Object Oriented Programming

Lab File

By ARYAN SINGH DHIMAN

UE213020

**INDEX:**

|  |  |
| --- | --- |
| **Program** | **Page No.** |
| WAP to find Largest, Second Largest and smallest among three numbers | 3 |
| WAP to interchange values of two numbers without using a third number: | 5 |
| WAP to show concept of type casting in C++ | 5 |
| WAP to print ASCII value of character entered by the user | 6 |
| WAP to find sum of first and last digit of the number entered by the user | 7 |
| WAP to print Fibonacci Series | 8 |
| WAP to find reverse of a given Integer number: | 9 |
| Write a C++ program to identify if an input is symbol, character, or digit | 10 |
| WAP a program to find prime numbers in given range n | 10 |
| Write a program that uses functions to :  a) Swap two integers  b) Swap two characters | 11 |
| WAP to print factorial of a number using recursion | 13 |
| WAP to convert decimal number to its equivalent binary number using recursion | 14 |
| WAP to find largest and smallest number from a given array | 15 |
| WAP to perform following sorting techniques:   1. Bubble Sort 2. Selection Sort | 16 |
| WAP to perform following sorting techniques:   1. Bubble Sort 2. Selection Sort | 18 |
| Write a C++ program that counts number of days till the date entered by the user | 21 |
| WAP to add two matrices of two-dimensional arrays using function | 23 |
| WAP to insert an element at Kth position in a list of n elements | 25 |
| WAP to implement the concept of nesting of classes | 27 |
| WAP to illustrate concept of static data members and functions | 28 |
| WAP to find average value using friend function | 29 |
| WAP to demonstrate use of default constructor and destructor | 31 |
| WAP to demonstrate use of parameterized constructor | 32 |
| WAP to demonstrate use of copy constructor and also show order of constructor and destructor invocation | 33 |
| WAP to implement usage of function overloading | 35 |
| WAP to illustrate the concept of unary operator overloading | 36 |
| WAP to calculate sum of two complex numbers by using operator overloading | 37 |
| Write a C++ that overloads the + operator and relational operators to perform the following operations:   1. Concatenation of two strings 2. Comparison of two strings | 38 |
| WAP to determine if given string is palindrome or not | 41 |
| WAP to use function templates to solve problems | 42 |
| WAP to illustrate how the following forms of inheritance are supported:   1. Single Inheritance 2. Multiple Inheritance 3. Multilevel Inheritance 4. Hierarchical Inheritance | 45 |
| WAP that illustrates how run time polymorphism is achieved using virtual functions | 47 |
| WAP to illustrate the use of pure virtual function | 49 |
| Write a C++ program that illustrates use of virtual base class in building class hierarchy | 50 |
| Write a C++ program that illustrates use of abstract class in building class hierarchy | 51 |
| Write a C++ program to display contents of a text file | 52 |
| Write a C++ program which copies one file to another | 53 |
| Write a C++ program that counts number of characters, lines and words in a text file | 54 |

**1.WAP to find Largest, Second Largest and smallest among three numbers:**

* This program takes three integers as input and and outputs the largest,second largest and smallest number.
* First we intialize an integer array of size three.
* Then using the stream insertion operator we take three integers as input from the user and store it in the array.
* Then we compare the first and second numbers and then compare the larger of the two with the third number.This way we can find largest ,second largest and smallest number.

Source Code:

#include <iostream>

#include <array>

using namespace std;

int main()

{

    array<int,3> arr;

    cout<<"Enter three numbers: ";

    for(int i=0;i<3;i++)

    {

        cin>>arr.at(i);

    }

    if(arr.at(0)>arr.at(1))

    {

        if(arr.at(0)>arr.at(2))

        {

            cout<<"Largest Number is : "<<arr.at(0)<<endl;

            if(arr.at(1)>arr.at(2))

            {

                cout<<"Second Largest Number is : "<<arr.at(1)<<endl;

                cout<<"Smallest Number is : "<<arr.at(2)<<endl;

            }

            else

            {

                cout<<"Second Largest Number is : "<<arr.at(2)<<endl;

                cout<<"Smallest Number is : "<<arr.at(1)<<endl;

            }

        }

        else

        {

            cout<<"Largest Number is : "<<arr.at(2)<<endl;

            cout<<"Second Largest Number is : "<<arr.at(0)<<endl;

            cout<<"Smallest Number is : "<<arr.at(1)<<endl;

        }

    }

    else

    {

        if(arr.at(1)>arr.at(2))

        {

            cout<<"Largest Number is : "<<arr.at(1)<<endl;

            if(arr.at(0)>arr.at(2))

            {

                cout<<"Second Largest Number is : "<<arr.at(0)<<endl;

                cout<<"Smallest Number is : "<<arr.at(2)<<endl;

            }

        }

        else

        {

            cout<<"Largest Number is : "<<arr.at(2)<<endl;

            cout<<"Second Largest Number is : "<<arr.at(1)<<endl;

            cout<<"Smallest Number is : "<<arr.at(0)<<endl;

        }

    }

    return 0;

}

Output:

Text

Description automatically generated

**2.WAP to interchange values of two numbers without using a third number:**

* The user is asked to enter two numbers.
* The numbers are stored in two variables ‘a’ and ‘b’.
* ‘a’ is assigned the sum of ‘a’ and ‘b’
* ‘b’ is assigned the difference of ‘a’ and ‘b’.As ‘a’ is ‘a+b’ the value originally in ‘a’ is assigned to ‘b’.
* Then ‘a’ is again assigned the difference of ‘a’ and ‘b’.Like in previous step ‘a’ is assigned original value of ‘b’.

Source Code:

#include <iostream>

using namespace std;

int main()

{

    cout<<"Enter two numbers\n";

    int a=0,b=0;

    cin>>a>>b;

    a=a+b;

    b=a-b;

    a=a-b;

    cout<<"The numbers after swapping are: "<<a<<" and "<<b;

    return 0;

}

Output:

Text

Description automatically generated

**3.WAP to show concept of type casting in C++:**

* The User is asked to enter two floating point numbers.
* These floats are then typecasted into integers using both implict and explicit typecasting.
* They are then outputted to the console.

Source Code:

#include <iostream>

using namespace std;

int main()

{

    float a,b;

    int c{0},d{0};

    cout<<"Enter two Floating Point Numbers: ";

    cin>>a>>b;

    cout<<"Numbers after Typecasting are: \n";

    c=(int)a;*//explicit*

    d=(b);*//impiicit*

    cout<<"Explicit typecast: "<<c<<'\n';

    cout<<"Implicit typecast: "<<d<<'\n';

    return 0;

}

Output:

Graphical user interface, text

Description automatically generated

**4.WAP to print ASCII value of character entered by the user:**

* The user is asked to enter a character.
* Then its ASCII value is outputted by typecasting the character variable.

