Applications:**

Applications of ternary operator are in software development for following areas,

* + Validation of forms for web applications
  + It can be used as replacement of if else statement in software coding

**Conclusion:**

**Questionnaire**

1. Which of the following is called as relational operator ?
   1. ==
   2. +=
   3. &&
   4. =
2. Predict the output of the following code ?

#include<stdio.h>

int main()

{

int i = 10;

printf("%d, %d\n", ++i, i++);

return 0;

}

1. 12, 11
2. 12, 10
3. 12, 12
4. Output may Vary from Compiler to Compiler
5. Guess the output ?

#include<stdio.h>

int main()

{

int k, num = 100;

k = (num > 50 ? (num <= 10 ? 100 : 200): 500);

printf("%d\n", num);

return 0;

}

* 1. 500
  2. 200
  3. 100
  4. 300

1. Which of the following cannot be used as LHS of the expression in for (exp1;exp2; exp3) ?

a) Variable b) Function c) typedef d) macros

1. A C variable cannot start with
2. An alphabet
3. A number
4. A special symbol other that underscore
5. Both b and c
6. A character variable can store at a time
7. 1 character
8. 8 character
9. 254 character
10. None of above

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  |  | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| 1. Can we use a switch statement to switch on strings? 2. True 3. False | | | | | |
| 1. The operator && in 'C' language is a 2. OR Operator 3. NOT Operator 4. And Operator 5. None of above 6. Associativity of an operator is either Left to Right or Right to Left.    1. False    2. True 7. Guess the output of the following program ?   #include<stdio.h>  int main()  {  int num1 = 30;  int num2 = 40;  int ans = num1 > num2;  printf("%d",ans);  return(0);  }   1. 1 2. 30 3. 0 4. 40   **Experiment No: 5**  **Aim:-Write an algorithm and draw flowchart and C program to check whether entered number is prime number or not using while loop**  **Theory :-**  Prime number is a number which is exactly divisible by one and itself only  Ex: 2, 3,5,7,………;  Loops are used to execute a set of statements repeatedly until a particular condition is satisfied.  How it Works  The below diagram depicts a loop execution,  loopflow diagram in C  If the Test Condition is true, then the loop is executed, and if it is false then the execution breaks out of the loop. After the loop is successfully executed the execution again starts from the Loop entry and again checks for the Test condition, and this keeps on repeating.  The sequence of statements to be executed is kept inside the curly braces { } known as the Loop body. After every execution of the loop body, condition is verified, and if it is found to be true the loop body is executed again. When the condition check returns false, the loop body is not executed, and execution breaks out of the loop  **Types of Loop**  There are 3 types of Loop in C language, namely:   1. while loop 2. for loop 3. do while loop   In Entry Controlled Loop, loop body is checked after checking the test condition i.e. condition is checked first after that loop body will execute while in Exit Controlled Loop, loop body will be executed first after that loop’s test condition is checked.  Entry Controlled Loops are : for, while  Exit Controlled Loop is : do while  while loop can be addressed as an entry control loop. It is completed in 3 steps.   * Variable initialization.(e.g int x = 0;) * condition(e.g while(x <= 10)) * Variable increment or decrement ( x++ or x-- or x = x + 2 )   **Syntax :**  variable initialization ;  while(condition)  {  statement(s);  variable increment or decrement  vaiable increment or decrement;  }  Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any nonzero value. The loop iterates while the condition is true.  When the condition becomes false, the program control passes to the line immediately following the loop.  Disadvantage: Possibility of entering an infinite loop if not properly coded:  **Algorithm:**  1. Start  2. read n  3. assign 2 to a  4. assign 0 to c, 1 to i  5. compute division of ‘a’ and ‘i’ ,store remainder in ‘r’  6. increment ‘i’, by ‘1’  7. compare ‘i’ and ‘a’ ,if ‘i’ is less than and equal to ‘a’, goto STEP 5  8. compare ‘c’ and 2, if ‘c’ is equal to 2, print a  9. increment ‘a’ by 1  10. compare ‘a’ and ‘n’, if a is less than and equal to n, goto STEP 5  11. Stop  **Flow chart:**  start  read n  a=2  c=0  i=1  r=a%i  yes  if(r==0)    c=c+1  no  i=i+1  if(i<=a)  yes  no  no  yes  if(c==2)  Print a  no  a=a+1  yes    if(a<=n)  no  stop  **Result:** …………………………………………………………………………………………  …………………………………………………………………………………………  …………………………………………………………………………………………  **Industrial Applications**:  Applications of while loop are in   1. software development where block of statements is to be executed repeatedly.   **Conclusion:**  **Questionnaire**  1. What is prime number?  2. What do you mean by looping ?  3. List out the types of looping statements.   1. State syntax of while loop.   5. What will be the output of the C program?  #include<stdio.h>  int main()  {  int i = 4;  while(i == 4--)  printf("Loop ");  return 0;  }   1. Compilation Error 2. Loop Loop Loop 3. Loop Loop Loop Loop 4. Prints Nothing   6. List disadvantages of while loop.  7. What will be the output of the C program?  #include<stdio.h>  int main()  {  int i = 0;  while(i++)  {  printf("Loop ");  if(i == 3)  break;  }  return 0;  }   1. Loop 2. Loop Loop Loop 3. Loop Loop Loop Loop 4. Prints Nothing   8. How many times above loop will execute?  int main()  {  int n=10,i=0;  while(1)  {  printf("END\n");  n++;  }  return 0;  }   1. 10 2. 5 3. 4 4. Infinite   9. Draw the flowchart for While loop.  10. What will be the output of the following code?  #include <stdio.h>  int main()  {  int i = 3;  while (i--)  {  int i = 100;  i--;  printf("%d ", i);  }  return 0;  }   1. Infinite Loop 2. 99 99 99 3. 99 98 97 4. 2 2 2 | | | | | |

**Experiment No: 6**

**Aim:-Write an algorithm and draw flowchart and C program to whether number is Armstrong number or not using do while loop**

**Theory:-**

A number is armstrong if the sum of cubes of individual digits of a number is equal to the number itself. For example 371 is an armstrong number as 33 + 73 + 13 = 371. Some other armstrong numbers are: 0, 1, 153, 370, 407.

Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop.

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

**Syntax:**

variable initialization ;

do

{

statement(s);

variable increment or decrement

vaiable increment or decrement;

} while(condition) ;

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

**Algorithm:**

1. Start

2. Read the user input

3. Assign temp=number

4. If temp is not equal to zero

do remainder = temp%10;sum = sum + remainder\*remainder\*remainder; temp = temp/10;

5. Check if number=sum then no is Armstrong no else not

7.Stop

**Flowchart:**

START

sum =0;

Input number

temp=number

Is temp!=0

Is number==sum

No. is Armstrong

No. is not Armstrong

STOP

remainder = temp%10;

sum = sum + remainder\*remainder\*remainder;

temp = temp/10;

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of do while loop are in

1. software development where block of statements is to be executed repeatedly.

**Conclusion:**

**Questionnaire**

1. What is mean by Armstrong number?

2. Give syntax of do-while loop.

3. Compare while and do-while loop.

4. What will be output of following C code?

#include<stdio.h>

int main(){

    int i=2,j=2;

    while(i+1 ? --i : j++)

         printf("%d",i);

    return 0;

}

5. How do while loop works?

6. Draw the flowchart for do-while loop.

7. Choose the correct statement

1. 0 represent a false condition
2. Non zero value represent a false condition
3. 1 represent a false condition
4. Anything that is not 1, represents a false condition

8. What will be the output of following program ?

#include <stdio.h>

void main()

{

    int cnt=1;

    do

    {

        printf("%d,",cnt);

        cnt+=1;

    }while(cnt>=10);

    printf("\nAfter loop cnt=%d",cnt);

    printf("\n");

}

1.[After loop cnt=1](javascript:void(0);)

1. [1,  
   After loop cnt=2](javascript:void(0);)
2. [After loop cnt=2](javascript:void(0);)

9. How many times this loop will execute?

#include<stdio.h>

int main(){

    char c=125;

    do

         printf("%d ",c);

    while(c++);

    return 0;

}

10.State the applications of do while loop.

**Experiment No: 7**

**Aim:-Write an algorithm and draw flowchart and c program to check**

**the given number is perfect or not**

**Theory:-**

A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

The syntax of a **for** loop in C programming language is −

for ( init ; condition ; increment )

{ statement(s); }

Here is the flow of control in a 'for' loop −

* The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
* Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
* After the body of the 'for' loop executes, the flow of control jumps back up to the **increment** statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
* The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

***Perfect number****:* A positive integer *n* is called a**perfect numbe**r if it is equal to the sum of all of its positive divisors, excluding n itself. **For example**, *6* is perfect integer number, because 1, 2 and 3 are its proper positive divisors and 1+2+3=6.

The next perfect number is *28* because 1+2+4+7+14=28.

The next perfect number is *496* because

1+2+4+8+16+31+62+124+248=496.

**Algorithm:**

step1 : start

step2 : read n , assign 1 to i, assign 1 to j and 0 to c

step3 : computethe division of i and j find remainder

step4 : if remainder is zero. compute sum of c and i,store in c.

step5 : increment j by 1

step6: if j<=i then goto step5

step7 : if product of 2 and i is equal to c then print c.

step8: increment i by 1

step9: if i<=n then goto step4

step10: stop

**Flowchart:**

no

yes

no

yes

no

yes

no

yes

c=c+i

Print c

i=i+1

if(2\*i==c)

if

(j<=i)

j=j+1

i=1

read n

start

j=1

c=0

r=i%j

if

(r==0)

if

(i<=n)

stop

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of for loop are in

1. software development where block of statements is to be executed repeatedly.

**Conclusion:**

**Questionnaire**

1.What is perfect number?

