Logo, company name

Description automatically generated

**2nd  year**

Subject - object-oriented programming (Lab)

Department of computer science

and

information technology

Subject Code – CSIT305

Branch & Section  – CSIT-3

Enrollment no. – 0827CI201151

Submitted to – prof.  Garima KumrawaT

Submitted by – Raj borasi

**List of Experiments:-**

1. Array.
2. Functions.
3. Classes.
4. Number and string.
5. Constructors and Destructor.

**Array**

**EXPERIMENT -1**

**AIM:**

**Write a program to enter 5 numbers and display first and last one only.**

**DESCRIPTION:**

**An array is a data structure that contain a group element .Typically these elements are the same data type , such as an integer , string ,char .Array are commonly used in a computer program to organize data so that a related of values can easily sorted or searched.**

**Source code:**

#include <iostream>

using namespace std;

int main()

{

    int a[5];

    for (int i = 0; i < 5; i++)

    {

        cin >> a[i];

    }

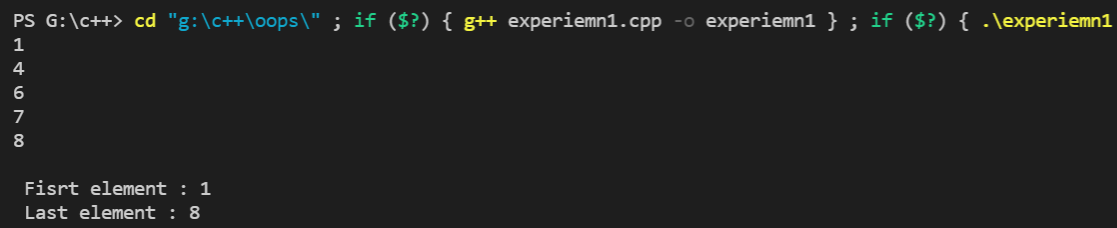
    cout << "\n Fisrt element : " << a[0];

    cout << "\n Last element : " << a[4];

    return 0 ;

}

**OUTPUT:**

****

**EXPERIMENT -2**

**AIM : Write a program to insert the element at specific position in array**

**DESCRIPTION :**

**An array is a data structure that contain a group element .Typically these elements are the same data type , such as an integer , string ,char .Array are commonly used in a computer program to organize data so that a related of values can easily sorted or searched.**

**SOURCE CODE :**

#include <iostream>

using namespace std;

int main()

{

    int size , pos, element;

    char y;

    cout << "\nEnter the size of array ";

    cin >> size;

    int array[size];

    do

    {

        cout << "Enter the element which you want to enter and also enter it's position" << endl;

        cin >> element >> pos;

        array[pos] = element;

        cout << "]\n Array at positon " << pos << " is " << array[pos];

        cout << "\n do you wannt to enter more element ";

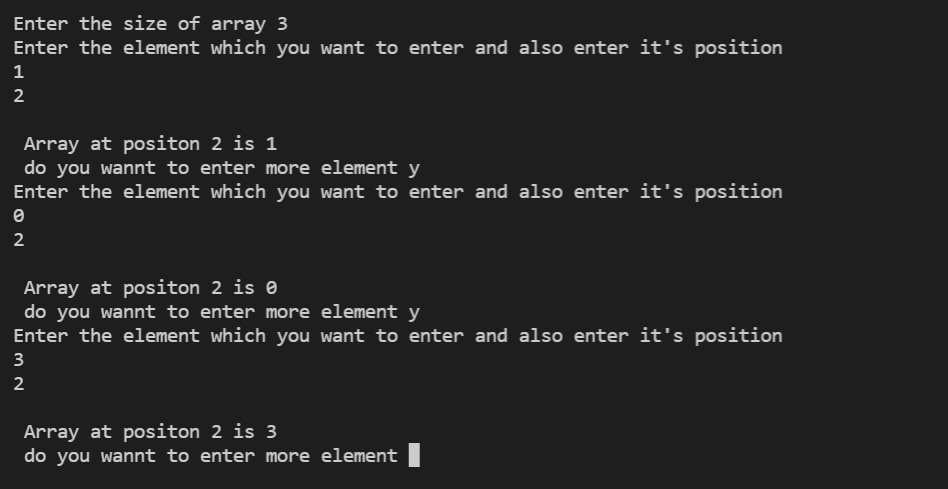
        cin >> y;

    } while (y == 'y');

    return 0;