Source Code:

#include <iostream>

using namespace std;

int main()

{

    char ch;

    cout<<"Enter character\n";

    cin>>ch;

    cout<<"ASCII value of inputted character is : "<<(int)ch;

    return 0;

}

Output:

A picture containing graphical user interface

Description automatically generated

**5.WAP to find sum of first and last digit of the number entered by the user:**

* The user is asked to enter a number.
* Then using a loop, the digits of the character are extracted using the modulus operator (%) one by one.
* Then the last and first digits are added and outputted to the console.

Source Code:

#include <iostream>

using namespace std;

int main()

{

    cout<<"Enter Number\n";

    int number{0};

    cin>>number;

    int temp{0},sum{0};

    sum=number%10;

    for(;;)

    {

        temp=number%10;

        number=number/10;

        if(!number)

        {

            break;

        }

    }

    sum=temp+sum;

 cout<<"Sum of first and last digit is: ";

    cout<<sum<<endl;

    return 0;

}

Output:

A picture containing text

Description automatically generated

**6.WAP to print Fibonacci Series:**

* The user is asked to enter the number up to which they want to print the Fibonacci sequence.
* The first and second term of the sequence (0 and 1 respectively) are added to get the third term which is added to the second term and so on till we get to a number greater than the inputted number.
* The numbers thus found are printed to the screen.

Source Code:

#include <iostream>

using namespace std;

int main()

{

    int a=0,b=1,c,i;

    cout<<"Please enter the number upto which you want the Fibonacci Sequence to be printed out\n";

    cin>>i;

    cout<<a<<" "<<b<<" ";

    for(;;)

    {

        c=a+b;

        if(c>i)

        {

            break;

        }

        cout<<c<<" ";

        a=b;

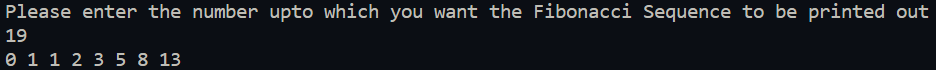
        b=c;

    }

    return 0;

}

Output:



**7.WAP to find reverse of a given Integer number:**

* The user is asked to input the number which they want to be reversed.
* First the length of the number is determined.
* Then the digits are extracted by multiplied by the appropriate power of ten and summed up.
* Then the reverse of the number is outputted.

Source Code:

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

    cout<<"Enter the number to be reversed\n";

    int num;

    cin>>num;

    int temp=num;

    int rev{0};

    int len=0;

    for(int k=0;temp!=0;k++)

    {

        temp=temp/10;

        len++;

    }

    temp=num;

    for(int i=0;temp!=0;i++)

    {

        rev=rev+(temp%10)\*pow(10,len-i-1);

        temp=temp/10;

    }

    cout<<rev;

    return 0;

}

Output:

A picture containing graphical user interface

Description automatically generated

**8.Write a C++ program to identify if an input is symbol, character, or digit:**

* The user is asked to input either a character, digit, or a symbol.
* Then using the functions in ‘cctype’ header file the input is checked and the relevant type is outputted.

Source Code:

#include <iostream>

#include <cctype>

using namespace std;

int main()

{

    cout<<"Enter character/digit/symbol: ";

    char ch;

    cin>>ch;

    if(isalpha(ch))

        cout<<"You have entered a Character";

    else if(isdigit(ch))

        cout<<"You have entered a Number";

    else

        cout<<"You have entered a Special Symbol";

    return 0;

}

Output:



**9.WAP a program to find prime numbers in given range n:**

* This program finds the prime number in the range inputted by the user.
* A loop is initiated which starts from 2 till n in which the number is divided by the numbers preceding it repeatedly to find if its prime.
* A prime if found is outputted to the screen.

Source Code:

#include <iostream>

using namespace std;

int main()

{

    int n{0};

    cout<<"Enter upper limit: ";

    cin>>n;

    cout<<"2 ";

    for(int i=2;i<=n;i++)

    {

        for(int j=2;j<i;j++)

        {

            if(i%j==0)

                break;

            else if(j=i-1)

                cout<<i<<" ";

        }

    }

    return 0;

}

Output:



**10.Write a program that uses functions to :**

**a) Swap two integers**

**b) Swap two characters**

* The user is asked to enter two integers.
* Then those numbers are passed to a function which takes two integer references as arguments .
* In the function, a temporary variable is assigned the value of one of the integers and then the integer whose value has been stored in temporary variable is assigned value of the other integer.
* The other integer is assigned the value of the temporary variable and thus the numbers are swapped.
* Similarly two floating point integers are swapped.

Source Code:

#include <iostream>

using namespace std;

void swap\_int(int &a,int &b);

void swap\_char(char &c,char &b);

int main()

{

    int num\_1,num\_2;

    char char\_1,char\_2;

    cout<<"Please enter two integers"<<endl;

    cin>>num\_1>>num\_2;

    cout<<"Before Swapping: First Integer is: "<<num\_1<<" ,Second Integer is: "<<num\_2<<endl;

    swap\_int(num\_1,num\_2);

    cout<<"After Swapping: First Integer is: "<<num\_1<<" ,Second Integer is: "<<num\_2<<"\n"<<endl;

    cout<<"Please enter two characters"<<endl;

    cin>>char\_1>>char\_2;

    cout<<"Before Swapping: First Character is: "<<char\_1<<" ,Second Character is: "<<char\_2<<endl;

    swap\_char(char\_1,char\_2);

    cout<<"After Swapping: First Character is: "<<char\_1<<" ,Second Character is: "<<char\_2<<endl;

    return 0;

}

void swap\_int(int &num\_1,int &num\_2)

{

    int temp{num\_1};

    num\_1=num\_2;

    num\_2=temp;

}

void swap\_char(char &char\_1,char &char\_2)

{

    char temp{char\_1};

    char\_1=char\_2;

    char\_2=temp;

}

Output:

Text

Description automatically generated

**11.WAP to print factorial of a number using recursion:**

* The user is asked to enter the number for which they want factorial of.
* The number is passed to a function. The function returns one if the argument passed to it is one else returns the argument multiplied with the return value of factorial function with the argument being current number decremented.
* After the factorial is found it is outputted to the console.