2. Write syntax of for loop?

3. For loop is a looping statement. True or False?

4. Draw the flow chart of For loop.

5. The following code ‘for(;;)’ represents an infinite loop. It can be terminated by.

a) break b) exit(0) c) abort() d) All of the mentioned

6. The correct syntax for running two variable for loop simultaneously is.

a) for (i = 0; i < n; i++) for (j = 0; j < n; j += 5)

b) for (i = 0, j = 0;i < n, j < n; i++, j += 5)

c) for (i = 0; i < n;i++){} for (j = 0; j < n;j += 5){}

d) None of the mentioned

7. Which for loop has range of similar indexes of 'i' used in for (i = 0;i < n; i++)?

a) for (i = n; i>0; i–)

b) for (i = n; i >= 0; i–)

c) for (i = n-1; i>0; i–)

d) for (i = n-1; i>-1; i–)

8. What is the output of this C code?

#include <stdio.h>

int main()

{

short i;

for (i = 1; i >= 0; i++)

printf("%d\n", i);

}

a) The control won’t fall into the for loop

b) Numbers will be displayed until the signed limit of short and throw a runtime error

c) Numbers will be displayed until the signed limit of short and program will successfully terminate

d) This program will get into an infinite loop and keep printing numbers with no errors

9. What is the output of this C code?

#include <stdio.h>

void main()

{

int k = 0;

for (k)

printf("Hello");

}

a) Compile time error b) hello c) Nothing d) Varies

10. In a for loop, if the condition is missing,then?

1. it is assumed to be present and taken to be false
2. it is assumed to be present and taken to be true
3. it result in the syntax error
4. execution will be terminated abruptly

**Experiment No: 8**

**Aim:-Write an algorithm and draw flowchart and C program to find sum of digits of a given number**

**Theory:-**

Sum of the individual digits means adding all the digits of a number.

Ex: 123 sum of digits is 1+2+3=6

In C, there are two statements break; and continue; specifically to alter the normal flow of a program.

Sometimes, it is desirable to skip the execution of a loop for a certain test condition or terminate it immediately without checking the condition.

For example: You want to loop through data of all aged people except people aged 65. Or, you want to find the first person aged 20.

In scenarios like these, continue; or a break; statement is used.

**C break Statement**

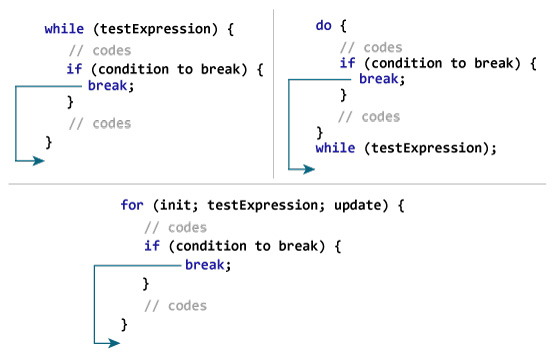
The break; statement terminates a loop ([for](https://www.programiz.com/cpp-programming/for-loop), [while and do..while loop](https://www.programiz.com/cpp-programming/do-while-loop)) and a [switch statement](https://www.programiz.com/cpp-programming/switch-case) immediately when it appears.

**Syntax of break**

break;

In real practice, break statement is almost always used inside the body of conditional statement (if...else) inside the loop.

**How break statement works?**



**C continue Statement**

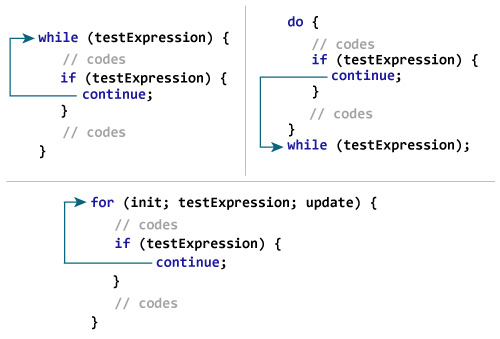
It is sometimes necessary to skip a certain test condition within a loop. In such case, continue; statement is used in C programming.

**Syntax of continue**

continue;

In practice, continue; statement is almost always used inside a conditional statement.

**How continue statement works?**



In C programming, goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

**Syntax of goto Statement**

goto label;

... .. ...

... .. ...

... .. ...

label:

statement;

... .. ...

In the syntax above, label is an identifier. When goto label; is encountered, the control of program jumps to label: and executes the code below it.

**Reason to Avoid goto Statement**

The goto statement gives power to jump to any part of program but, makes the logic of the program complex and tangled.

In modern programming, goto statement is considered a harmful construct and a bad programming practice.

The goto statement can be replaced in most of C program with the use of break and continue statements.

**Algorithm:**

Step1 : start

Step2 : read n

Step3 : assign 0 to s

Step4 : compute division of n and 10 store remainder in r

Step5 : compute sum of s and r, store result in s

Step6 : compute division of n and 10, store quotient in n

Step7 : compare n with 0,if n greater than 0 goto step4

Step8 : print s

Step9 : stop

**Flowchart:**

Start

read n

S=0

S=0

r=n%10

n=n/10

s=s+r

If (n>0)

yes

no

print s

stop

**Result:** ……………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of for loop are in

1. software development where block of statements is to be executed repeatedly.

**Conclusion:**

**Questionnaire**.

1.Which of the following is correct with respect to “Jump Statements” in C?

1. Continue statement
2. goto statement
3. return statement
4. break statement
5. All of the above

2.In a for loop, if the condition is missing, then infinite looping can not be avoided by a

1. Continue statement
2. goto statement
3. return statement
4. break statement

3.Difference between while loop and do-while loop.

4.Which of the following comments about for loop are not correct?

1. Index value is retained outside the loop
2. Index value can be changed from within the loop
3. goto can be used to jump,out of loop
4. Body of the loop can not be empty

5**.** Which of the following comment about for loop are correct?

1. Using break is equivalent to using a goto that jumps to the statement immediately following the loop
2. Continue is used to by pass the remainder of the current pass of the loop
3. if comma operator is used,then the value returned is the value of the right operand
4. All of above

6 .Break statement can be simulated by using ?

a.  goto b.  return c. Exit

7. The continue statemment cannot be used with

1. for
2. switch
3. do
4. while

8. Can we use continue statement without using loop?

9.What is the output of this C code?

for (i=0; i<10; ++i)

printf("%d", i&1);

* 1. 0101010101
  2. 0111111111
  3. 0000000000
  4. 1111111111

10.Consider the following program fragment

if(a > b)

if(b > c)

s1;

else s2;

s2 will be executed if

1. a <= b
2. b > c
3. b >= c and a <= b
4. a > b and b <= c

**Experiment No: 9**

**Aim:-Write an algorithm and draw flowchart and C program to find the GCD (greatest common divisor) of two given integers**

**Theory:-**

According to Mathematics, the Greatest Common Divisor (GCD) of two or more integers is the largest positive integer that divides the given integer values without remainder. For example, the GCD value of integer 8 and 12 is 4 because, both 8 and 12 are divisible by 1, 2, and 4 (remainder is 0) and the largest positive integer among the factors 1, 2, and 4 is 4.

The Greatest Common Divisor (GCD) is also known as the Highest Common Factor (HCF), or Greatest Common Factor (GCF), or Highest Common Divisor (HCD), or Greatest Common Measure (GCM). Here we will show you, How to write a c program to find GCD of two numbers using For Loop, While Loop, Functions, and Recursion.

Ex: GCD(12,24) is 12

Formula: GCD= product of numbers/ LCM of numbers

**Functions in C**

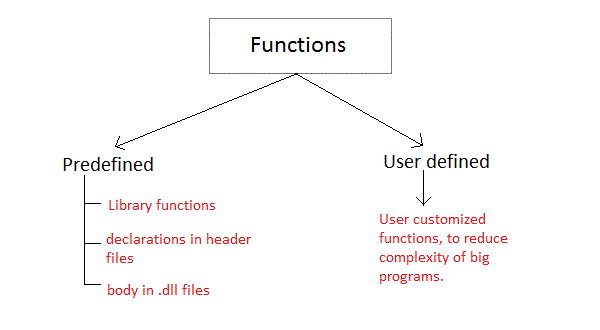
A function is a block of code that performs a particular task.

There are many situations where we might need to write same line of code for more than once in a program. This may lead to unnecessary repetition of code, bugs and even becomes boring for the programmer. So, C language provides an approach in which you can declare and define a group of statements once in the form of a function and it can be called and used whenever required.

These functions defined by the user are also know as User-defined Functions

C functions can be classified into two categories,

1. Library functions
2. User-defined functions



Library functions are those functions which are already defined in C library, example printf(), scanf(), strcat() etc. You just need to include appropriate header files to use these functions. These are already declared and defined in C libraries.

A User-defined functions on the other hand, are those functions which are defined by the user at the time of writing program. These functions are made for code reusability and for saving time and space.

**Benefits of Using Functions**

* It provides modularity to your program's structure.
* It makes your code reusable. You just have to call the function by its name to use it, wherever required.
* In case of large programs with thousands of code lines, debugging and editing becomes easier if you use functions.
* It makes the program more readable and easy to understand.

**Function Declaration**

returntype functionName(type1 parameter1, type2 parameter2,...);

Function declaration informs the compiler about the function name, parameters is accept, and its return type. The actual body of the function can be defined separately. It's also called as Function Prototyping. Function declaration consists of 4 parts.

* returntype
* function name
* parameter list
* terminating semicolon

**Function definition Syntax**

returntype functionName(type1 parameter1, type2 parameter2,...)

{

// function body goes here

}

The first line returntype functionName(type1 parameter1, type2 parameter2,...) is known as function header and the statement(s) within curly braces is called function body.

Note: While defining a function, there is no semicolon(;) after the parenthesis in the function header, unlike while declaring the function or calling the function.

functionbody

The function body contains the declarations and the statements(algorithm) necessary for performing the required task. The body is enclosed within curly braces { ... } and consists of three parts.