}

**OUTPUT :**

****

**EXPERIMENT -3**

**AIM :write a program to revrese a array**

**Description :**

**An array is a data structure that contain a group element .Typically these elements are the same data type , such as an integer , string ,char .Array are commonly used in a computer program to organize data so that a related of values can easily sorted or searched.**

**SOURCE CODE :**

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

    int size;

    cout << "\n Enter the size of array ";

    cin >> size;

    int array[size];

    cout << "\n Enter the element of array ";

    for (int i = 0; i < size; i++)

    {

        cin >> array[i];

    }

    cout << "\n Your initial array is :";

    for (int i = 0; i < size; i++)

    {

        cout << array[i] << " ";

    }

    cout << endl;

    cout << "\n Reverse of you array is :";

    reverse(array, array + size);

    for (int i = 0; i < size; i++)

    {

        cout << array[i] << " ";

    }

    return 0;

}

**Output :**

****

**EXPERIMENT -4**

**AIM: Write the program for addition of two matrix .**

**DESCRIPTION :**

**An array is a data structure that contain a group element .Typically these elements are the same data type , such as an integer , string ,char .Array are commonly used in a computer program to organize data so that a related of values can easily sorted or searched.**

**SOURCE CODE :**

#include <iostream>

using namespace std;

int main()

{

    int row, col;

    cout << "Enter the number of rows" << endl;

    cin >> row;

    cout << "Enter the number of cols" << endl;

    cin >> col;

    int a[row][col];

    int b[row][col];

    int add[row][col];

    cout << "Enter the elements  ";

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cin >> a[i][j];

        }

    }

    cout << "your first matrix is :-" << endl;

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cout << a[i][j] << "\t";

        }

        cout << endl;

    }

     cout << "Enter the elements  of second ";

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cin >> b[i][j];

        }

    }

     cout << "your second matrix is :-" << endl;

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cout << b[i][j] << "\t";

        }

        cout << endl;

    }

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            add[i][j] = a[i][j] + b[i][j];

        }

    }

 cout << " Addition of matrix :"<<endl;

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cout << add[i][j] << "\t";

        }

        cout << endl;

    }

    return 0;

}

**Output :**

****

**EXPERIMENT -5**

**AIM : Write a program to multiply two matrix using multi dimensional array.**

**DESCRIPTION :**

**An array is a data structure that contain a group element .Typically these elements are the same data type , such as an integer , string ,char .Array are commonly used in a computer program to organize data so that a related of values can easily sorted or searched.**

**SOURCE CODE:**

#include <iostream>

using namespace std;

int main()

{

    int row, col;

    cout << "Enter the number of rows" << endl;

    cin >> row;

    cout << "Enter the number of cols" << endl;

    cin >> col;

    int a[row][col];

    int b[row][col];

    int mul[row][col];

    cout << "Enter the elements  ";

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cin >> a[i][j];

        }

    }

    cout << "your first matrix is :-" << endl;

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cout << a[i][j] << "\t";

        }

        cout << endl;

    }

    cout << "Enter the elements  of second ";

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cin >> b[i][j];

        }

    }

    cout << "your second matrix is :-" << endl;

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            cout << b[i][j] << "\t";

        }

        cout << endl;

    }

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

            mul[i][j] = 0;

        for (int k = 0; k < col; k++)

        {

            mul[i][k] += a[i][k] \* b[k][j];

        }

        }

}

cout << " multiple of matrix :" << endl;

for (int i = 0; i < row; i++)

{

    for (int j = 0; j < col; j++)

    {

        cout << mul[i][j] << "\t";