Source Code:

#include <iostream>

using namespace std;

int rec\_fac(const int num);

int main()

{

    cout << "Enter Number for which you want Factorial: ";

    int num;

    cin>>num;

    cout<<"Factorial of "<<num<<" is: "<<rec\_fac(num);

    return 0;

}

int rec\_fac(const int num)

{

    if(num==1)

        return num;

    else

        return num\*rec\_fac(num-1);*//multiplies num by integers less than it till it reaches 1*

}

Output:

Text

Description automatically generated with medium confidence

**12.WAP to convert decimal number to its equivalent binary number using recursion:**

* This program converts the inputted number from decimal to binary.
* It recursively converts the number.
* The user inputted number is passed as argument to the function.
* First if the argument is zero then zero is returned and if argument is one then one is returned.
* Else the sum of the remainder of argument with two and the return value of the function with half of the current argument passed times ten is returned.
* The binary number thus found is outputted.

Source Code:

#include <iostream>

using namespace std;

long int dec\_to\_bin(int num);

int main()

{

    int num;

    cout<<"Enter Decimal Number which you want converted to binary: ";

    cin>>num;

    cout<<"Binary form is : "<<dec\_to\_bin(num);

    return 0;

}

long int dec\_to\_bin(int num)

{

    if(num==0)

        return 0;

    else if(num==1)

        return 1;

    else

        return (num%2)+10\*dec\_to\_bin(num/2);*//finds remainder and then adds 10 times remainder of half of input*

}

Output:



**13.WAP to find largest and smallest number from a given array:**

* The array of integers inputted by the user is sorted by using bubble sort.
* In bubble sort the first two numbers of the array are compared and swapped if first number is greater than second. Then the next two integers are checked similarly. If end of array is reached the first two numbers are again selected and this is done till array is sorted.
* Then from the sorted array the largest and smallest numbers are outputted.

Source Code:

#include <iostream>

#include <array>

using namespace std;

int main()

{

    int n;

    cout<<"Enter the size of array : ";

    cin>>n;

    int arr[n];

    cout<<"Enter the numbers\n";

    for(int i=0;i<n;i++)

    {

        cin>>arr[i];

    }

    for(int l=0;l<n-1;l++)*//bubble sort*

    {

        if(arr[l]>arr[l+1])

        {

        int temp=arr[l];

        arr[l]=arr[l+1];

        arr[l+1]=temp;

        l=-1;

        }

    }

    cout<<"\nLargest number is: "<<arr[n-1]<<" ,Smallest Number is: "<<arr[0]<<endl;

    return 0;

}

Output:

A screenshot of a computer

Description automatically generated with medium confidence

**14.WAP to perform following sorting techniques:**

1. **Bubble Sort**
2. **Selection Sort**

* The numbers inputted by the user are stored in an array.
* In Bubble sort the first two numbers of the array are compared and swapped if first number is greater than second. Then the next two integers are checked similarly. If end of array is reached the first two numbers are again selected and this is done till array is sorted.
* In Selection sort the smallest number of the array is found and is put at the starting position.Then the second smallest number is found and it is put at the second position by swapping. This is done till last number of array is selected.

Source Code:

#include <iostream>

using namespace std;

void bubble\_sort(int arr[],int size\_of\_array);

void selection\_sort(int arr[],int size\_of\_array);

int main()

{

    cout << "Enter number of elements in the array to be sorted: ";

    int num;

    cin>>num;

    int \*arr\_1=new int[num];

    int \*arr\_2=new int[num];

    cout<<"Enter elements of the array"<<endl;

    for(int i=0;i<num;i++)

    {

        cin>>arr\_1[i];

        arr\_2[i]=arr\_1[i];

    }

    cout<<"\nArray after bubble sort"<<endl;

    bubble\_sort(arr\_1,num);

    for(int i=0;i<num;i++)

    {

        cout<<arr\_1[i]<<" ";

    }

    selection\_sort(arr\_2,num);

    cout<<"\n\nArray after selection sort"<<endl;

    for(int i=0;i<num;i++)

    {

        cout<<arr\_2[i]<<" ";

    }

    delete [] arr\_1;

    delete [] arr\_2;

    return 0;

}

void bubble\_sort(int arr[],int size\_of\_array)

{

    for(int l=0;l<size\_of\_array-1;l++)*//bubble sort*

    {

        if(arr[l]>arr[l+1])

        {

        int temp=arr[l];

        arr[l]=arr[l+1];

        arr[l+1]=temp;

        l=-1;

        }

    }

}

void selection\_sort(int arr[],int size\_of\_array)

{

    int checker=1;

    for(int n=0;;n++)

    {

        checker=0;

        int minimum=arr[n],position=n;

        for(int t=n+1;t<size\_of\_array;t++)*//selecting element of minimum size and finding its position*

        {

            if(arr[t]<minimum)

            {

                minimum=arr[t];

                position=t;

                checker++;

            }

        }

        if(checker==0)

            break;

        arr[position]=arr[n];*//swapping minimum with starting position*

        arr[n]=minimum;

    }

}

Output:

Text

Description automatically generated

**15.WAP to search an item using:**

**a)Linear Search**

**b)Binary Search**

* In linear search the elements of the array are compared one by one till the element is found.
* In binary search the a sorted array is used.The element to be found is first compared with the element in the middle if it matches the the position is returned.
* Else if the element to be found is greater than the middle element then we select the array on the right side of middle element else we select the array of left side of middle element.
* We repeat the above steps again with the selected portion of the array.

Source Code:

#include <iostream>

using namespace std;

int linear\_search(const int source[],int length,int element);

int binarySearch(int source[],int length,int element);

void bubble\_sort(int arr[],int size\_of\_array);

int main()

{

    cout<<"Enter length of array: ";

    int len;

    cin>>len;

    cout<<"Enter elements of the array: ";

    int \*arr=new int[len];

    for(int i=0;i<len;i++)

    {

        cin>>arr[i];

    }

    cout<<"Enter element to be searched: ";

    int element\_find;

    cin>>element\_find;

    bubble\_sort(arr,len);*//array needs to be sorted in case it has to be used with binary search*

    cout<<"\nBy Linear Search\n\n";

    int temp=linear\_search(arr,len,element\_find);

    if(temp!=(-1))

        cout<<"Element "<<element\_find<<" is at the position :"<<temp+1<<endl;

    else

        cout<<"Element not found\n";

    cout<<"\nBy Binary Search\n\n";

    int temp\_2=binarySearch(arr,len,element\_find);

    if(temp\_2!=(-1))

        cout<<"Element "<<element\_find<<" is at the position :"<<temp\_2+1<<endl;

    else

        cout<<"Element not found\n";

    delete [] arr;

    return 0;

}

int linear\_search(const int source[],int length,int element)

{

    int i=0;

    for(i=0;i<length;i++)*//goes through the array one by one and compares each element with with the element to be found*

    {

        if(source[i]==element)

        {

            return i;

        }

    }

    if(i==(length))

        return -1;

    return 0;

}

int binarySearch(int source[],int length,int element)

{

    int end\_of\_array,start\_of\_array,middle\_of\_array;

    end\_of\_array=length-1;

    start\_of\_array=0;

    while(start\_of\_array<=end\_of\_array)

