MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Banjara Hills, Hyderabad, Telangana

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

C Programming Laboratory Manual

Academic Year 2016-2017

Table of Contents

I Contents

II Programs

Part I

Contents

1. Vision of the Institution

To be part of universal human quest for development and progress by contributing high calibre, ethical and socially responsible engineers who meet the global challenge of building modern society in harmony with nature.

2. Mission of the Institution

- To attain excellence in imparting technical education from undergraduate through doctorate levels by adopting coherent and judiciously coordinated curricular and co-curricular programs.
- To foster partnership with industry and government agencies through collaborative research and consultancy.
- To nurture and strengthen auxiliary soft skills for overall development and improved employability in a multi-cultural work space.
- To develop scientific temper and spirit of enquiry in order to harness the latent innovative talents.
- To develop constructive attitude in students towards the task of nation building and empower them to become future leaders
- To nourish the entrepreneurial instincts of the students and hone their business acumen.
- To involve the students and the faculty in solving local community problems through economical and sustainable solutions.

3. Department Vision

To contribute competent computer science professionals to the global talent pool to meet the constantly evolving societal needs.

4. Department Mission

Mentoring students towards a successful professional career in a global environment through quality education and soft skills in order to meet the evolving societal needs.

5. Programme Education Objectives

1. Graduates will demonstrate technical skills and leadership in their chosen fields of employment by solving real time problems using current techniques and tools.

2. Graduates will communicate effectively as individuals or team members and be successful in the local and global cross cultural working environment.

3. Graduates will demonstrate lifelong learning through continuing education and professional development.

4. Graduates will be successful in providing viable and sustainable solutions within societal, professional, environmental and ethical contexts

6. Programme Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **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.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **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.
6. **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.
7. **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.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **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.
12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

7. Programme Specific Outcomes

The graduates will be able to:

PSO1: Demonstrate understanding of the principles and working of the hardware and software aspects of computer systems.

PSO2: Use professional engineering practices, strategies and tactics for the development, operation and maintenance of software

PSO3: Provide effective and efficient real time solutions using acquired knowledge in various domains.

8. Introduction

- Computer program is designed to solve problem. Nowadays it makes ease a lot of our works.
- The simple steps to find a solution to problems are the same steps used to write a program and basically can be defined as follows:
  1. Define the problem.
  2. Devise a plan to solve it.
  3. Implement the plan.
  4. Test the result to see whether the problem is solved.
- When creating a program in C/C++:
  1. Determine the objectives of the program.
  2. Decide which method the program will use to solve the problem while preparing the pseudocode or flowchart.
  3. Translate this pseudo code or flowchart into a computer program using the C/C++ language.
  4. Run and test the program.

The objective of this lab is to introduce the student to programming. This is done by writing several programs in C, C++ and discussing how to compile, link, and execute them on Linux. Programs 1-8 of the curriculum are to be developed in C language whereas 9-16 are to be developed in C++. It is assumed that the student has no prior knowledge of C/C++. However, the student must allocate significant time to read suggested materials. The student is expected to work on as many examples as possible.

Minimum Hardware Requirements

- 1 Ghz Pentium IV
- 256 MB RAM
- 1 GB Disk space

Minimum Software Requirements

- Operating System like Linux or Windows XP
- C++ compiler or Turbo C.

Part II

Programs

Program 1

Sinx and Cosx values using series expansion

Problem Definition

Finding out Sinx , cosx values using a C program.

Problem Objective

To practice the concept of loops.

Problem Description

We can evaluate sinx, cosx using the following series expansion.

Sinx = x – $x^3/3!$ + $x^5/5!$ – $x^7/7!$ + ...

Cosx = 1 – $x^2/2!$ + $x^4/4!$ – $x^6/6!$ + ...

SINE FUNCTION COMPUTATION

ALGORITHM:

```
STEP 1:  [START]
         BEGIN

STEP 2:  [INPUT NO: OF TERMS UP TO WHICH THE VALUE OF SIN(X) SHOULD BE
          EVALUATED IN SERIES \& X IN RADIANS]

         READ N,X.

STEP 3:  [SET UP INITIAL CONDITIONS FOR FIRST TERM THAT CANNOT BE
         CALCULATED ITERATIVELY]

         TSIN <- X
         TERM <- X.
         I <- 1
         ERROR <- 10-6
         COUNT <- 1
STEP 4:  [REPEAT]
         REPEAT STEP5, STEP6, STEP7
         UNTIL COUNT $<$= N AND TERM $<$ ERROR.

STEP 5:  [GENERATE NEXT TERM]

         TERM <- (-) TERM *X*/I*(I-1)
```

```
STEP 6:  [GENERATE THE CURRENT SUM]

          TSIN <- TSIN + TERM.

STEP7: [UPDATE THE VALUE OF I \& COUNT]

          I <- I + 2
          COUNT <- COUNT + 1
STEP 6: [STOP]
          END
```

NOTE:- Similarly calculate the value of cos(x) = 1-x^2/2!+x^4/4!-x^6/6!....
HINT : SUM=1, TERM=1.