* local variable declaration(if required).
* function statements to perform the task inside the function.
* a return statement to return the result evaluated by the function(if return type is void, then no return statement is required).

### **Calling a function**

When a function is called, control of the program gets transferred to the unction.

functionName(argument1, argument2,...);

### **Passing Arguments to a function**

Arguments are the values specified during the function call, for which the formal parameters are declared while defining the function.It is possible to have a function with parameters but no return type. It is not necessary, that if a function accepts parameter(s), it must return a result too. It is possible to have a function with parameters but no return type. It is not necessary, that if a function accepts parameter(s), it must return a result too.

### **Returning a value from function**

A function may or may not return a result. But if it does, we must use the return statement to output the result. return statement also ends the function execution, hence it must be the last statement of any function. If you write any statement after the return statement, it won't be executed. The datatype of the value returned using the return statement should be same as the return type mentioned at function declaration and definition. If any of it mismatches, you will get compilation error

**Algorithm :**

**Main program:**

Step1 : start

Step2 : read a,b

Step3 : call sub program g=GCD(a,b)

Step4 : print the g value

Step5 : stop

**Sub program**:

Step1 : initialize the p=1, q, remainder

Step2 : remainder=p-(p/q\*q)

Step3 : remainder=0 return q else goto step 4

Step4 : GCD(q,remainder) return to main program

**Flowchart:**

Read a,b

Gcd( )

If n>m

Return

Gcd (n,m)

If n==0

Return m

Call the same function

Return gcd ( n,m%n)

Return to main program

true

false

true

F

Call sub program

G=gcd(a,b)

Print gcdvalue

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

1. Applications of GCD are reducing fractions

**Conclusion:**

**Questionnaire**

1. What is gcd?
2. Differntiate between local variables and global variables?
3. Explain actual parameters and formal parameters.
4. What is the difference between User Define and Library Functions?
5. Are exit () and return statements same in function definition?
6. Can we define a function with in a function?
7. What are called and calling functions?
8. Can we use single statement in loop body without using curly braces.
9. Is nested loop possible?
10. Can goto statement transfer program control from one function to another function ?

**Experiment No: 10**

**Aim:-Write an algorithm and C program to generate Pascal's triangle**

**Theory:-**

Pascal’s triangle which is used for a coefficient in the equation in polynominals.

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

The first thing one needs to know about Pascal’s triangle is that all the numbers outside the triangle are “0”s. To build the triangle, start with a “1” at the top, the continue putting numbers below in a triangular pattern so as to form a triangular array. So, each new number added below the top “1” is just the sum of the two numbers above, except for the edge which are all “1”s.

This can be summarized as:

0 row =1  
1 row = (0+1), (1+0) = 1, 1  
2 row = (0+1), (1+1), (1+0) = 1, 2, 1  
3 row = (0+1), (1+2), (2+1), (1+0) = 1, 3, 3, 1  
4 row = (0+1), (1+3), (3+3), (3+1), (1+0) = 1, 4, 6, 4, 1

**Storage classes in C**

In C language, each variable has a storage class which decides the following things:

scope i.e where the value of the variable would be available inside a program.

default initial value i.e if we do not explicitly initialize that variable, what will be its default initial value.

lifetime of that variable i.e for how long will that variable exist.

The following storage classes are most oftenly used in C programming,

1. Automatic variables
2. External variables
3. Static variables
4. Register variables

### **Automatic variables: auto**

**Scope:** Variable defined with auto storage class are local to the function block inside which they are defined.

**Default Initial Value:** Any random value i.e garbage value.

**Lifetime:** Till the end of the function/method block where the variable is defined.

A variable declared inside a function without any storage class specification, is by default an automatic variable. They are created when a function is called and are destroyed automatically when the function's execution is completed. Automatic variables can also be called local variables because they are local to a function. By default they are assigned garbage value by the compiler.

include<stdio.h>

void main()

{

int detail;

// or

auto int details; //Both are same

}

**External or Global variable**

**Scope:** Global i.e everywhere in the program. These variables are not bound by any function, they are available everywhere.

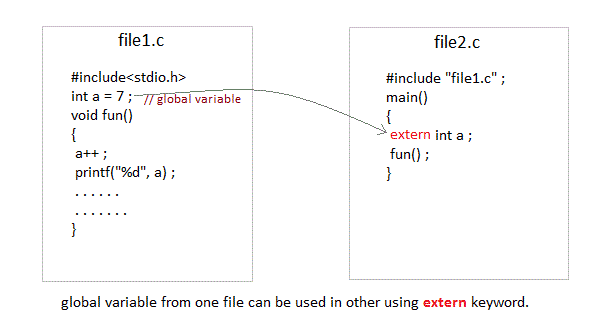
**Default initial value:** 0(zero).

**Lifetime:** Till the program doesn't finish its execution, you can access global variables.

A variable that is declared outside any function is a Global Variable. Global variables remain available throughout the program execution. By default, initial value of the Global variable is 0(zero). One important thing to remember about global variable is that their values can be changed by any function in the program.

**extern keyword**

The extern keyword is used with a variable to inform the compiler that this variable is declared somewhere else. The extern declaration does not allocate storage for variables.



int main()

{

extern int x; //informs the compiler that it is defined somewhere else

x = 10;

printf("%d", x);

}

int x; //Global variable x

**Static variables**

**Scope:** Local to the block in which the variable is defined

**Default initial value:** 0(Zero).

**Lifetime:** Till the whole program doesn't finish its execution.

A static variable tells the compiler to persist/save the variable until the end of program. Instead of creating and destroying a variable every time when it comes into and goes out of scope, static variable is initialized only once and remains into existence till the end of the program. A static variable can either be internal or external depending upon the place of declaration. Scope of internal static variable remains inside the function in which it is defined. External static variables remain restricted to scope of file in which they are declared.

They are assigned 0 (zero) as default value by the compiler.

#include<stdio.h>

void test(); //Function declaration (discussed in next topic)

int main()

{

test();

test();

test();

}

void test()

{

static int a = 0; //a static variable

a = a + 1;

printf("%d\t",a);

}

1 2 3

### **Register variable**

**Scope:** Local to the function in which it is declared.

**Default initial value:** Any random value i.e garbage value

**Lifetime:** Till the end of function/method block, in which the variable is defined.

Register variables inform the compiler to store the variable in CPU register instead of memory. Register variables have faster accessibility than a normal variable. Generally, the frequently used variables are kept in registers. But only a few variables can be placed inside registers. One application of register storage class can be in using loops, where the variable gets used a number of times in the program, in a very short span of time.

NOTE: We can never get the address of such variables.

**Syntax :**

register int number;

Note: Even though we have declared the storage class of our variable number as register, we cannot surely say that the value of the variable would be stored in a register. This is because the number of registers in a CPU are limited. Also, CPU registers are meant to do a lot of important work. Thus, sometimes they may not be free. In such scenario, the variable works as if its storage class is auto.

**Which storage class should be used and when**

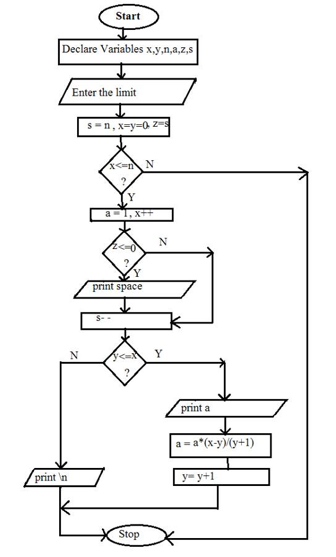
To improve the speed of execution of the program and to carefully use the memory space occupied by the variables, following points should be kept in mind while using storage classes

* We should use static storage class only when we want the value of the variable to remain same every time we call it using different function calls.
* We should use register storage class only for those variables that are used in our program very oftenly. CPU registers are limited and thus should be used carefully.
* We should use external or global storage class only for those variables that are being used by almost all the functions in the program.
* If we do not have the purpose of any of the above mentioned storage classes, then we should use the automatic storage class.

**Algorithm:**

1. Start
2. Declare variables *x, y, n, a, z, s*
3. Enter the limit
4. Initialize the value of variables, s=n , x=0, y=0 , z=s
5. Do the following operations in loop
   1. x = 0 to n
   2. a= 1, x++
   3. z=s to 0
   4. print space
   5. z—-
   6. y = o to x
   7. print a
   8. a = a\*(x-y)/(y+1)
   9. y= y+1
   10. go to next line
6. Repeat the process to n
7. Print the final required triangle
8. Stop

**Flowchart:**



**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

* 1. Pascal's Triangle is a widely used for predicting probability, determining binomial coefficients.

**Conclusion:**

**Questionnaire**

1. Define storage class.
2. List the different storage classes in C?
3. What are register variables?
4. What are the advantages of using register variables?
5. What is static variable?
6. Default value of extern variable is garbage or zero?
7. State the use of global variable?
8. Lifetime of the auto variable is global or local?
9. Stoarage of register variable is CPU or memory?
10. Define scope of the static variable.

**Experiment No: 11**

**Aim:-Write an algorithm and draw flowchart and C program to print Fibonacci series up to n number using for loop**

**Theory:-**

**C Standard Library Functions:**

C Standard library functions or simply C Library functions are inbuilt [functions](https://www.programiz.com/c-programming/c-functions) in C programming.

The prototype and data definitions of the functions are present in their respective header files, and must be included in your program to access them.

**For example:** If you want to use printf() function, the header file <stdio.h> should be included.

**Advantages of using C library functions**

There are many library functions available in C programming to help you write a good and efficient program. Below are the 4 most important advantages of using standary library functions.

**1.** **They work**

One of the most important reasons you should use library functions is simply because they work.

These functions have gone through multiple rigorous testing and are easy to use.

**2. The functions are optimized for performance**

Since, the functions are "standard library" functions, a dedicated group of developers constantly make them better.

In the process, they are able to create the most efficient code optimized for maximum performance.