    {

        middle\_of\_array=(end\_of\_array+start\_of\_array)/2;*//finds the middle element of the array*

        if(source[middle\_of\_array]==element)*//if middle element is equal to element to be found returns middle element*

            return middle\_of\_array;

        else if(source[middle\_of\_array]>element)

*//if middle element is greater than element to be found then the left half of the array is taken in next iteration*

            end\_of\_array=middle\_of\_array-1;

        else*//if middle element is smaller than right half of the array is taken in next iteration*

            start\_of\_array=middle\_of\_array+1;

    }

    return -1;

}

void bubble\_sort(int arr[],int size\_of\_array)

{

    for(int l=0;l<size\_of\_array-1;l++)*//bubble sort*

    {

        if(arr[l]>arr[l+1])

        {

        int temp=arr[l];

        arr[l]=arr[l+1];

        arr[l+1]=temp;

        l=-1;

        }

    }

}

Output:

Text

Description automatically generated

**16.Write a C++ program that counts number of days till the date entered by the user:**

* The user is asked to input two dates.
* Then for the smaller date number of days till the next year is found.
* For the larger date number of days from the start of the year of the larger date is found.
* Then the number of days between the two years is found.
* These three results are added to find the number of days between the two dates.

Source Code:

#include<iostream>

using namespace std;

int leap\_yr(int);

void exchange(int &,int &);

int main()

{

    int d1,m1,y1,d2,m2,y2,s,d;

    cout<<"Enter the first date "<<endl;

    cin>>d1>>m1>>y1;

    cout<<"Enter the second date "<<endl;

    cin>>d2>>m2>>y2;

    if (y1>y2)

    {

        exchange(d1,d2);

        exchange(m1,m2);

        exchange(y1,y2);

    }

    int arr[12]={31,28,31,30,31,30,31,31,30,31,30,31};

    s=0;

    for (int i=m1; i<=11; i++)

        s=s+arr[i];

    if (m1<=2)

    {

        if (leap\_yr(y1)==1)

            s++;

    }

    d= arr[m1-1]-d1;

    int t1=s+d;

    s=0,d=0;

    for (int i=0; i<=m2-2;i++)

        s=s+arr[i];

    if (m2>2)

    {

        if(leap\_yr(y2)==1)

            s++;

    }

    d=d2;

    int t2=s+d;

    int f=0;

    for(int i= y1+1; i<y2;i++)

    {

        f=f+365;

        if(leap\_yr(i)==1)

            f++;

    }

    cout<<"Total number of days are "<<t1+t2+f;

}

int leap\_yr (int y)

{

    if (y%400==0)

        return 1;

    else if (y%100==0)

        return 0;

    else if (y%4==0)

        return 1;

    else

        return 0;

}

void exchange (int &a,int &b)

{

    int t;

    t=a;

    a=b;

    b=t;

}

Output:

Text

Description automatically generated with medium confidence

**17.WAP to add two matrices of two-dimensional arrays using function:**

* The user-entered matrices are stored in two 2D arrays.
* The arrays are passed as arguments to a function.
* A new 2D array of the same size is created and the elements at same position are added together and stored in the newly created array.
* The contents of the new array are outputted to the console.

Source Code:

#include <iostream>

using namespace std;

int\*\* add\_two\_array(int \*\*arr,int \*\*arr\_2,int length,int width);

int main()

{

    cout << "Enter the dimensions of the matrices" << endl;

    int length,width;

    cin>>length>>width;

    int \*\*first=new int\*[length];*//initializing array*

    int \*\*second=new int\*[length];

    for(int k=0;k<length;k++)

    {

        first[k]=new int[width];

        second[k]=new int[width];

    }

    cout<<"Enter elements of the first array\n";

    for(int i=0;i<length;i++)

        for(int j=0;j<width;j++)

            cin>>first[i][j];

    cout<<"\nEnter elements of the second array\n";

    for(int i=0;i<length;i++)

        for(int j=0;j<width;j++)

            cin>>second[i][j];

    add\_two\_array(first,second,length,width);

    for(int k=0;k<length;k++)*//deallocating memory*

    {

        delete [] first[k];

        delete [] second[k];

    }

    delete [] first;

    delete [] second;

    return 0;

}

int\*\* add\_two\_array(int \*\*arr,int \*\*arr\_2,int length,int width)

{

    cout<<"Array after addition is :\n\n";

    int \*\*temp=new int\*[length];*//allocating memory*

    for(int k=0;k<length;k++)

    {

        temp[k]=new int[width];

    }

    for(int i=0;i<length;i++)*//adding each element*

    {

        for(int j=0;j<width;j++)

        {

            temp[i][j]=arr[i][j]+arr\_2[i][j];

            cout<<temp[i][j]<<" ";

        }

        cout<<endl;

    }

    return temp;

}

Output:

A screenshot of a computer

Description automatically generated with medium confidence

**18.WAP to insert an element at Kth position in a list of n elements:**

* This program provides functionality to insert element at any position within a list.
* First the list is copied to a temporary array till the position where the new element is to be inserted.
* Then new element is added to the temporary array and the rest of the list is attached to temporary array and the list and temporary array are swapped.

Source Code:

#include <iostream>

using namespace std;

void insert\_element(int\*,int,int,int);

int main()

{

    cout <<"Enter length of list: ";

    int length=0;

    cin>>length;

    int\* lis=new int[length];

    cout<<"Enter elements of the list: ";

    for(int i=0;i<length;i++)

    {

        cin>>lis[i];

    }

    cout<<"Enter number to be inserted: ";

    int number=0;

    cin>>number;

    cout<<"Enter position to be inserted at: ";

    int pos=0;

    for(;;)

    {

        cin>>pos;

        if(pos<=length)

            break;

        else

            cout<<"Invalid position"<<endl;

    }

    insert\_element(lis,length,number,pos);

    cout<<"List after insertion: ";

    for(int i=0;i<length+1;i++)

    {

        cout<<lis[i]<<" ";

    }

    delete [] lis;

    return 0;

}

void insert\_element(int\* source,int src\_len,int element,int position)*//{1,2,3,4},4,0,3*

{

    int\* temp=new int[src\_len+1];

    for(int i=0;i<position-1;i++)*//add elements in temp till the position to be inserted*

    {

        temp[i]=source[i];

    } *//after completion of loop temp={1,2}*

*//  add the element to be inserted*

    temp[position-1]=element;*//temp={1,2,0}*

    for(int j=position-1,k=position;j<src\_len;j++,k++)*//add the rest of the elements*

    {

        temp[k]=source[j];

    } *//after completion of loop={1,2,0,3,4}*

    delete [] source;*//deleting source's memeory*

    source=new int[src\_len+1];

    for(int i=0;i<src\_len+1;i++)*//copying elements in temp to source*

        source[i]=temp[i];

    delete [] temp;*//deleting temp's memory*

}

Output:

Text

Description automatically generated

**19.WAP to implement the concept of nesting of classes:**

* In this program the behaviour of nested classes is shown.
* The outer class has a member of type ‘int’ while the inner class has member of type ‘char’.
* Using a method of the inner class the member of outer class is type casted into type ‘char’ and stored into member of inner class and displayed.