    }

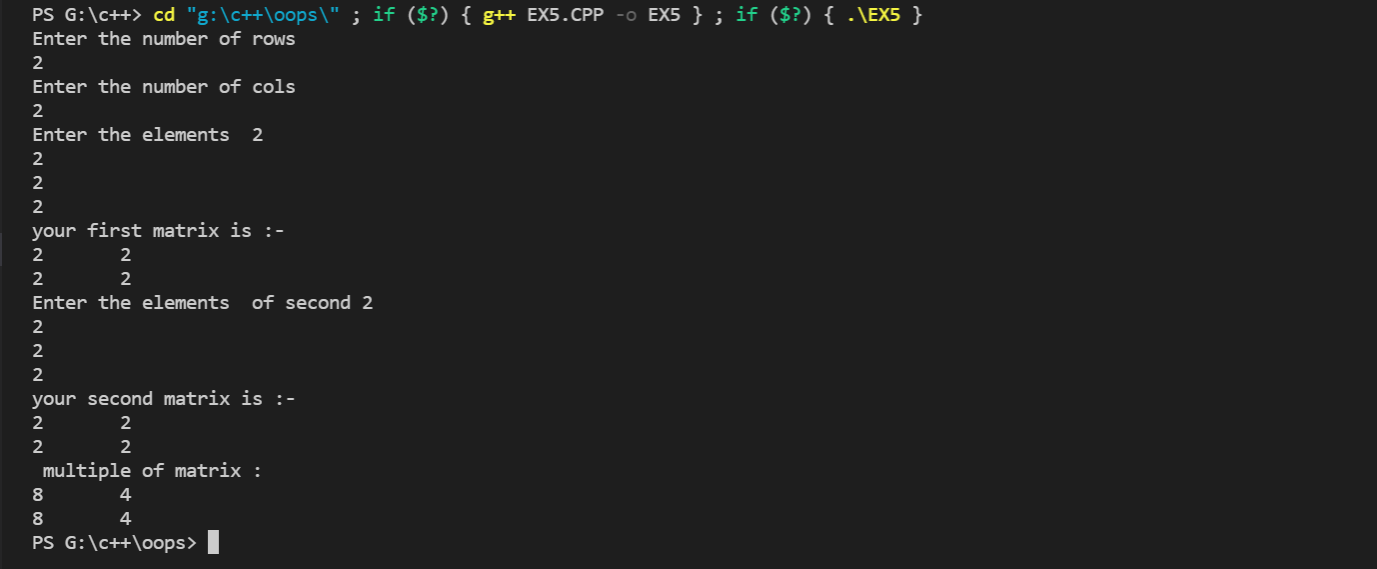
    cout << endl;

}

return 0;

}

**OUTPUT:**

****

***FUNCTIONS***

**EXPERIMENT -1**

**AIM:** Write a program to find largest among

Three number using function.

**DESCRITION:**

A function is a block a of code which only runs when it is called .

You can pass data , know as parameters,into a function.

Function are used to perform certain action and they are important for resuing code; define the code once and use it mant times.

**SOURCE CODE :**

#include <iostream>

using namespace std;

int greatest(int x, int y, int z)

{

    if (x > y && x > z)

    {

        return x;

    }

    else if (y > x && y > x)

    {

        return y;

    }

    else

        return z;

}

int main()

{

    int a, b, c;

    cout << " Enter three number to find greates "<<endl;

    cin >> a >> b >> c;

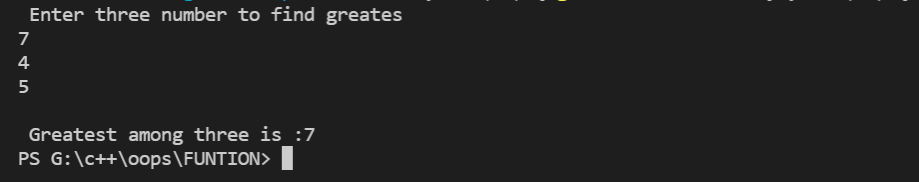
    int g = greatest(a, b, c);

    cout << "\n Greatest among three is :" << g;

    return 0;

}

**OUTPUT:**

****

**EXPERIMENT -2**

**AIM:** Write a program to find area of triangle ,circle, and rectangle using function overloading.

**DESCRIPTION :** A function is a block a of code which only runs when it is called .

You can pass data , know as parameters,into a function.

Function are used to perform certain action and they are important for resuing code; define the code once and use it mant times.

**SOURCE CODE:**

#include<iostream>

using namespace std ;

int area(int r){

    int area  = 3.14\*r\*r;

    return area;

}

int area(int l ,int b ,int h)

{

    int area = l\*b\*h;

    return area;

}

int area(int h ,int b){

    int area = (h\*b)/2;

    return area;

}

int main()

{

    int r,l,b,h,h1,b1;

    cout<<"\n Enter the radious of triangle "<<endl;

    cin>>r;

    cout<<"\n area of circle is : "<<area(r);

    cout<<"\n Enter the length breadth and height of rectangle: "<<endl;

    cin>>l>>b>>h;

    cout<<"\n area of rectangel is : "<<area(l,b,h);

    cout<<"\n Enter the height and base of triangle: "<<endl;