**3.** **It saves considerable development time**

Since the general functions like printing to a screen, calculating the square root, and many more are already written. You shouldn't worry about creating them once again.

It saves valuable time and your code may not always be the most efficient.

**4. The functions are portable**

With ever changing real world needs, your application is expected to work every time, everywhere.

And, these library functions help you in that they do the same thing on every computer.

This saves time, effort and makes your program portable.

**C Library Functions Under Different Header File**

|  |  |
| --- | --- |
| [<ctype.h>](https://www.programiz.com/c-programming/library-function/ctype.h) | Character type functions |
| [<math.h>](https://www.programiz.com/c-programming/library-function/math.h) | Mathematics functions |
| <stdio.h> | Standard Input/Output functions |
| <stdlib.h> | Standard Utility functions |
| [<string.h>](https://www.programiz.com/c-programming/library-function/string.h) | String handling functions |
| <time.h> | Date time functions |

A fibonacci series is defined as follows

The first term in the sequence is 0

The second term in the sequence is 1

The sub sequent terms are found by adding the preceding two terms in the sequence

Formula:

let t1,t2,…………tn be terms in fibinacci sequence

t1=0, t2=1

tn=tn-2+tn-1……where n>2

**Algorithm:**

Step1 : start

Step2 : read n

Step3 : assign 0 to f1,1 to f2

Step4 : assign 1 to i

Step5 : print f1 and f2

Step6 : compute the sum of f1 and f2, store result in f

Step7 : print f

Step8 : assign f2 to f1 and f to f2

Step9 : increment i by 1

Step10 : compare i and n-2 if i less than or equal n-2 then goto step6

Step11 : stop

**Flowchart:**

Start

read n

f1=0

f2=1

i = 1

print f1 f2

f=f1+f2

print f

f1=f2

f2=f

i=i+1

If (i<=n-2)

yes

no

stop

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

* 1. Fibonacci numbers are important in the computational run-time analysis

**Conclusion:**

**Questionnaire**

1. What is the Fibonacci sequence explain logic.

1. Give recursive functionfor Fibonacci series calculation.
2. When is a switch statement can be better than an if statement?
3. What is a main() and difference between void main() and int main()?
4. How will you differentiate ++a and a++?
5. Write a statement to check whether any number is EVEN or ODD ?
6. State the limitation of ternary operator, comparing with if statement?
7. What are standard library functions?
8. What are advantages of using standard library functions?

10. printf() is available in which header file?

**Experiment No: 12**

**Aim:-Write an algorithm and C program to find both the largest and smallest number in a list of integers**

**Theory :-**

**Arrays in C**

In C language, arrays are reffered to as structured data types. An array is defined as finite ordered collection of homogenous data, stored in contiguous memory locations.

Here the words,

* finite means data range must be defined.
* ordered means data must be stored in continuous memory addresses.
* homogenous means data must be of similar data type.

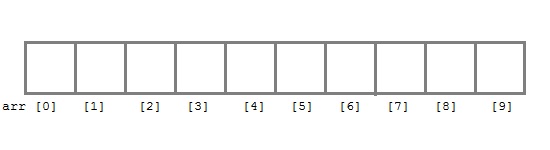
**Declaring an Array**

Like any other variable, arrays must be declared before they are used. General form of array declaration is,

**data-type variable-name[size];**

/\* Example of array declaration \*/

int arr[10];

 Here int is the data type, arr is the name of the array and 10 is the size of array. It means array arr can only contain 10 elements of int type.

Index of an array starts from 0 to size-1 i.e first element of arr array will be stored at arr[0] address and the last element will occupy arr[9].

Initialization of an Array

After an array is declared it must be initialized. Otherwise, it will contain garbage value(any random value). An array can be initialized at either compile time or at runtime.

**Compile time Array initialization**

Compile time initialization of array elements is same as ordinary variable initialization. The general form of initialization of array is,

data-type array-name[size] = { list of values };

/\* Here are a few examples \*/

int marks[4]={ 67, 87, 56, 77 }; // integer array initialization

float area[5]={ 23.4, 6.8, 5.5 }; // float array initialization

int marks[4]={ 67, 87, 56, 77, 59 }; // Compile time error

One important thing to remember is that when you will give more initializer(array elements) than the declared array size than the compiler will give an error.

#include<stdio.h>

main()

{

int i;

int arr[] = {2, 3, 4}; // Compile time array initialization

for(i = 0 ; i < 3 ; i++)

{

printf("%d\t",arr[i]);

}

}

2 3 4

**Runtime Array initialization**

An array can also be initialized at runtime using scanf() function. This approach is usually used for initializing large arrays, or to initialize arrays with user specified values. Example,

#inmain()

{

int arr[4];

int i, j;

printf("Enter array element");

for(i = 0; i < 4; i++)

{

scanf("%d", &arr[i]); //Run time array initialization

}

for(j = 0; j < 4; j++)

{

printf("%d\n", arr[j]);

}

}

### **Two dimensional Arrays**

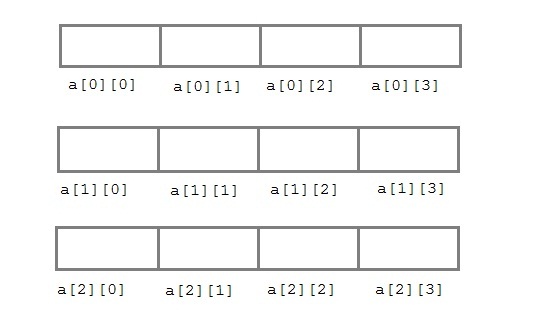
C language supports multidimensional arrays also. The simplest form of a multidimensional array is the two-dimensional array. Both the row's and column's index begins from 0.

Two-dimensional arrays are declared as follows,

data-type array-name[row-size][column-size]

/\* Example \*/

int a[3][4];



An array can also be declared and initialized together. For example,

int arr[][3] =

{

{0,0,0},

{1,1,1}

};

This program contains n number of elements, in these elements we can find the largest and smallest numbers and display these two numbers.

**An array** is a collection of data that holds fixed number of values of same type. For example: if you want to store marks of 100 students, you can create an array for it.

float marks[100];

The size and type of arrays cannot be changed after its declaration.

**How to insert and print array elements?**

int mark[5] = {19, 10, 8, 17, 9}

// insert different value to third element

mark[3] = 9;

// take input from the user and insert in third element

​scanf("%d", &mark[2]);

// take input from the user and insert in (i+1)th element

scanf("%d", &mark[i]);

// print first element of an array

printf("%d", mark[0]);

// print ith element of an array

printf("%d", mark[i-1]);

**Algorithm :**

Step1 : start

Step2 : declare array a[100]

Step3 : initialize i=1

Step 4 : read n numbers into array

Step4 : if i<n do as follows. If not goto step 5

Read a[i]

Increment i

Goto step 4

Step5 : min=a[0], max=a[0]

Step6 : initialize i=0

Step7 : if i<n do as follows. If not goto step 8

If a[i]<min

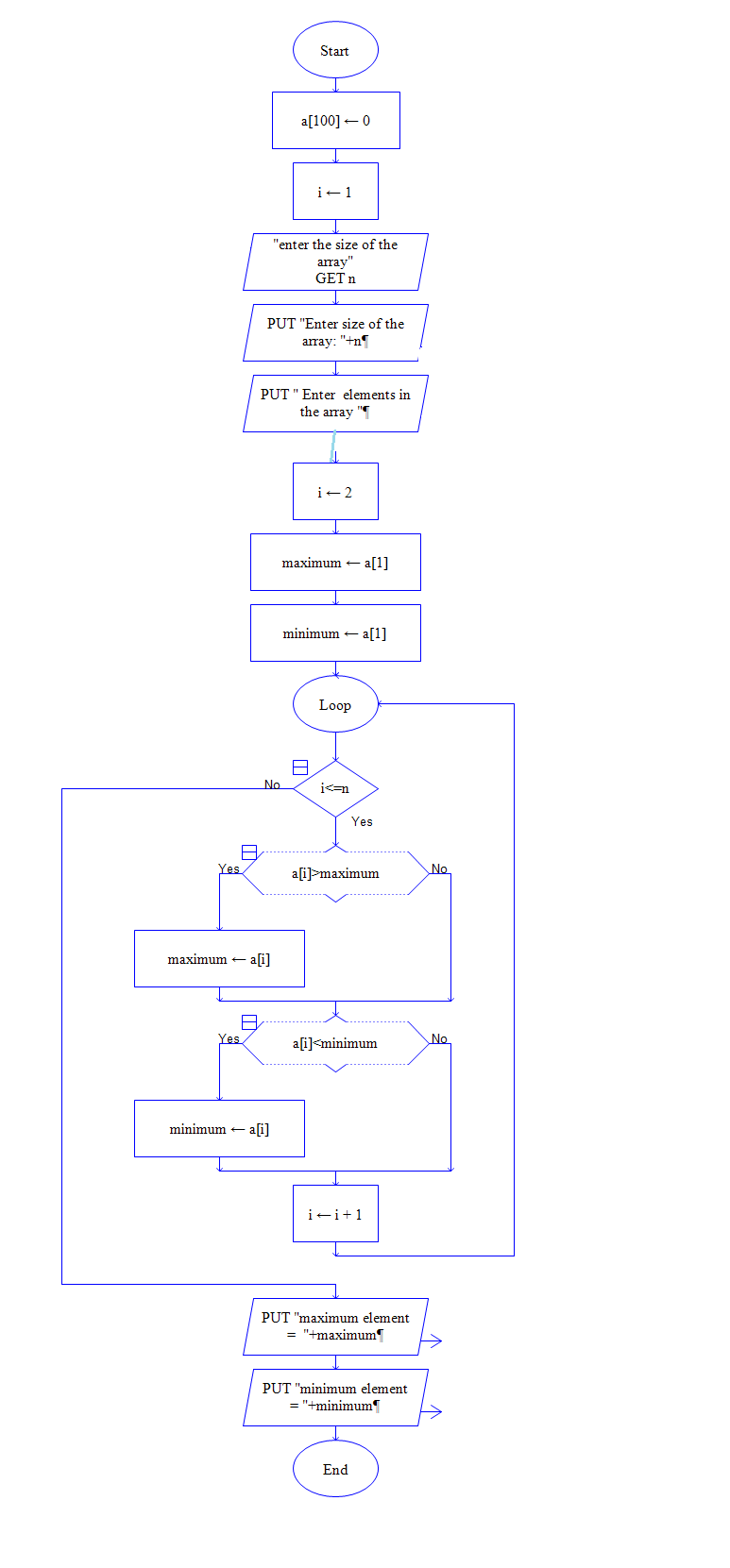
Assign min=a[i]

Increment i goto Step 7

Step8 : print min,max

Step9 : stop

**Flowchart:**



**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of for Array are in

* To store list of Employee or Student names,
* To store marks of students,
* To store list of numbers or characters etc.