Source Code:

#include <iostream>

using namespace std;

class outer

{

  public:

    int number;

    class inner

    {

        public:

        char c;

        void nested\_function(outer o)

    {

    c=o.number;

    }

    };

};

int main()

{

    cout<<"Enter ASCII value\n";

    outer out;

    cin>>out.number;

    outer::inner in;

    in.nested\_function(out);

    cout<<"Character is : "<<in.c;

}

Output:

Graphical user interface, text

Description automatically generated

**20.WAP to illustrate concept of static data members and functions:**

* In this program the concept of static data members is shown.
* The class ‘cls’ has a static data member named ‘count’ which has its value incremented every time an object of class ‘cls’ is instantiated and decremented every time an object is destroyed.
* By using parenthesis to limit scope of variables we can show number of instantiated objects.

Source Code:

#include<iostream>

using namespace std;

class cls

{

    static int count;

public:

    static void put\_count();

    cls();

    ~cls();

};

int cls::count;

cls::cls()

{

    count++;

}

cls::~cls()

{

    count--;

}

void cls::put\_count()

{

    std::cout<<"Number of instantiated objects of cls is : "<<count<<std::endl;

}

int main()

{

    cls first;

    first.put\_count();

    cls second;

    second.put\_count();

    {

        cls third;

        third.put\_count();

        {

            cls fourth;

            fourth.put\_count();

        }

        third.put\_count();

    }

    first.put\_count();

}

Output:

Graphical user interface, text, application, chat or text message

Description automatically generated

**21.WAP to find average value using friend function:**

* The program has two classes ‘INTEGER’ and ‘FLOAT’ which has an integer variable and a float variable as members respectively.
* Using the average function which is friendly to both INTEGER and FLOAT the average of the two numbers is calculated.
* The user enters an integer and a floating-point number.
* Object of the type ‘INTEGER’ and ‘FLOAT’ are created and using them the average of the input numbers is found and outputted to the console.

Source Code:

#include<iostream>

using namespace std;

class INTEGER;

class FLOAT

{

    public:

    float x;

    friend float average(INTEGER, FLOAT);

    FLOAT(float x1)

    {

        x=x1;

    }

};

class INTEGER

{

    public:

    float y;

    friend float average (INTEGER,FLOAT);

    INTEGER(int y1)

    {

        y=y1;

    }

};

float average (INTEGER a, FLOAT b)

{

    float avg= (float)(a.y+b.x)/2;

    return avg;

}

int main()

{

    int i;

    float f;

    cout<<"Enter an Integer: ";

    cin>>i;

    cout<<"Enter a Float: ";

    cin>>f;

    INTEGER r(i);

    FLOAT q(f);

    cout << "Average of the two is :"<<average(r,q);

}

Output:

Text

Description automatically generated

**22.WAP to demonstrate use of default constructor and destructor:**

* This program shows the behaviour of default constructor and destructor.
* We have a class ‘default\_behaviour’ that contains a default constructor and destructor. It also has an integer as its member.
* When we initiate an object of that class the constructor sets the integer member value to zero.
* When the default constructor is called it outputs “Default Constructor Called” and similarly for default destructor.

Source Code:

#include <iostream>

using namespace std;

class default\_behaviour

{

    public:

        int number;

        default\_behaviour(){

            number=0;

            cout<<"Default Constructor Called\n";

        }

        ~default\_behaviour()

        {

            cout<<"Default Destructor Called\n";

        }

};

int main()

{

    {

    default\_behaviour d;

    cout<<"Value of number in default\_behaviour class has been initialized to 0 by default constructor\n";

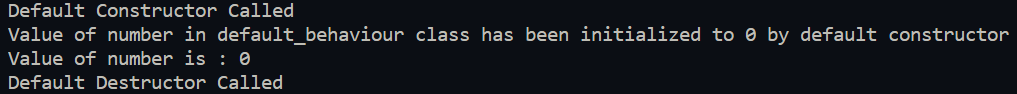
    cout<<"Value of number is : "<<d.number<<"\n";

    }

    return 0;

}

Output:



**23.WAP to demonstrate use of parameterized constructor:**

* We have class “Integer” which has three integers as its members and three parameterized constructors.
* In the first constructor only one parameter is passed, and it is assigned to the first member.
* In the second constructor two parameters are passed, and they are assigned to the first and second members respectively.
* In the third constructor three parameters are passed, and they are assigned to all the members.
* In ‘main’ we ask the user to enter three numbers.
* We then create three objects and pass one, two and three parameters to their constructors.
* The contents of these objects are then displayed.
* As only one member was assigned value in first object the other two members have garbage values.
* Similarly the second object has its last member having garbage value.

#include <iostream>

using namespace std;

class Integer

{

    int i,k,l;

    public:

    Integer(int x):i{x}{};

    Integer(int x,int y):i{x},k{y}{};

    Integer(int x,int y,int z):i{x},k{y},l{z}{};

    void display()

    {

        cout<<i<<' '<<k<<' '<<l<<' ';

    }

};

int main()

{

    cout<<"Please enter three numbers: ";

    int num1,num2,num3;

    cin>>num1>>num2>>num3;

    Integer a(num1);

    Integer b(num1,num2);

    Integer c(num1,num2,num3);

    cout<<"Value of first object is: ";

    a.display();

    cout<<'\n';

    cout<<"Value of second object is: ";

    b.display();

    cout<<'\n';

    cout<<"Value of third object is: ";

    c.display();

    cout<<'\n';

}

Output:

Text

Description automatically generated

**24.WAP to demonstrate use of copy constructor and also show order of constructor and destructor invocation:**

* This program shows usage of copy constructor.
* An Integer class is defined which has an 'int' as its member.
* It also has a copy constructor that copies values from one object to another.
* The user is asked to enter a number this is first used to initialize an object using parameterized constructor and then another object is initialized using copy constructor and they have their contents displayed.