Input:
For example x = 30 (in degrees), n = 10
Output:
Sinx = 0.50
Cosx = 0.666666

Validation/Result:
The program must take input 'x' in radians and then substitute the value in the equation.

FREQUENCY OF OCCURRENCE OF CHARACTERS AND SPECIAL CHARACTERS LIKE $n,$t, WHITE SPACESS

Problem Definition

The student has to determine the frequency of occurrence of characters and special characters in a piece of text.

Problem Objective:

Make the student practice the concept of Special Characters.

Problem Description

Initially the program takes count of words, spaces, Lines, Tabs & Characters as zero, we read the contents of the file character by character, based upon the character that is scanned we increase the count of the respective type of characters.

ALGORITHM

```
STEP 1: STORE THE INPUT TEXT IN A FILE

STEP 2: [INITIALIZE THE COUNTER]
         WORD <- 0, SPACE <- 0, LINES <- 0, TABS <- 0,
         CHARACTERS <- 0

STEP 3: [OPEN THE INPUT TEXT FILE IN READ MODE]
         READ IT CHARACTER BY CHARACTER AND UPDATE THE
         SPECIFIC COUNT BY COMPARING THE CHARACTERS AS
         FOLLOWS:-

         IF(CH=LINE)
         THEN
         BEGIN
              LINES <- LINES+1
              WORD <- WORD+1;
              CHARACTERS <- CHARACTERS+1
         END
         IF(CH='TAB')
         THEN
         BEGIN
              TABS <- TABS+1
              WORDS <- WORDS+1
              CHARACTERS <- CHARACTERS+1
```

```
                END
                IF  (CH=' ')
                THEN
                BEGIN
                     SPACES <- SPACES+1
                     WORDS <- WORDS+1
                     CHARACTERS <- CHARACTERS+1
                END

STEP 4:  [UPDATE CHARACTERS IN NEITHER OF ABOVE CASES]
         CHARACTERS <- CHARACTERS+1

STEP 5:  [CLOSE THE FILE USING THE FILE POINTER]
         CLOSE FILE_POINTER

STEP 6:  [PRINT THE COUNTS]
         PRINT WORDS, CHARACTERS, LINES, SPACES, TABS.

STEP 7: [STEP]
        END.
```

Input: $vi file1.txt

Output: $ Words 5 characters 10 lines 2 spaces 4 tabs 0

Conclusion: The desired results are achieved.

Program 3

Write a C program that performs Bubble Sort with an array.

Problem Definition

Sorting is defined as arrangement of set of elements in an order.

Problem Description

To arrange an array in ascending or descending order using bubble sort .

- Compare the adjacent elements in a list to see that the highest or lowest vale element in the list bubbles up to occupy its proper position.
- In ascending order Lowest element bubbles up to occupy one position high in the list with every iteration.
- Compare (a[I] > a[I+1]) exhauastively and if true exchange elements.
- Make n-I comparisons in each iteration , number of iterations = n-1 with exchanges
- In descending order highest element bubbles up to occupy one position high in the list with every iteration.
- With every iteration atleast one element gets sorted , therefore the number of comparisons gets reduced by 1 each time.

ALGORITHM:

```
STEP 1: [START]
          BEGIN
STEP 2: [INPUT VALUE OF NUMBER OF ELEMENTS IN THE ARRAY]
          READ N
          READ A[I] I=1,2,3,....N.
STEP 4: [INITIALIZE]
          I <- 1
STEP 5: [REPEAT]
          REPEAT THE STEPS STEP6, STEP7, STEP8 UNTIL I<=(N-1)
STEP 6: [INITIALIZE]
          J <- 1
STEP 7: [REPEAT]
          REPEAT STEP8 UNTIL(J<=N-1)
STEP 8: [COMPARE]
          IF(A[J]<A[J-1]
          BEGIN
              TEMP <- A[J]
```

```
            A[J] <- A[J+1]
            A[J+1] <- TEMP
         END
STEP 9:  [OUTPUT]
         PRINT THE SORTED ARRAY
STEP 10: [STOP]
         END.
```

Input: n = 6 (Size of Array)

56 23 90 14 67 12

(Before sort: a[0] = 56, a[1] = 23, a[2] = 90, a[3] = 14, a[4] = 67, a[5] = 12)

Output:

SORTED LIST: 12 14 23 56 67 90

(After sort: a[0] =12, a[1] =14, a[2] = 23, a[3] = 56, a[4] = 67, a[5] = 90)

Conclusion: The desired results are achieved.