**Conclusion:**

**Questionnaire**

1. Define Array?
2. How memory is allocated to array elements?
3. State the applications of Array.
4. How one dimensional array is declared?
5. Define two dimensional array?
6. Differntiate between one dimensional and two dimensional array?
7. How two dimensional array is initialzed?
8. How are individual array elements are identified?
9. In what way does an array differ from an ordinary variable?
10. How the array elements are inserted?

**Experiment No: 13**

**Aim:-Write an algorithm and C program to determine if the given string is a palindrome or not**

**Theory:-**

**String and Character Array**

String is a sequence of characters that is treated as a single data item and terminated by null character '\0'. Remember that C language does not support strings as a data type. A string is actually one-dimensional array of characters in C language. These are often used to create meaningful and readable programs.

For example: The string "hello world" contains 12 characters including '\0' character which is automatically added by the compiler at the end of the string.

**Declaring and Initializing a string variables**

There are different ways to initialize a character array variable.

char name[13] = "StudyTonight"; // valid character array initialization

char name[10] = {'L','e','s','s','o','n','s','\0'}; // valid initialization

Remember that when you initialize a character array by listing all of its characters separately then you must supply the '\0' character explicitly.

Some examples of illegal initialization of character array are,

char ch[3] = "hell"; // Illegal

char str[4];

str = "hell"; // Illegal

# **Console Input / Output functions:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Formatted Functions** | | | **Unformatted Functions** | | |
| Type | Input | Output | Type | Input | Output |
| Char | scanf( ) | printf( ) | char | getch( )  getche( )  getchar( ) | putch( )  putchar( ) |
| Int | scanf( ) | printf( ) | int | - | - |
| Float | scanf( ) | printf( ) | float | - | - |
| String | scanf( ) | printf( ) | string | gets( ) | puts( ) |

**String Input and Output**

Input function scanf() can be used with %s format specifier to read a string input from the terminal. But there is one problem with scanf() function, it terminates its input on the first white space it encounters. Therefore if you try to read an input string "Hello World" using scanf() function, it will only read Hello and terminate after encountering white spaces.

However, C supports a format specification known as the edit set conversion code %[..] that can be used to read a line containing a variety of characters, including white spaces.

#include<stdio.h>

#include<string.h>

void main()

{

char str[20];

printf("Enter a string");

scanf("%[^\n]", &str); //scanning the whole string, including the white spaces

printf("%s", str);

}

Another method to read character string with white spaces from terminal is by using the gets() function.

char text[20];

gets(text);

printf("%s", text);

The puts() function prints the charater array or string on the console. The puts() function is similar to printf() function, but we cannot print other than characters using puts() function.

**Unformatted Functions:**

getch() -It takes input, but not echo the character on console, but requires enter key.

getche() -It takes input, but not echo character and does not requires enter key.

getchar() -It takes input, echo character and requires enter key.

putchar(): This function is used to print one character on the screen, and this may be any character from C characterset(i.e it may be printable or non printable characters).

putch(): The putch() function is used to display all alphanumeric characters throught the standard output device like monitor. this function display single character at a time.

**Palindrome**:if the reverse of a string is equal to original string then it is called palindrome.

**Logic to check palindrome string**

The basic idea behind checking palindrome is if it can be read same from forward and backward then it is palindrome else not. Here in the below algorithm we will traverse the string character by character in both direction at the same time if they are equal then the string is palindrome.

Below is the step by step descriptive logic to check palindrome string.

1. Input a string from user, store it in some variable say str.
2. Find length of the given string and store it in some variable say endIndex.
3. Initialize another variable, to traverse the string in forward direction say startIndex = 0.
4. Run a loop until either startIndex >= endIndex or str[startIndex] != str[endIndex]. Otherwise increment startIndex and decrement endIndex.
5. Finally after loop check if startIndex >= endIndex then string is palindrome.

**Algorithm:**

Step1: Start

Step2: read string

Step3: set flag=0

Step4: for i=0, j=len-1 increment I decrement j

Step5: if string[i]=string[j]

Set flag =1

Else

Set flag=0

Break

Step6: if flag = 1

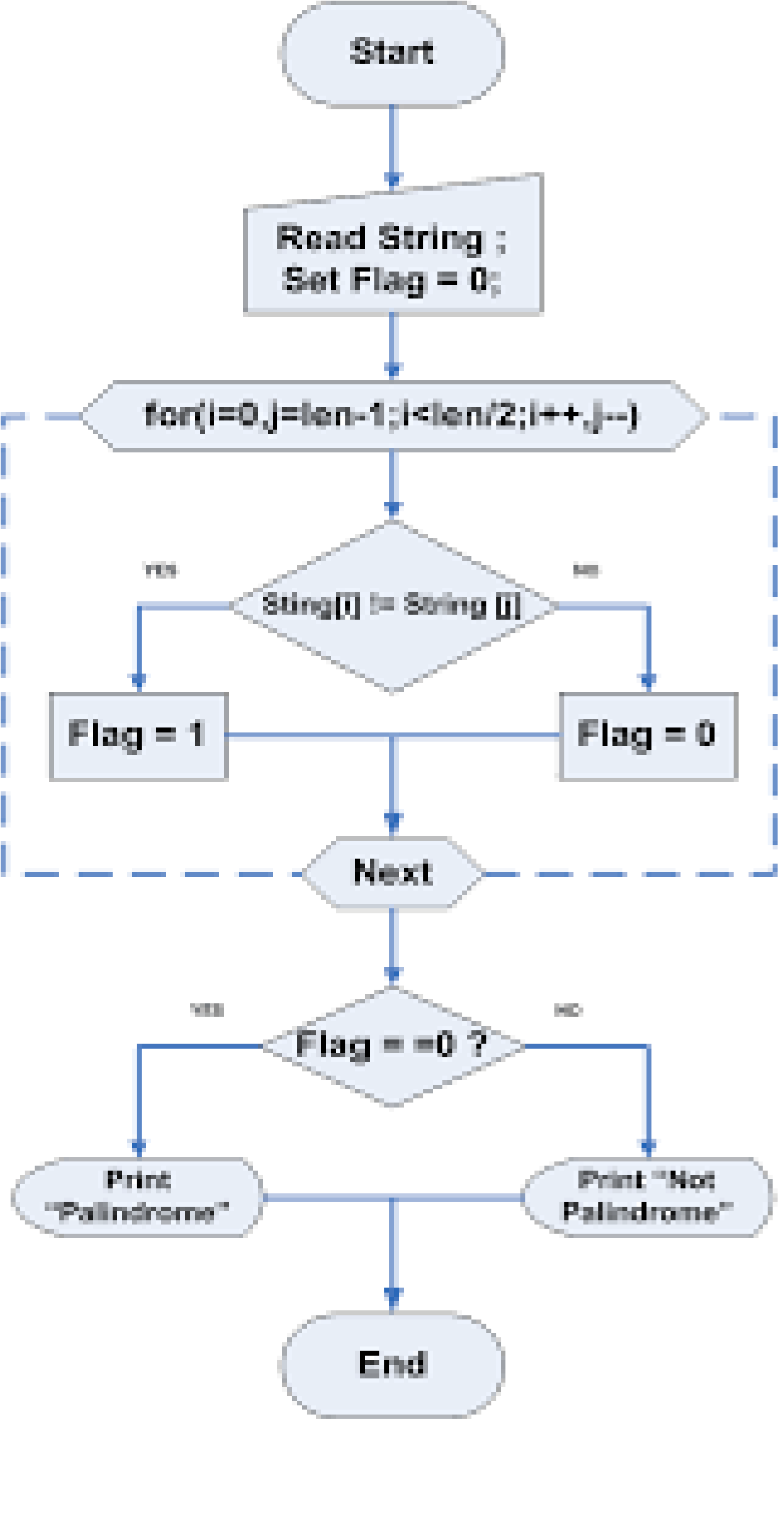
print ‘ String is Palindrome’

Else

print ‘String is not palindrome”

Step7: Stop

**Flowchart:**

****

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of string are

1. For comparison of strings
2. For checking palindrome

**Conclusion:**

**Questionnaire**

1. Define string in C.
2. Give generalized format to declare string in C.
3. How to initialize string?
4. Explain any five standard string functions with example?
5. What is the difference between strings and character arrays?
6. Give some string processing functions.
7. What is the delimiter for string?
8. What is the use of getchar() and gets() functions?
9. State the use of puts() function?
10. Differntiate between getchar() and getch() function?

**Experiment No: 14**

**Aim:-Write an algorithm and C program to count the lines, words and charactersin a given text**

**Theory:-**

**String Handling Functions**

C language supports a large number of string handling functions that can be used to carry out many of the string manipulations. These functions are packaged in string.h library. Hence, you must include string.h header file in your programs to use these functions.