Source Code:

#include <iostream>

using namespace std;

class Integer

{

    int i;

    static int count;

    public:

    Integer(){

        count++;

        cout<<"\nDefault constructor called for: "<<count<<endl;

        }

    ~Integer(){

        cout<<"\nDefault destructor called for: "<<count;

        count--;

        }

    Integer(const Integer &I){

*this*->i=I.get\_data();

        count++;

        cout<<"\nCopy constructor called for : "<<count<<endl;

    }

    void set\_data(int x){i=x;};

    int get\_data() const {return i;};

    static int get\_count()

    {

        return count;

    }

};

int Integer::count;

int main()

{

    int num;

    cout<<"Enter a number : ";

    cin>>num;

    Integer a;

    a.set\_data(num);

    cout<<"Value of Object Initialized by Default Constructor is : "<<a.get\_data()<<endl;

    Integer b{a};

    cout<<"Value of Object Intialized by Copy Constructor is : "<<b.get\_data()<<endl;

}

Output:

Text

Description automatically generated

**25.WAP to implement usage of function overloading:**

* We have an overloaded “add\_numbers” function. The three overloaded functions have two, three and four parameters respectively.
* These functions return the sum of their arguments.
* We ask the user to enter four numbers and then using the overloaded functions we display the sum of the first two , first three and sum of all the numbers.

Source Code:

#include <iostream>

using namespace std;

int add\_numbers(int a,int b)

{

    cout<<"Function with two parameters called\n";

    cout<<"Sum of 1st and 2nd : "<<(a+b)<<"\n";

    return a+b;

}

int add\_numbers(int a,int b,int c)

{

    cout<<"Function with three parameters called\n";

    cout<<"Sum of 1st ,2nd and 3rd : "<<(a+b+c)<<"\n";

    return a+b+c;

}

int add\_numbers(int a,int b,int c,int d)

{

    cout<<"Function with four parameters called\n";

    cout<<"Sum of all numbers : "<<(a+b+c+d)<<endl;

    return a+b+c+d;

}

int main()

{

    int a,b,c,d;

    cout<<"Enter four numbers: ";

    cin>>a>>b>>c>>d;

    cout<<"\n";

    add\_numbers(a,b);

    cout<<"\n";

    add\_numbers(a,b,c);

    cout<<"\n";

    add\_numbers(a,b,c,d);

    cout<<'\n';

}

Output:

Text

Description automatically generated

**26.WAP to illustrate the concept of unary operator overloading:**

* We have an “Integer” class that has an integer variable as a member.
* We overload the unary operator which multiplies the member with -1 if the number is positive.
* We ask the user to input a negative number and pass it to an “Integer” object.
* We use the unary operator on the object and then display contents of the object.

Source Code:

#include <iostream>

using namespace std;

class Integer

{

    int i;

    public:

    Integer(int x=0){i=x;};

    int operator+(){

        if(*this*->i<0)

            i=i\*(-1);

        return i;

    };

    int get\_data(){return i;}

};

int main()

{

    cout<<"Please Enter a negative number: ";

    int num;

    for(;;)

    {

        cin>>num;

        if(num>=0)

        {

            cout<<"Please Enter a Negative number: ";

        }

        else

            break;

    }

    Integer a(num);

    cout<<"Value of a is "<<a.get\_data()<<"\n";

    +a;

    cout<<"Value of a after using unary + operator : "<<a.get\_data()<<"\n";

}

Output:

Graphical user interface, text

Description automatically generated

**27.WAP to calculate sum of two complex numbers by using operator overloading:**

* A ‘Complex’ class is created which has two integer members ‘real’ and ‘imaginary’.
* The binary ‘+’ operator is overloaded for this class.
* In ‘main’ function the user is asked to enter two complex numbers and they are used to construct two objects.
* Then using the binary ‘+’ operator the sum of the complex numbers is taken and outputted.

Source Code:

#include <iostream>

using namespace std;

class Complex

{

    int real;

    int imaginary;

    public:

    Complex(int a=0,int b=0):real(a),imaginary(b){};

    int get\_real() const {return *this*->real;}

    int get\_imaginary() const{return *this*->imaginary;}

    void display()

    {

        cout<<real<<" + "<<imaginary<<"i\n";

    }

    Complex& operator+(const Complex &x)

    {

*this*->real+=x.get\_real();

*this*->imaginary+=x.get\_imaginary();

        return \**this*;

    }

};

int main()

{

    int a,b;

    cout<<"Enter 1st Complex Number: ";

    cin>>a>>b;

    Complex first(a,b);

    cout<<"Enter 2nd Complex Number: ";

    cin>>a>>b;

    Complex second(a,b);

    first=first+second;

    cout<<"Sum of the two complex numbers is : ";

    first.display();

    cout<<"\n";

}

Output:

Text

Description automatically generated

**28.Write a C++ that overloads the + operator and relational operators to perform the following operations:**

1. **Concatenation of two strings**
2. **Comparison of two strings**

* A ‘String’ class is created with the appropriate constructor which takes a pointer to ‘char’ as argument and copies it to the member in the class using ‘strcpy’.
* The binary ‘+’ operator is overloaded for this class. The strings are concatenated using the ‘strcat’ function.
* Similarly, the ‘>’, ‘<’, and ‘==’ operator is overloaded and the strings are compared using ‘strcmp’ function.
* The user is asked to input two strings. These strings are then used to create two ‘String’ objects.
* Then the concatenated string is displayed and using the relational operators comparison of the strings is outputted.

Source Code:

#include <iostream>

#include <cstring>

#include <iomanip>

#include <string>

using namespace std;

class String

{

    char\* str;

public:

    String(char\* temp)

    {

        if(temp==nullptr)

        {

            str=new char[1];

            \*str='\0';

        }

        else

        {

            str=new char[strlen(temp)+1];

            strcpy(str,temp);

        }

    }

    char\* get\_string() const {return str;};

    String& operator+(const String& S)

    {

        strcat(*this*->str,S.get\_string());

        return \**this*;

    }

    bool operator<(const String& S)

    {

        if(strcmp(*this*->str,S.get\_string())<0)

        {

            return true;

        }

        else

            return false;

    }

    bool operator>(const String& S)

    {

         if(strcmp(*this*->str,S.get\_string())>0)

        {

            return true;

        }

        else

            return false;

    }

    bool operator==(const String& S)

    {

        if(strcmp(*this*->str,S.get\_string())==0)

            return true;

        else

            return false;

    }

    friend ostream& operator<<(ostream &os,String str);

};

ostream& operator<<(ostream &os,const String str)

{

    os<<str.get\_string();

    return os;

}

int main()

{

    char\* str\_1=new char[70];

    char\* str\_2=new char[70];

    cout<<"Enter first string"<<endl;

    cin.getline(str\_1,69);

    cout<<"Enter the second string"<<endl;

    cin.getline(str\_2,69);

    String first(str\_1);

    String second(str\_2);

    delete [] str\_1;

    delete [] str\_2;

    cout<<"Concatenating the strings :"<<(first+second)<<endl;

    cout<<boolalpha;

    cout<<"Is first string greater than second string :"<<(first>second)<<endl;

    cout<<"Is first string smaller than second string :"<<(first<second)<<endl;

    cout<<"Is first string equal to second string :"<<(first==second)<<endl;

}

Output:

Text

Description automatically generated

**29.WAP to determine if given string is palindrome or not:**

* The user is asked to enter a string.
* Then using the ‘is\_palindrome” function it is determined if string is palindrome or not.
* In the ‘is\_palindrome’ function a copy of the input string is made and the copied string is reversed.
* Then using the overloaded ‘==’ operator ,if strings are equal then the string is palindrome else it is not palindrome.