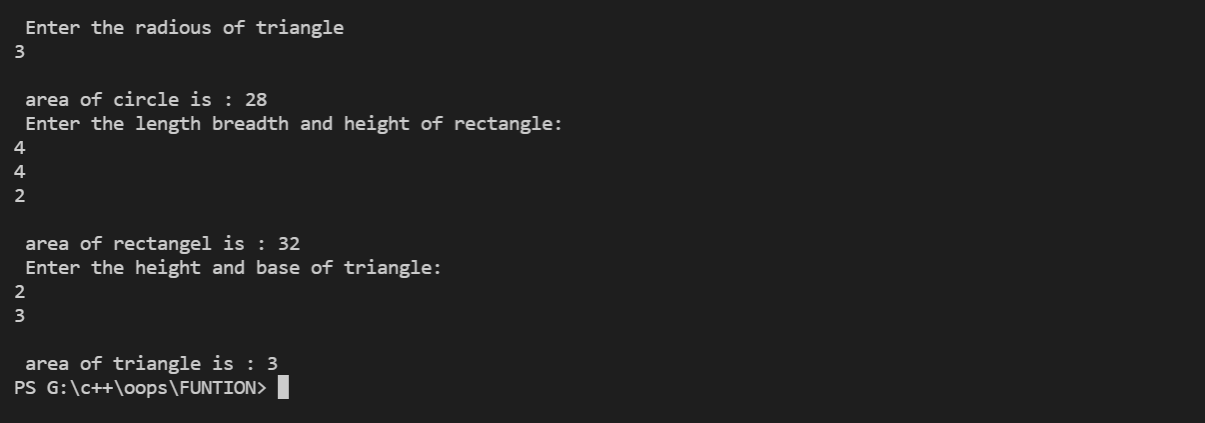
    cin>>h1>>b1;

    cout<<"\n area of triangle is : "<<area(h1,b1);

    return 0 ;

}

**OUTPUT:**

****

**EXPERMINT-3**

**AIM:** Write a program for additition , multiplications , subtraction using function.

**DESCRITION:** A function is a block a of code which only runs when it is called .

You can pass data , know as parameters,into a function.

Function are used to perform certain action and they are important for resuing code; define the code once and use it mant times.

**SOURCE CODE:**

#include <iostream>

using namespace std;

int add(int a, int b)

{

    int add = a + b;

    return add;

}

int sub(int a, int b)

{

    int sub = a - b;

    return sub;

}

int mul(int a, int b)

{

    int mul = a \* b;

    return mul;

}

int main()

{

    int a, b;

    cout<<"Enter two number to perform multipication ,subtraction and addition "<<endl;

    cin>>a>>b;

    cout<<"addition  : "<<add(a,b)<<endl;

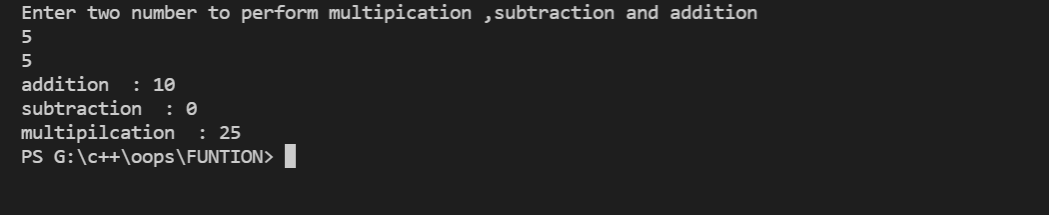
    cout<<"subtraction  : "<<sub(a,b)<<endl;

    cout<<"multipilcation  : "<<mul(a,b)<<endl;

    return 0;

}

**OUTPUT:**

****

**EXPERIMENT -4**

**AIM:** Write a program to swap two values using call by value and call by refrence method.

**DESCRITION: :** A function is a block a of code which only runs when it is called .

You can pass data , know as parameters,into a function.

Function are used to perform certain action and they are important for resuing code; define the code once and use it mant times.

**SOURCE CODE:**

#include<iostream>

using namespace std ;

  void swap1(int &a,int &b){

           int temp =a;

            a =b;

            b=temp; }

int swap2(int x,int y){

      int temp =x;

            x =y;

            y=temp;

     cout<<"\nVAlue of A after swaping by call by value :"<<x;

      cout<<"\nVAlue of B after swaping by call by value :"<<y;

}

int main()

{

      int a,b;

      cout<<"\nEnter value of a :";

      cin>>a;

      cout<<"\nEnter value of b: ";

      cin>>b;

      cout<<"\nVAlue of A before swaping :"<<a;

      cout<<"\nVAlue of B before swaping :"<<b;

      swap2(a,b);

      swap1(a,b);