The following are the most commonly used string handling functions.

|  |  |
| --- | --- |
| **Method** | **Description** |
| strcat() | It is used to concatenate(combine) two strings |
| strlen() | It is used to show length of a string |
| strrev() | It is used to show reverse of a string |
| strcpy() | Copies one string into another |
| strcmp() | It is used to compare two string |

In this program we have to count the no of lines, no of words and no of characters in a given program or given text by using the string function. Program reads the input text and then count number of lines by identifying new line character”\n”.

For a every line in given text,

* Using for loop search for a empty space in between the words in the string. Consecutively increment a variable. This variable gives the count of number of words
* No of characters = length of each line of text
* Adding number of words for every line and number of characters for every line we can find total number of characters and words for given text

**Algorithm :**

Step 1: Start

Step 2: Read the text until an empty line

Step 3: Compare each character with newline char ‘\n’ to count no of lines

Step 4: Compare each character with tab char ‘\t\’ or space char ‘ ‘ to count no of words

Step 5: Compare first character with NULL char ‘\0’ to find the end of text

Step 6: No of characters = length of each line of text

Step 7: Print no of lines, no of words, no of chars

Step 8: Stop

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of string are in

1. Calculating lenth of string
2. To reverse the string
3. To create duplicate strin
4. To compare string

**Conclusion:**

**Questionnaire**

|  |
| --- |
| 1. If the two strings are found to be unequal then strcmp returns difference between the first non-matching pair of characters. |
| |  |  |  |  | | --- | --- | --- | --- | | a. | True | b. | False | |
| 2. It is necessary that for the string functions to work safely the strings must be terminated with '\0'. |
| |  |  |  |  | | --- | --- | --- | --- | | **a.** | True | **b.** | False | |

3**.** State input and output functions for string.

4. What is the size of string and character.

5. Which conversion symbol is used to print the string.

6. Differntiate between strcmp() and strcpy().

7. What is the difference between strcat() and strlen().

8. Strcat function adds null character  
 a) Only if there is space  
 b) Always  
 c) Depends on the standard  
 d) Depends on the compiler

9. What is the output of this C code?

#include <stdio.h>

int main()

{

char str[10] = "hello";

char \*str1 = "world";

strncat(str, str1, 9);

printf("%s", str);

}

a) helloworld  
 b) Undefined behaviour  
 c) helloworl  
 d) hellowor

10.  What will be the value of var for the following?  
    var = strcmp(“Hello”, “World”);

a) -1  
b) 0  
c) 1  
d) strcmp has void return-type

**Experiment No: 15**

**Aim:-Write an algorithm and C program that perform Multiplication of Two Matrices**

**Theory:-**

Program takes the two matrices and checks for possibility of multiplication and perform multiplication if possible.

To perform matrix operations we need Two dimensional ARRAY. Multi-dimensional Array is a nothing different than any Array but the only difference is that it has more than one dimension to it e.g. square has two dimension and cube has three dimension. The Dimension of array is decided by us in number of square brackets [] selected. If We select two dimension than we have to take two square brackets[][]. It has to be accessed with the help of index number ranging from 0 to n-1 and 0 to m-1. (e.g. num[n][m] will have num[0][0],num[0][1],num[2][2] so on.)

Toal 3 two dimensional array are required. 2 for input matrices. One to save the save result of multiplication.

**Condition for Multiplication of two Matrices (matrix) :** In order to multiply two matrices, A and B, the number of columns in A must be equal to the number of rows in B. Thus, if A is an m x n matrix and B is r x s matrix, n=r.

**Algorithm:**

Step1 : start

Step2 : read the size of matrices A,B – m,n

Step3 : read the elements of matrix A

Step4 : read the elements of matrix B

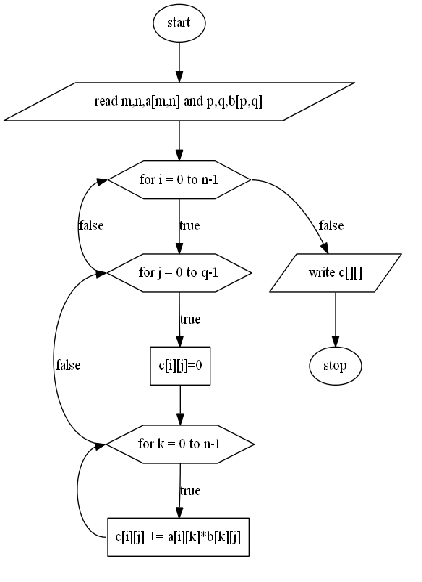
Step5 : check if n=p, if not print matrices can not be multiplied

Step 6 : Otherwise perform the multiplication of matrices

Step 7 : Print the resultant matrix

Step 8 : Stop

**Flowchart:**

****

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of 2 dimensional array are in

1. Image Processing
2. Computer Graphics
3. Computer Games
4. Animation

**Conclusion:**

**Questionnaire**

1**.** When an array is partially initialized, the rest of its elements will automatically be set to zero.  
a.True b. False

2. If you have an array of 20 elements, what is the good way to assign a value to each element?  
a. Assigning each element individually without using a loop  
b. Using a loop to assign a value to each element

3.  When an array is partially initialized, the rest of its elements will automatically be set to zero.  
a.True b. False

4. A two-dimensinal array represents data in the form of table with rows and columns.  
a.True b. False

5.  The whole array can be passed to a function. However, it can't be changed by the code in that function.  
a.True b. False

6. Which of the following statements about the array are not true?  
a. To use an array, previously it is declared and defined.  
b. A one-dimensional array can store data in the form of table with many rows and columns.  
c. The position of every element of the array is used to access that element that position.  
d. A one-dimensional array can store different types of data

|  |
| --- |
| 8. How will you print \n on the screen? |
| |  |  |  |  | | --- | --- | --- | --- | | **a.** | printf("\n"); | **b.** | echo "\\n"; | | **c.** | printf('\n'); | **d.** | printf("\\n"); | |

9.Find any errors in the following C program:  
int main(){ // This is another C program with some errors  
a,b,s int;  
d float;  
printf("The end of the program");  
return 0;  
}

10.If originally x=2,y=3, and z=5, what is the value of x,y, and z after executing the following code?  
if(x+1==y) y=y+1;  
else x++;

**Experiment No: 16**

**Aim:-Write an algorithm to find factorial of entered number using recursion. Using this function, write a C program to compute the binomial coefficient.**

**Theory:-**

When Function is call within same function is called **Recursion**. The function which call same function is called **recursive function**. In other word when a function call itself then that function is called **Recursive function**.

Recursive function are very useful to solve many mathematical problems like to calculate factorial of a number, generating Fibonacci series, etc.

How recursion works?

void recurse()

{

... .. ...

recurse();

... .. ...

}

int main()

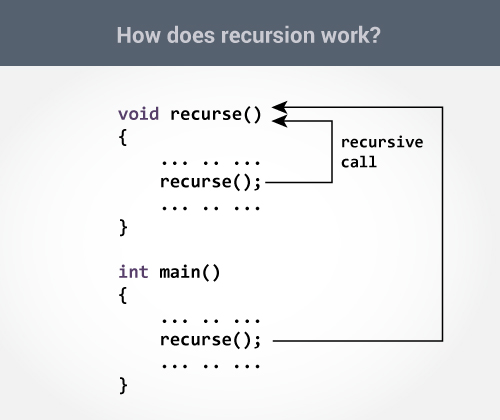
{

... .. ...

recurse();

... .. ...

}



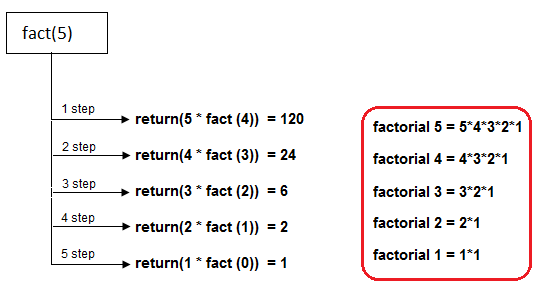
The recursion continues until some condition is met to prevent it.To prevent infinite recursion, [if...else statement](https://www.programiz.com/c-programming/c-if-else-statement) (or similar approach) can be used where one branch makes the recursive call and other doesn't.

Factorial of a positive number n is given by:

factorial of n (n!) = 1\*2\*3\*4....n

The factorial of a negative number doesn't exist. And the factorial of 0 is 1

A [binomial coefficient](http://en.wikipedia.org/wiki/Binomial_coefficient) C(n, k) can be defined as the coefficient of X^k in the expansion of (1 + X)^n. For example, your function should return 6 for n = 4 and k = 2, and it should return 10 for n = 5 and k = 2.