Source Code:

#include <iostream>

#include <string>

using namespace std;

bool is\_palindrome(string str)

{

    string temp;

    for(int i=(int(str.size())-1);i>=0;i--)

    {

        temp.push\_back(str.at(i));

    }

    if(temp==str)

        return 1;

    else

        return 0;

}

int main()

{

    string str;

    cout<<"Enter String:";

    getline(cin,str);

    cout<<boolalpha;

    cout<<"String is palindrome :"<<is\_palindrome(str)<<endl;

}

Output:



**30.WAP to use function templates to solve problems:**

* In this program we have created a ‘MyVector’ class. It contains a pointer to a generic type and an integer variable position.
* The constructor takes no parameters and allocates one block of memory.
* In the ‘push\_back’ method a new pointer to generic type is created and is allocated memory one block more than the memory allocated by the pointer in the class. The contents of the pointer in the class are copied to the temporary pointer and the new element is inserted at the end. The memory for the pointer in the class is deallocated and then reallocated and contents from the temporary pounter is copied to the pointer in class.
* Similarly, In ‘pop\_back’ method the memory only till the second last block is copied and rest remains same.
* In output we add and remove variables from the ‘MyVector’ objects and display them.

Source Code:

#include <iostream>

using namespace std;

template <class T>

class MyVector

{

    T\* arr;

    int position;

    public:

    MyVector(void)

    {

        arr=new T[1];

        position=0;

    };

    ~MyVector()

    {

        delete [] arr;

    }

    void push\_back(T input)

    {

        T \*temp=new T[position+1];

        for(int i=0;i<position;i++)

        {

            temp[i]=arr[i];

        }

        temp[position]=input;

        position++;

        delete [] arr;

        arr=new T[position];

        for(int i=0;i<position;i++)

        {

            arr[i]=temp[i];

        }

        delete [] temp;

    }

    void pop\_back()

    {

        T \*temp=new T[position-1];

        for(int i=0;i<(position-1);i++)

        {

            temp[i]=arr[i];

        }

        delete [] arr;

        position--;

        arr=new T[position];

        for(int i=0;i<position;i++)

        {

            arr[i]=temp[i];

        }

    }

    T at(int x)

    {

        try

        {

            if(x<0)

            {

                throw -1;

            }

            else if(x>=position)

            {

                throw '1';

            }

            else

            {

                return arr[x];

            }

        }

        catch(const int x)

        {

            std::cout<<"Out of bounds(underflow)" << '\n';

            throw -1;

        }

        catch(const char c)

        {

            std::cout<<"Out of bounds(overflow)"<<'\n';

            throw '1';

        }

        return 0;

    }

};

int main()

{

    MyVector<int> vec;

    cout<<"Pushing 10 and 12 to the first vector: \n";

    vec.push\_back(10);

    vec.push\_back(12);

    cout<<vec.at(0)<<" "<<vec.at(1)<<endl;

    cout<<"Removing 12 and adding 15 to the first vector: \n";

    vec.pop\_back();

    vec.push\_back(15);

    cout<<vec.at(0)<<" "<<vec.at(1)<<endl;

    cout<<"Pushing a and b to the second vector: \n";

    MyVector<char> vec\_2;

    vec\_2.push\_back('a');

    vec\_2.push\_back('b');

    cout<<vec\_2.at(0)<<" "<<vec\_2.at(1)<<endl;

    cout<<"Removing b and adding v to the first vector: \n";

    vec\_2.pop\_back();

    vec\_2.push\_back('v');

    cout<<vec\_2.at(0)<<" "<<vec\_2.at(1)<<endl;

}

Output:

Text

Description automatically generated

**31.WAP to illustrate how the following forms of inheritance are supported:**

1. **Single Inheritance**
2. **Multiple Inheritance**
3. **Multilevel Inheritance**
4. **Hierarchical Inheritance**

* This program shows various methods of inheritance by making multiple interrelated classes that are used to perform operations on numbers.

Source Code:

#include<iostream>

using namespace std;

class values              *//base class*

{

    int a;

    int b;

    public:

    values(int x,int y):a{x},b{y}{}

    int get\_a(){

        return a;

    }

    int get\_b(){

        return b;

    }

};

class add:public virtual values *//single inheritance*

{

    int c;

    public:

    add(int x,int y):values(x,y){}

    void addition()

    {

        c=get\_a()+get\_b();

    }

    void display\_sum()

    {

        cout<<"Sum of Values: "<<c<<endl;

    }

    int get\_c()

    {

        return c;

    }

};

class subtract:public virtual values,public add *//multiple inheritance*

{

    int d;

    public:

    subtract(int x,int y):values(x,y),add(x,y){}

    void subtraction()

    {

        d=get\_a()-get\_b()-get\_c();

    }

    void display\_diff()

    {

        cout<<"Difference of first value and its sum: "<<d<<endl;

    }

};

class larger: public values

{

    int e;

    public:

    larger (int x,int y):values(x,y)

    {

        if (get\_a()>get\_b())

            e= get\_a();

        else

            e= get\_b();

    }

    void display1()

    {

        cout<<"Larger number is: "<<e<<endl;

    }

    int get\_e()

    {

        return e;

    }

};

class smaller: public values

{

    int f;

    public:

    smaller(int x,int y):values(x,y)

    {

        if (get\_a()<get\_b())

            f=get\_a();

        else

            f=get\_b();

    }

    void display2()

    {

        cout<<"Smaller number is :"<<f<<endl;

    }

};

class multiply:public larger *//multilevel inheritance*

{

    int g;

    public:

    multiply(int x,int y):larger(x,y)

    {

        g=get\_a()\*get\_b();

    }

    void showmult()

    {

        cout<<"the Result of Multiplication of First Value and Second Value is: "<<g;

    }

};

int main()

{

    int x,y;

    cout<<"Enter two numbers: ";

    cin>>x>>y;

    add sum1(x,y);

    sum1.addition();

    sum1.display\_sum();

    subtract diff1(x,y);

    diff1.addition();

    diff1.subtraction();

    diff1.display\_diff();

    larger o1(x,y);

    o1.display1();

    smaller o2(x,y);

    o2.display2();

    multiply o3(x,y);

    o3.showmult();

}

Output:

A screenshot of a computer

Description automatically generated with medium confidence

**32.WAP that illustrates how run time polymorphism is achieved using virtual functions:**

* In this program we have A and B which are two classes and B is derived from A.
* We have a display method in both the classes which is virtual in A.
* Using this we can access correct methods of class B while using pointer to class A.
* In ‘main’ we create an object of type ‘A’ and an object of type ‘B’ and call their display function normally and then using a pointer of type ‘A’.