Program 4

Write C programs to perform - a) Linear Search and b) Binary Search

Problem Definition

To search an element in a given list.

Problem Description

Linear Search:

The element to be searched is called search key(sk).

Compare the first element with the rest of the elements in the list one after the other. (a[i] is equal to sk, i = 0 to n)

If the sk matches with the array element then search is successful search otherwise search is unsuccessful.

Note: Best case: Sk found in first comparison.

Worst case: sk present at the end of the list then n-comparisons.

Average case: sk present in the middle of the list.

Binary search:

1. Read the size of array and search key(sk).
2. Sort the list.
3. Divide the list by determining the middle element.
4. If sk < mid-value search in the first half of the list discarding the other half from consideration.
5. If sk > mid-value search in the second half of the list discarding the other half from consideration.
6. Repeat dividing the list (further either the first half or the second half) until sk = mid-value.
7. If sk is found search is successful otherwise search is unsuccessful.

Note: Better performance compared to linear search but applicable only on sorted lists.

Input: n = 6 (Size of Array)

56 23 90 14 67 12
sk = 90

Output: Search key found at 3rd position in the list.

Program 5

Write a C program (with pointers) to perform - a. Matrix Addition b. Matrix Multiplication

Problem Definition

To write a program for Matrix Addition.

Problem Description

To practice the concept of loops, 2D arrays.

1. Read any two one dimensional arrays. (a and b)
2. Add corresponding index elements in the arrays.
3. Store the result in the corresponding index element in the result matrix

Allocate memory for every matrix using a = (int *)malloc(n * sizeof(int)); where int *a; Use similar declaration and statement of allocation of second matrix and result matrix. Any element is accessed by the following notation using pointers: *(a + i* n + j) where

- a is a pointer (int *a;)
- n = no: of columns of matrix (whose base address is a)
- Repeat until number of rows
- Repeat until the number of columns
- *(c+i*n+j) = *(a+i*n+j) +*(b+i*n+j)

ALGORITHM FOR ADDITION OF MATRICES :

```
STEP 1 : [START]
         BEGIN
STEP 2 : [READ NO : OF ROW & COLUMN]
         READ R,C
STEP 3 : [INPUT A MATRIX]
         READ MATRIX A
         FOR I=0 TO R
            FOR J=0 TO C
                READ A[I,J]

STEP 4 : [INPUT B MATRIX]
         READ MATRIX B
         FOR I=0 TO R
```

```
            FOR J=0 TO C
                READ B[I,J]

STEP 5 : [FIND SUM OF MATRICES AND STORE IN RESULT MATRIX C]
         FOR I=0 TO R
            FOR J=0 TO C
                C[I,J]=A[I,J]+B[I,J]

STEP 6 : [OUTPUT THE RESULT MATRIX 'C']
         FOR I=0 TO R
             FOR J=0 TO C
                  PRINT C[I,J]
STEP 7 : [STOP]
         END
```

Input:

```
       rows = 3
       cols = 3

     a =  |1  2  3|    b  =   |0 0 0|
          |4  5  6|           |0 0 0|
          |7  8  9|           |0 0 0|
```

Output:

```
        c =  |1  2  3|
             |4  5  6|
             |7  8  9|
```

b) Matrix Multiplication:

Problem Definition

To write a program for Matrix Multiplication:

Problem Description

1. Read any two 2-dimensional arrays. (a and b)
2. Multiply both matrices if no: of rows of a = no: of columns of b.
3. Multiply each row of first array with every column of the Second array.
4. Store the result in the corresponding index element in the result matrix.