**main program**

Step1 : start

Step2 : read n, r

Step3 : call the sub program fact(n), bincoeff(n,r)

Step4 : Print values of factorial and binomial coefficient

Step5 : Stop

**Pseudo code**

Function factorial(n)

if(n<=0)

return 1;

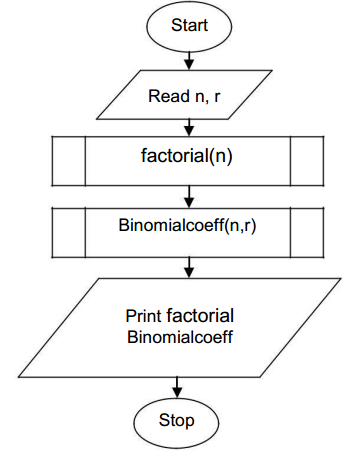
else return n\*factorial(n-1)

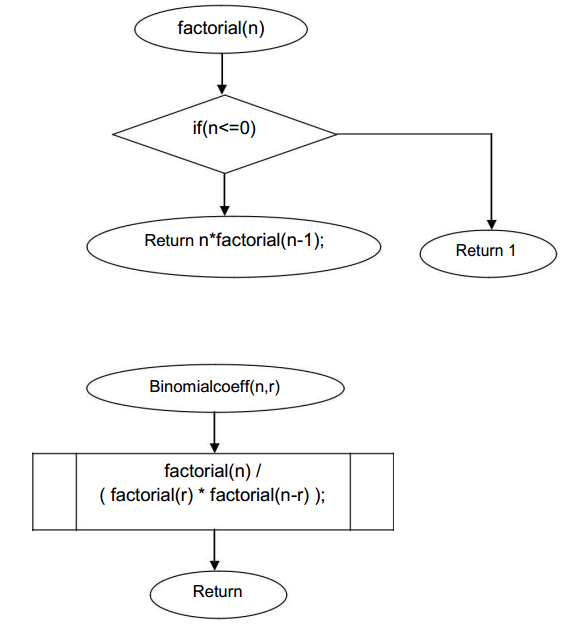
**Pseudo code**

Function bincoeff(n,r)

Return factoria(n)/factorial(r)\*factorial(n-r));

**Flowchart:**

****

****

**Result:** …………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of Recursion in

1. Software development where function is to be executed repeatedly within its body.
2. Operating System

**Conclusion:**

**Questionnaire**

1. Functions cannot return more than one value at a time

|  |  |  |  |
| --- | --- | --- | --- |
| a. | True | b. | False |

2 What is recursion?

3. If return type for a function is not specified, it defaults to int

|  |  |  |  |
| --- | --- | --- | --- |
| a. | True | b. | False |

4. Functions cannot return a floating point number

|  |  |  |  |
| --- | --- | --- | --- |
| **a.** | Yes | **b.** | No |

5.How recursion works?

6. If a function contains two return statements successively, the compiler will generate warnings. Yes/No ?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **a.** | | | Yes | | **b.** | No | |
| 7.The keyword used to transfer control from a function back to the calling function is | | | | | | |
| |  |  |  |  | | --- | --- | --- | --- | | a. | switch | b. | goto | | c. | go back | d. | return |   8. Which of the following statements are correct about the program?  include<stdio.h>  int main()  {  printf("%p\n", main());  return 0;  }   |  |  | | --- | --- | | a. | It prints garbage values infinitely | | b. | Runs infinitely without printing anything | | c. | Error: main() cannot be called inside printf() | | d. | No Error and print nothing |  1. What is disadvantage of recursive function?   10. In C all functions except main() can be called  recursively. | | | | | | |
| a. | | True | | b. | | | False |

**Experiment No: 17**

**Aim:-Write a C program to calculate the Sum**

**Sum=1-x2/2! +x4/4!-x6/6!+x8/8! x10/10!**

**Theory :**

#### List of inbuilt C functions in math.h file:

#### “math.h” header file supports all the mathematical related functions in C language. All the arithmetic functions used in C language are given below.

|  |  |
| --- | --- |
| **Function** | **Description** |
| [floor ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-floor-function/) | This function returns the nearest integer which is less than or equal to the argument passed to this function. |
| [round ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-round-function/) | This function returns the nearest integer value of the float/double/long double argument passed to this function. If decimal value is from “.1 to .5”, it returns integer value less than the argument. If decimal value is from “.6 to .9”, it returns the integer value greater than the argument. |
| [ceil ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-ceil-function/) | This function returns nearest integer value which is greater than or equal to the argument passed to this function. |
| [sin ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate sine value. |
| [cos ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate cosine. |
| [cosh ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate hyperbolic cosine. |
| [exp ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate the exponential “e” to the xth power. |
| [tan ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate tangent. |
| [tanh ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate hyperbolic tangent. |
| [sinh ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculate hyperbolic sine. |
| [log ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) | This function is used to calculates natural logarithm. |
| [log10 (](http://fresh2refresh.com/c/c-arithmetic-functions/c-sin-cos-tan-exp-log-function/) ) | This function is used to calculates base 10 logarithm. |
| [sqrt ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-sqrt-function/) | This function is used to find square root of the argument passed to this function. |
| [pow ( )](http://fresh2refresh.com/c/c-arithmetic-functions/c-pow-function/) | This is used to find the power of the given number. |
| [trunc()](http://fresh2refresh.com/c/c-arithmetic-functions/c-trunc-function/) | This function truncates the decimal value from floating point value and returns integer value. |

**Algorithm :**

**main program:**

Step1 : start

Step2 : declare x,i,n,s=0,c

Step3 : read x value

Step4 : for i=0 , n=0; i<=10; i=i+2, n++ goto step 5

Step5 : s=s+(pow(-1,n)\*pow(x,i)/fact(i))

Step6 : print s value

Step7 : stop

**Sub program:**

Step1 : while x!=0 goto Step 2

Step2 : y=y+x; x—

Step3 : return y

Step4 : return to main program

**FLOWCHART:**

Read x,I,n,c

S = 0

S=s+(pow(-1,n)\*pow(x,i) / fact(i))

Print s

Sub Program

While x!=0

Y=y\*x

x--

Return y

Fact ()

Return tomain program

start

stop

i=0,n=0 i<=10

i=i+2,n++

Result

………………………………………………………………………………………

…………………………………………………………………………………………

**Industrial Applications:**

Applications of for loop are in

1. software development where block of statements is to be executed repeatedly.
2. sin and cosine series used them a lot in the Fourier Series, which is important in electrical engineering, vibration analysis, acoustics, optics, signal and image processing, and data compression, among other fields.

**Conclusion:**

**Questionnaire**

1. What is the difference between library functions and user defined functions.
2. What is the use of math.h header file?
3. Explain the use of ceil() function?
4. Explain the syntax of floor() function?
5. State the different functions available in math.h header file?
6. State the use pf pow() function?
7. Write syntax of sqrt() function.
8. Explain working of trunc() function.
9. What are the applications of recursive functions?

10. Every function must return a value

|  |  |  |  |
| --- | --- | --- | --- |
| **a.** | Yes | **b.** | No |

**Experiment No: 18**

**Aim:-Write a C program to maintain a record of student details using an array of structures**

Note- Use four fields (Roll number, Name, Marks, and Grade). Assume appropriate data type for each field. Print the marks of the student, given student name as input.

**Theory:-**

A structure is a user defined data type in C. A structure creates a data type that can be used to group items of possibly different types into a single type.

***How to create a structure?***  
‘struct’ keyword is used to create a structure. Following is an example.

|  |
| --- |
| struct addrress  {     char name[50];     char street[100];     char city[50];     char state[20]     int pin;  }; |

***How to declare structure variables?***  
A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

|  |
| --- |
| // A variable declaration with structure declaration.  struct Point  {     int x, y;  } p1;  // The variable p1 is declared with 'Point'      // A variable declaration like basic data types  struct Point  {     int x, y;  };    int main()  {     struct Point p1;  // The variable p1 is declared like a normal variable  } |

***How to access structure elements?***  
Structure members are accessed using dot (.) operator.

|  |
| --- |
| struct Point  {     int x, y;  };    int main()  {     struct Point p1 = {0, 1};       // Accesing members of point p1     p1.x = 20;     printf ("x = %d, y = %d", p1.x, p1.y);       return 0;  } |

***What is an array of structures?***  
Like other primitive data types, we can create an array of structures.

|  |
| --- |
| struct Point  {     int x, y;  };    int main()  {     // Create an array of structures     struct Point arr[10];       // Access array members     arr[0].x = 10;     arr[0].y = 20;       printf("%d %d", arr[0].x, arr[0].y);     return 0;  } |

**Algorithm:**

Step 1: [Start]

Step 2: [Input the number of students]

Read n

Step 3: [Input roll number,name,marks,grade]

Read roll,grade,marks,name

Step 4: [Input name of student whose marks to be displayed]

Read sname

Step 5: [Search student details]

For i= 0 thru n in steps of 1

If (strcmp (sname, s[i].name)==0)

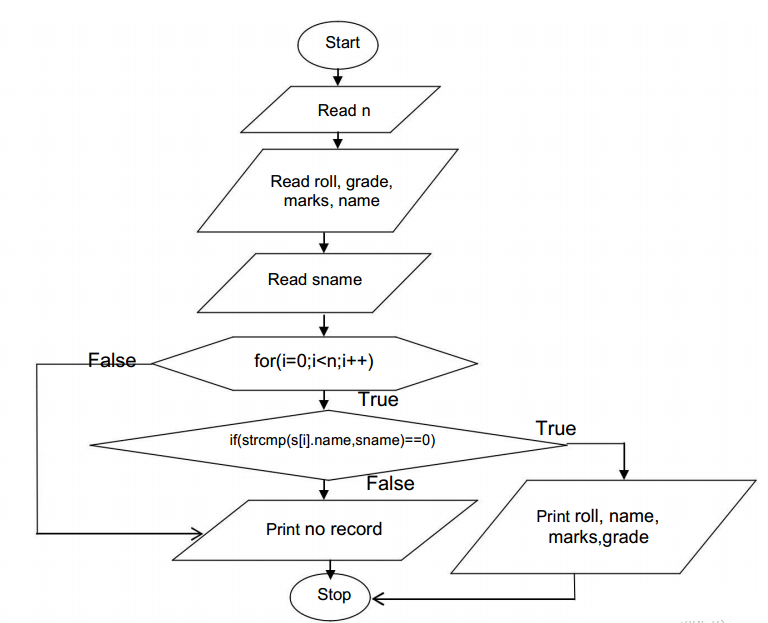
Print roll, name, grade marks

Else

Print record not found

Step 6: [Stop]

**Flowchart:**

****

**Result**

………………………………………………………………………………………

…………………………………………………………………………………………

**Industrical Application:**

1. It is most used data structure.
2. It is used in every possible situation where you need to gather  similar objects at one place. Simple Example can be collection of all the book titles in a Library Management systems

**Conclusion**

**Questionnaire**

1. Define structure. Which keyword is used to declare structure ?

2. Declare and initialize structure with example ?

3. What is the array of structure. why it is used ?

4. How memory is allocated for structure. what is the size of structure ?

5. What is union?

6. What is the difference between structure and union ?

7. What is the structure within structure ?

8. What are applications of array of structure ?

9. What is exit() function in C ?

10. What is the difference between exit() and return() in C?

**Experiment No: 19**

**Aim:-Write a C program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers**

**Theory:-**

A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address. The general form of a pointer variable declaration is:

type \*var-name;

Here, **type** is the pointer's base type; it must be a valid C data type and **var-name** is the name of the pointer variable. The asterisk \* you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Following are the valid pointer declaration:

int \*ip; /\* pointer to an integer \*/

double \*dp; /\* pointer to a double \*/

float \*fp; /\* pointer to a float \*/

char \*ch /\* pointer to a character \*/

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

**How to use Pointers?**

There are few important operations, which we will do with the help of pointers very frequently. **(a)** we define a pointer variable **(b)** assign the address of a variable to a pointer and **(c)** finally access the value at the address available in the pointer variable. This is done by using unary operator **\*** that returns the value of the variable located at the address specified by its operand.