Source Code:

#include <iostream>

using namespace std;

class A

{

    public:

    virtual void display()

    {

        cout<<"This is A\n";

    }

};

class B: public A

{

    public:

    void display()

    {

        cout<<"This is B\n";

    }

};

int main()

{

    A a;

    cout<<"Accessing display method of A using dot operator\n";

    a.display();

    B b;

    cout<<"Accessing display method of B using dot operator\n";

    b.display();

    A \*ptrToObject=new B;

    cout<<"Accessing display method of B using a pointer of type A\n";

    ptrToObject->display();

}

Output:

Text

Description automatically generated

**33.WAP to illustrate the use of pure virtual function:**

* In this program a pure virtual function is used to make an abstract menu class.
* This class is used to make a main menu class in which we define the virtual display method and use to display the main menu.

Source Code:

#include <iostream>

#include <iomanip>

using namespace std;

*//Abstract Class for menus*

class Menu

{

    public:

    virtual void display()=0;

};

class MainMenu : public Menu

{

    public:

    void display()

    {

        cout<<setw(20)<<setfill('\*')<<'\*'<<endl;

        cout<<"You are in Main Menu\n";

        cout<<setw(20)<<'\*'<<endl;

    }

};

int main()

{

    MainMenu menu;

    cout<<"\n\n\n";

    menu.display();

}

Output:

Text

Description automatically generated

**34.Write a C++ program that illustrates use of virtual base class in building class hierarchy:**

* In this program we show the importance of virtual base class by having a hierarchical inheritance and multiple inheritance.
* By using virtual base class, we ensure only one copy of the grandparent class is inherited.
* The class ‘A’ is inherited by both class ‘B’ and class ‘C’. Then class ‘D’ is created by inheriting classes ‘B’ and ‘C’. Because we use virtual base class only one copy of ‘A’ is inherited.

Source Code:

#include <iostream>

using namespace std;

class A

{

    protected:

    int a;

    public:

    A(int x):a{x}{};

    virtual void display(){cout<<a<<endl;}

};

class B : virtual public A

{

    protected:

    int b;

    public:

    B(int x):A(x\*2),b{x}{};

    void display(){cout<<b<<' '<<a<<endl;}

};

class C : virtual public A

{

    protected:

    int c;

    public:

    C(int x):A(x\*5),c{x}{};

    void display(){cout<<c<<' '<<a<<endl;}

};

class D : public C,public B

{

    protected:

    int d;

    public:

    D(int x):A(1000\*x),C(x\*100),B(x\*10),d{x}{};

    void display(){cout<<d<<' '<<c<<' '<<b<<' '<<a<<endl;}

};

int main()

{

    A a(1);

    B b(2);

    C c(3);

    D d(4);

    a.display();

    b.display();

    c.display();

    d.display();

}

Output:

A picture containing text, device, meter

Description automatically generated

**35. Write a C++ program that illustrates use of abstract class in building class hierarchy:**

* In this program a pure virtual function is used to make an abstract menu class. This class is used to make a main menu class in which we define the virtual display method and use to display the main menu.

Source Code:

#include <iostream>

#include <iomanip>

using namespace std;

*//Abstract Class for menus*

class Menu

{

    public:

    virtual void display()=0;

};

class MainMenu : public Menu

{

    public:

    void display()

    {

        cout<<setw(20)<<setfill('\*')<<'\*'<<endl;

        cout<<"You are in Main Menu\n";

        cout<<setw(20)<<'\*'<<endl;

    }

};

int main()

{

    MainMenu menu;

    cout<<"\n\n\n";

    menu.display();

}

Output:

Text

Description automatically generated

**36.Write a C++ program to display contents of a text file:**

* An ‘ifstream’ object is created and the input file is opened.
* Then using a ‘for’ loop and ‘getline’ the contents of the file are displayed to the console.

Source Code:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

    ifstream input("36\_file.txt");

    if(!input.is\_open())

    {

        cout<<"File could not be opened\n";

    }

    else

    {

        string str;

        for(;input;)

        {

            if(getline(input,str))

                cout<<str<<endl;

        }

    }

}

Output:

Text

Description automatically generated

**37.Write a C++ program which copies one file to another:**

* In this program we use the ‘ofstream’ and ‘ifstream’ class to open a file for writing and reading respectively.
* We then use the ‘getline’ function to extract lines from the input text file and the stream insertion operator to insert these lines into the output text files.

Source Code:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

    ifstream src("37\_source.txt");

    ofstream dest("37\_destination.txt");

    try

    {

        if (!src.is\_open())

        {

            throw -1;

        }

        if (!dest.is\_open())

        {

            throw '1';

        }

        cout << "Starting Copy operation\n";

        string temp\_str;

        while (getline(src, temp\_str))

        {

            dest << temp\_str<<endl;

        }

        cout << "Copy operation finished\n";

    }

    catch (const int x)

    {

        cout << "Source file could not be opened\n";

    }

    catch (const char c)

    {

        cout << "Destination file could not be opened\n";

    }

}

Output:



**38.Write a C++ program that counts number of characters, lines and words in a text file:**

* This program counts the number of lines, words, and characters in a text file.
* This is done by reading the text file line by line using ‘getline’ function and then examining the line for words and characters.
* After going through all the lines, the final count is displayed.

Source Code:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

    int character=0,line=0,words=0;

    ifstream input\_file("38\_file.txt");

    try

    {

        if(!input\_file.is\_open())

        {

            throw -1;

        }

        else

        {

            string temp\_string;

            while(getline(input\_file,temp\_string))

            {

                line++;

                for(unsigned int i=0;i<temp\_string.size();i++)

                {

                    if(temp\_string.at(i)==' ' || (i==(temp\_string.size()-1)))

                    {

                        words++;

                    }

                    if(temp\_string.at(i)!=' ')

                    {

                        character++;

                    }

                }

            }

            cout<<"Number of Lines are : "<<line<<"\nNumber of words are : "<<words<<"\nNumber of characters are: "<<character<<endl;

        }

    }

    catch(const int x)

    {

        std::cout << "Could not open file" << '\n';

    }

}

Output:

Graphical user interface, text

Description automatically generated