Hint: Repeat for each row in first Matrix (j <= rows of a)
Repeat for each of column of second Matrix (j < = columns of b)
Repeat for each element in every column of second Matrix (k <= rows of a OR columns of b) *(c+i*n+j) = (c+i*n+j) + *(a+i*n+k) +*(b+i*n+j)

MULTIPLICATION OF MATRICES

ALGORITHM:

```
STEP 1: [START]
        BEGIN
STEP 2: [INPUT ORDER OF THE TWO MATRICES A & B]
        READ M, N, P, Q
STEP 3: [INPUT MATRIX A & MATRIX B]
        FOR  I = 0  TO  M
             FOR  I = 0  TO  N
                  READ A
        FOR  I = 0  TO  P
             FOR  I = 0  TO  Q
                  READ B
STEP 4: [FIND THE PRODUCT OF A & B]

        FOR  I = 0  TO   M
            FOR  I = 0  TO   N
                  C [I , J ] = 0

                  FOR  K = 0   TO  N
                      *(C+I*N+J) = (C+I*N+J) + *(A+I*N+K) +*(B+I*N+J)

STEP 5: [OUTPUT THE RESULT MATRIX]
        FOR  I = 0  TO  M
```

```
            FOR  I = 0  TO  Q
                 PRINT C [ I , J ]

STEP 6: [STOP]
        END.
```

Input:

```
      rows = 3
      cols = 3

      a =  |1  2  3|     b   =   |1 0 0|
           |4  5  6|             |0 1 0|
           |7  8  9|             |0 0 1|
```

Output:

```
      c  =  |1  2  3|
            |4  5  6|
            |7  8  9|
```

Conclusion: The desired results are achieved.

Program 6

GENERATION OF ADDRESS LABELS USING STRUCTURES

Problem Definition

Display the name, age, designation, address & pincode of a person.

Problem Description

To practice the concept of structures.

ALGORITHM :

```
STEP 1:  [DEFINE A STRUCTURE TYPE THAT CONTAINS  PERSON NAME,
         AGE, DESIGNATION, ADDRESS AND PIN CODE].

         RECORD PERSON
         BEGIN
              P_NAME
              P_DESIG
              P_ADDRESS
              P_PIN
         END

STEP 2:  [READ THE NUMBER OF LABELS TO BE GENERATED.
         INFORMATION FOR EACH PERSON].
         READ N

STEP 3: [USING THIS STRUCTURE READ THE INFORMATION OF EACH PERSON]
STEP 4: PRINT THE LABEL OF EACH PERSON
        A)  ONE BY ONE AND
        B)  PRINT TWO LABELS AT A TIME.

       /*Pseudo Code*/
       Structure is only a template to represent information we can
 declare the structure variables as following

 struct tag name V1,V2;

       Accessing members in a structure:-

       Ex: struct person
       {
             char name[15];
             char desig[15];
```

```
        int age;
        float salary;
 }p;
 p .name = "xyz";
 p.desig = "teacher";
 p.age = 25;
 p.salary = 10000.00;

scanf( ) can be used to read the values like
scanf(\%s",p.name);
scanf(\%d",&p.age);

Array of structures:
struct person p1[10];
for ( i=0; i<10; i++)
{
      Scanf(\%s",p1[i] . title);
}
```

Conclusion: The desired results are achieved.

Program 7

SEQUENTIAL FILE OPERATIONS

Problem Definition

A file whose contents are accessed sequentially is called a sequential acess file.

Problem Description

To practice the concepts of Sequential Files.

A file is created and saved. Then it is accessed through a C program. Various operations like opening a file, copying the contents of a file in to another file, displaying the contents of a file on a monitor, merging two files into a new file can be done.

ALGORITHM :

```
STEP 1: [START]
        BEGIN

STEP 2: [OPEN A FILE IN WRITE MODE WITH THE NAME \INPUT"]
        OPEN THE FILE WITH FILE_POINTER F1.
        (  USE:   f1 = fopen (\input" , \w" );    )

STEP 3: [WRITE DATA INTO THE FILE BY READING FROM KEYBOARD]
        UNTIL IF IS NOT END OF FILE.

STEP 4: [CLOSE THE FILE]
        CLOSE THE FILE WITH FILE_POINTER F1

STEP 5: [OPEN THE SAME FILE IN READ MODE]
        F1 = FOPEN( \ INPUT \ , "R" ) ;

STEP 6: [OUTPUT ITS CONTENTS TO THE SCREEN]
        PRINT DATA FROM FILE USING FORMATTED OR UNFORMATTED FUNCTIONS
        ( USE: printf( ), puts( ), putch( ) )


STEP 7: [CLOSE THE FILE]
        CLOSE THE FILE WITH FILE_POINTER F1

STEP 8: [STOP]
        END.

Note: Similar Programs can be implemented to copy a file into another and
      compare any two files.
```

Finding Maximum, Minimum and Sum of given set of Numbers Using C++.