**C Pointers in Detail:**

Pointers have many but easy concepts and they are very important to C programming. There are following few important pointer concepts which should be clear to a C programmer:

|  |  |
| --- | --- |
| **Concept** | **Description** |
| C - Pointer arithmetic | There are four arithmetic operators that can be used on pointers: ++, --, +, - |
| [C - Array of pointers](http://www.tutorialspoint.com/cprogramming/c_array_of_pointers.htm) | You can define arrays to hold a number of pointers. |
| [C - Pointer to pointer](http://www.tutorialspoint.com/cprogramming/c_pointer_to_pointer.htm) | C allows you to have pointer on a pointer and so on. |
| [Passing pointers to functions in C](http://www.tutorialspoint.com/cprogramming/c_passing_pointers_to_functions.htm) | Passing an argument by reference or by address both enable the passed argument to be changed in the calling function by the called function. |
| [Return pointer from functions in C](http://www.tutorialspoint.com/cprogramming/c_return_pointer_from_functions.htm) | C allows a function to return a pointer to local variable, static variable and dynamically allocated memory as well. |

**Sum, Mean and Standard Deviation**

**Sum** of all elements of an array can be found by adding values of all elements of an array. Using for loop, addition can be done for all elements.

ptr=a;

for(i=0;i<n;i++)

{

sum=sum+ \*ptr;

ptr++;

}

**Mean** can be found by dividing sum by total number of elements in an array.

mean=sum/n;

**Standard deviation** can be calculated by following steps,

**Step 1:** Find the mean.

**Step 2:** For each data point, find the square of its distance to the mean.

**Step 3:** Sum the values from Step 2.

**Step 4:** Divide by the number of data points.

**Step 5:** Take the square root.

**Pseudocode.**

ptr=a;

for(i=0;i<n;i++)

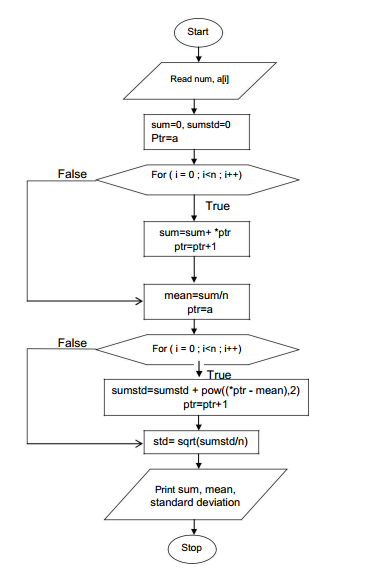
{

sumstd=sumstd + pow((\*ptr -mean),2); ptr++;

}

std= sqrt(sumstd/n);

**Flowchart:**

****

**Result**

………………………………………………………………………………………

…………………………………………………………………………………………

………………………………………………………………………………………….

**Industrical Application:**

Applications of Pointers are in

1.We can access array using pointer. We can store base address of array in pointer.

2.Another pointer application is to allocate memory dynamically.

3.Passing Strings to function

**Conclusion:**

**Questionnaire**

1. What is the advantage of pointer?
2. What is the difference between malloc() and calloc() functions?
3. How to declare string using pointer.
4. For the following statements will arr[3] and ptr[3] fetch the same character?  
   chararr[] = "IndiaBIX";  
   char \*ptr = "IndiaBIX";

|  |  |  |  |
| --- | --- | --- | --- |
| **a.** | Yes | **b.** | No |

1. Explain pointer to pointer with example.
2. Pointer is special kind of variable which is used to stored \_\_\_\_\_\_\_\_\_\_ of the variable.
3. Variable name
4. Data type
5. Address
6. Value
7. Pointer variable is declared using preceding \_\_\_\_\_\_\_\_\_ sign
8. &
9. \*
10. %
11. ^
12. Address stored in the pointer variable is of type \_\_\_\_\_\_\_\_\_\_.
13. Character
14. Integer
15. Float
16. Array
17. In order to fetch the address of the variable we write preceding \_\_\_\_\_\_\_\_\_ sign before variable name.
18. ,
19. \*
20. &
21. %
22. In which header file is the NULL macro defined ?
23. stdio.h
24. stddef.h
25. math.h
26. stdio.h & stddef.h

**Experiment No: 20**

**Aim:-Write an algorithm and C program to swap two numbers using call by reference.**

**Theory:-**

Actual parameters: The parameters that appear in function calls.

Formal parameters: The parameters that appear in function declarations.

For example: We have a function declaration like this:

int sum(int a, int b);

The a and b parameters are formal parameters.

We are calling the function like this:

int s = sum(10, 20); //Here 10 and 20 are actual parameters

or

int s = sum(n1, n2); //Here n1 and n2 are actual parameters

[**Call by value**](https://www.tutorialspoint.com/cprogramming/c_function_call_by_value.htm)

This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

[**Call by reference**](https://www.tutorialspoint.com/cprogramming/c_function_call_by_reference.htm)

This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

|  |  |
| --- | --- |
| **call by value** | **call by reference** |
| In *call by value*, a copy of actual arguments is passed to formal arguments of the called function and any change made to the formal arguments in the called function have no effect on the values of actual arguments in the calling function. | In *call by reference*, the location (address) of actual arguments is passed to formal arguments of the called function. This means by accessing the addresses of actual arguments we can alter them within from the called function. |
| In call by value, actual arguments will remain safe, they cannot be modified accidentally. | In *call by reference*, alteration to actual arguments is possible within from called function; therefore the code must handle arguments carefully else you get unexpected results. |

**Algorithm:**

Step 1: Start the program.

Step 2: Take input a and b.

Step 3: Call the function swap(a,b)

Step 3a: Start function.

Step 3b: Assign temp← &a

Step 3c: Assign &a ← &b

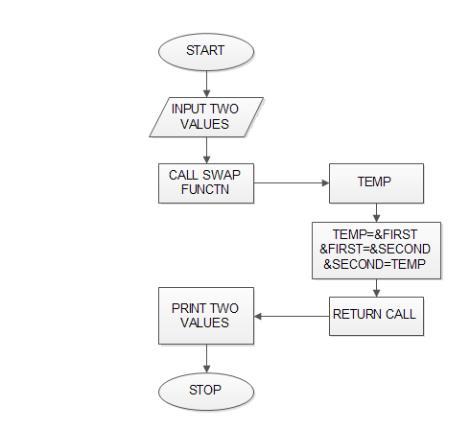
Step 3d: Assign &b ← temp

Step 3e: End function

Step 4: Print a and b.

Step 5: Stop the program.

**Flowchart:**



**Result**

………………………………………………………………………………………

…………………………………………………………………………………………

………………………………………………………………………………………….

**Industrical Application:**

1.To pass the variables to function using pass by reference scheme.

2. Provides effective way of implementing the [different data structures](http://c4learn.com/tutorials/data-structure/) such as tree,graph,linked list

**Conclusion**

**Questionnaire**

1. Which of the following is the correct syntax to send an array as a parameter to function?  
   a) func(&array); b) func(#array);  
   c) func(\*array); d) func(array[size]);
2. Comment on the following pointer declaration.

int \*ptr, p;

a) ptr is a pointer to integer, p is not

b) ptr and p, both are pointers to integer

c) ptr is a pointer to integer, p may or may not be

d) ptr and p both are not pointers to integerWhat is EOF ?

1. Comment on the following C statement.

const int \*ptr;

a) You cannot change the value pointed by ptr

b) You cannot change the pointer ptr itself

c) You May or may not change the value pointed by ptr

d) You can change the pointer as well as the value pointed by it

1. Which is an indirection operator among the following?  
   a) & b) \*  
   c) -> d) .
2. Which of following logical operation can be applied to pointers?  
   (Assuming initialization int \*a = 2; int \*b = 3;)  
   a) a | b b) a ^ b  
   c) a & b d) None of the mentioned
3. What will be the output of the following C code?

#include <stdio.h>

int main()

{

int ary[4] = {1, 2, 3, 4};

printf("%d\n", \*ary);

}

a) 1 b) Compile time error

c) Some garbage value d) Undefined variable

1. What are the elements present in the array of the following C code?

int array[5] = {5};

a)5, 5, 5, 5, 5  
b) 5, 0, 0, 0, 0  
c) 5, (garbage), (garbage), (garbage), (garbage)  
d) (garbage), (garbage), (garbage), (garbage), 5

1. Which of the following can never be sent by call-by-value?  
   a) Variable b) Array  
   c) Structures d) Both Array and Structures
2. Which of the following declaration will result in run-time error?  
   a) int \*\*c = &c;  
   b) int \*\*c = &\*c;  
   c) int \*\*c = \*\*c;  
   d) none of the mentioned
3. What will be the output of the following C code?

#include <stdio.h>

void main()

{

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

int \*p = a;

int \*r = &p;

printf("%d", (\*\*r));

}

a) 1

b) Compile time error

c) Address of a

d) Junk value