Problem Description

To apply the cin, cout commands and arrays.

ALGORITHM :

```
STEP 1  : [ START ]
          BEGIN

STEP 2  : [INPUT THE TOTAL NUMBER OF ELEMENTS IN LIST]
          READ N
STEP 3  : [INPUT THE FIRST ELEMENT IN THE LIST]
          READ FIRST
STEP 4  : [SET TEMPORARY MAXIMUM]
          MAX = FIRST
          COUNT = 1

STEP 5  : [REPEAT]
          REPEAT STEPS STEP6, STEP7, UNTIL COUNT <N.

STEP 6  : [READ NEXT ELEMENT IN THE LIST]
          READ NEXT
          COUNT <- COUNT + 1

STEP 7  : [COMPARE NEXT ELEMENT WITH PREVIOUS MAXIMUM
          NUMBER AND UPDATE MAXIMUM]

          IF (NEXT > MAX)
          MAX <- NEXT
STEP 8  : [OUTPUT THE MAXIMUM ELEMENT]
          PRINT MAX
 STEP 9 : [STOP]
          END.
```

Class Structure:

```
class SumMaxMin
{
     private :
          int a[10];
          int n;
     public :
          void readElements ( ) ;
          int getMax ( ) ;
          int getMin ( ) ;
          int getSum ( ) ;
}
```

NOTE: Similarly calculate the minimum number using NEXT¡ MIN (IN PLACE OF MAX)

DESCRIPTION:

Data Members:
int a[10] :- for storing n numbers.
int n :- for storing size value.
Member Functions :

1. void readElements ( ):- This method is responsible for reading value of size and elements from the user and storing in "n" and in array "a" respectively.
2. int getMax ( ) :- Responsible for returning maximum value of given "n" numbers.
3. int getMin ( ) :- Responsible for returning minimum value of given "n" numbers.
4. int getSum ( ):- Responsible for returning Sum value of given "n" numbers.

Conclusion: The desired results are achieved.

Program 9

Implementation of a matrix class Using C++.

Problem Description

To practice the concepts of classes.

Class Structure:

```
class  Matrix
{
     private :
          int a[10] [10];
          int rows, columns;
     public:
          Matrix ( int , int ) ;
          void readElements ( ) ;
}
```

DESCRIPTION:

Data Members:

1. int a[10] [10] :- Two-Dimensional Array used to store matrix elements.
2. int rows, columns :-Variables used to store the order of Matrix.

Member Functions:

1. Matrix (int , int) :- Constructor with 2 arguments which accepts row and column value. Optionally, we can have default constructor and a setOrder (int , int) function that can be used to set the order of matrix.
2. void readElements ( ) :- Resposible for accepting elements from user and storing it in array "a".
3. void displayElements ( ) :- Resposible for Displaying elements of matrix stored in array "a".
4. Matrix operator+ (Matrix) :- operator '+' being overloaded to work with Matrix objects.
5. Matrix operator* (Matrix) :- operator '*' being overloaded to work with Matrix objects.

Program 10

Classes for Bank Account, Student information, Library catalog.

a) Class Structure for Student:

```
class Student
{
     private :
          int rollno ;
          char name [ 25 ] ;
          char branch[ 25 ] ;
     public :
          void setDetails ( int , char * , char * ) ;
          void getDetails ( ) ;
};
```

b) Class Structure for BankAccount :

```
class BankAccount
{

     private :
          int accno;
          char name[25];
          float balance;
     public :
          BankAccount ( int , char * , float = 0.0 ) ;
          void deposit ( float dAmt ) ;
          void withdraw ( float wAmt ) ;
          float getBalance ( ) ;
};
```

c) Class Structure for Library :

```
class Library
{
     private :
          int cardno ;
          char name [ 25  ;
          char slot [ 2 ] [ 30 ] ;

/* Assuming two book slots are allowed on a card. A book name indicates
```

```
slot is used otherwise A null value indicates slot is empty.*/

     public :

          Library ( int , char * ) ;
          bool issueBook ( char *bookName ) ;
          bool returnBook ( int slotNumber ) ;
// slotNumber can take either a value 0 or 1.
          int isSlotAvailable ( );
 /* if not returns -1, if there is then returns 0 or 1 is return based on
which slot is available.*/
          void showStatus ( ) ; // Display status of slots.
};
```

Program 11

Create a Complex Class with Operator Overloading.

Problem Description

To apply the concepts of operator overloading.

Class Structure:

```
class Complex
{
     private :
          int real ;
          int imaginary ;
     public:
          void setNumber ( int , int ) ;
          void display( );
          Complex operator+ ( Complex ) ;
          Complex operator- ( Complex ) ;
          Complex operator* ( Complex ) ;
};
```

DESCRIPTION :

Data Members :

1. int real :- Variable used to store real value of a complex number.
2. int imaginary :- Variable used to store imaginary value of a complex number.

Member Functions :

1. void setNumber (int , int) :- Responsible for forming a complex number.
2. void display( ) :- Responsible for displaying a complex number.
3. Complex operator + (Complex) :- Overloading '+' operator to add two Complex objects (numbers).
4. Complex operator - (Complex) :- Overloading '-' operator to subtract two Complex objects (numbers).
5. Complex operator * (Complex) :- Overloading '*' operator to subtract two Complex objects(numbers).

Conclusion : The desired results are achieved

Program 12

Creation of inheritance hierarchy for graphic shapes.

Problem Description

To apply the concepts of inheritance.

Class Structure:

```
class Shape
{
     public:
     virtual void area( ) = 0;
};

class Square : public Shape
{
     private:
          int side;
     public:
          Square ( int =10 ) ;
          void area ( )
          {
               cout<<side*side;
          }
};
```

DESCRIPTION :

- This program is based on the concept of run-time polymorphism. (Function Overriding).
- In this method, C++ compiler executes a function by selecting the it at the time of function call (run-time).
- When the signatures of any two functions are same in the base class and derived class then the C++ compiler will be in an ambiguity to select the function from base class or the derived class.
- The solution is to use a pointer to decide at run-time which method is called. The problem to the suggested solution is that the pointer of a base class always refers to the method inside base class whereas a pointer to the derived class can never refer to the base class method (since the hierarchy of calls is to be maintained)

- Therefore the possible solution is to use the concept of function overriding in which the base class method is declared with the keyword "virtual" and the pointer of the base class is then allowed to refer to the derived class method with the same pointer.

```
main ( )
{

     Shape *p;

     Square s;

     p = &s ;

     p -> area( ) ; // calls area( ) of Square class.

     Rectangle r ;

     p = &r ;

     p -> area ( );   // calls area( ) of Rectangle class.

}
```

Conclusion : The desired results are achieved.

Program 13

Template functions for min( ) and max( ) for finding minimum and maximum in a list of numbers.

Problem Description

To practice the concepts of function templates.

Class Structure:

```
template<class T>
class MaxMin
{
     private :
          T a[10];
          int n;
     public :
          void readElements();
          T max();
          T min();
};
```

DESCRIPTION :

Data Members :

1. T a[10] :- Array a for storing n numbers of type based on template.
2. int n :- n for storing array size value.

Member Functions :

1. void readElements ( ) :- Responsible for Accepting size value and that many elements from the user and storing it in "n" and in array "a" respectively.
2. T max ( ) :- Responsible for returning maximum value from given "n" numbers.
3. T min ( ) :- Responsible for returning minimum value from given "n" numbers.

Conclusion : The desired results are achieved.

Program 14

Finding number of characters, words and sentences in a given text.

Class Structure :

```
class Text
{
     private :
          char str [ 100 ] ;
     public :
          void setText ( char * ) ;
          void displayText ( ) ;
          int countCharacters ( ) ;
          int countWords ( ) ;
          int countSentences ( ) ;
};
```

DESCRIPTION :

Data Members :
char str [100] :- Used for storing given text.

Member Functions :

1. void setText (char *) :-Responsible for setting accepted string into str.
2. void displayText ( ) :- Resposible for displaying the text whose nature is being determined.
3. int countCharacters ( ) :-Responsible for Counting characters in given string "str" and returning that value.
4. int countWords ( ) :- Responsible for Counting Words in given string "str" and returning that value.
5. int countSentences ( ) :- Responsible for Counting Sentences in given string "str" and returning that value.

Conclusion : The desired results are achieved.

Program 15

Program to Write Error Handling Code using Exceptions

Problem Description

To apply the concept of exception handling in C++ programming.

Class Structure :

```
try
{
     c=a/b;
}

throw c;
Catch(type c)
{

}
```

Conclusion : The desired results are achieved.