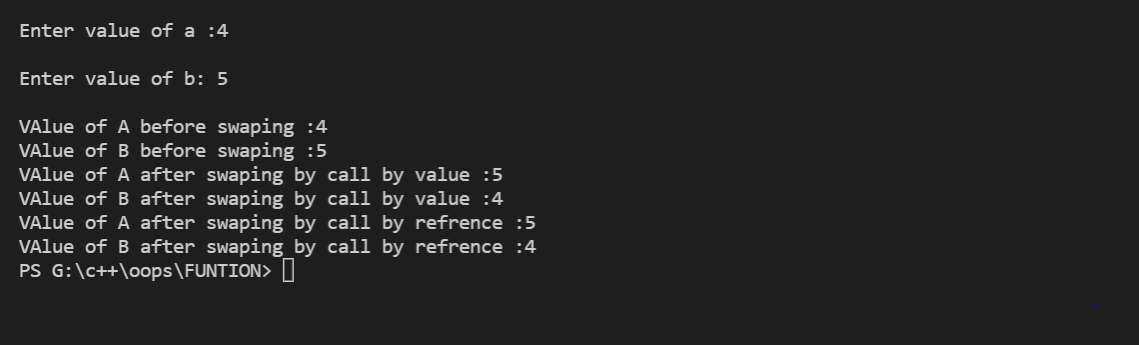
      cout<<"\nVAlue of A after swaping by call by refrence :"<<a;

      cout<<"\nVAlue of B after swaping by call by refrence :"<<b;

    return 0 ;

}

**OUTPUT:**

****

**EXPERIMENT -5**

**AIM:** Write a program for number conversions .

**DESCRITION: :** A function is a block a of code which only runs when it is called .

You can pass data , know as parameters,into a function.

Function are used to perform certain action and they are important for resuing code; define the code once and use it many times.

**SOURCE CODE:**

#include <iostream>

using namespace std;

void decToBinary(int n)

{

    int binaryNum[32];

    int i = 0;

    while (n > 0)

    {

        binaryNum[i] = n % 2;

        n = n / 2;

        i++;

    }

    for (int j = i - 1; j >= 0; j--)

        cout << binaryNum[j];

}

int main()

{

    int n ;

    cout<<"\nEnter the number to convert into decimal ";

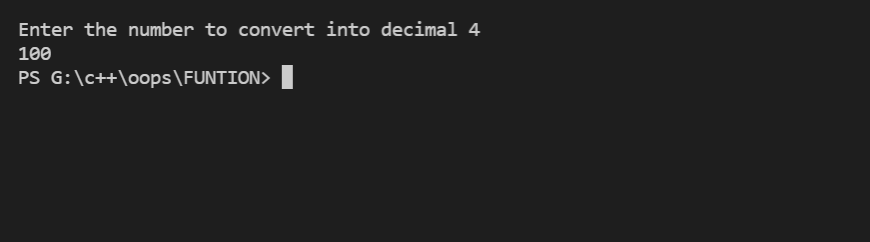
    cin>>n;

    decToBinary(n);

    return 0;

}

**OUTPUT:**

****

**Class and Object**

**EXPERIMENT -1**

**AIM:**

**Find the factorial by defining function outside the class.**

**DESCRIPTION:**

A class is a blueprint for the object.  
  
We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, we need to create objects.

**SOURCE CODE :**

#include <iostream>

using namespace std;

class factorial

{

public:

    int factorail1(int f);

} f;

int factorial ::factorail1(int f)

{

    int fact = 1;

    for (int i = 1; i <= f; i++)

    {

        fact = fact \* i;

    } return fact;

}

int main()

{

    int a;

    cout << "\nEnter the number to find factorail ";

    cin >> a;

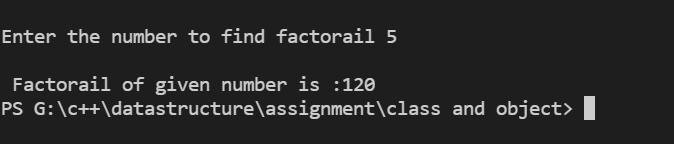
    int fact = f.factorail1(a);

      cout << "\n Factorail of given number is :"<<fact;

    return 0;

}

**OUTPUT:**



**EXPERIMENT -2**

**AIM:**

**C++ program to find greatest between 3 numbers by defining the functions inside class**

**DESCRIPTION:**

A class is a blueprint for the object.  
  
We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, we need to create objects.

**SOURCE CODE :**

// GREATEST AMONG THREE NUMBER BY DDEFINING FUNCTION INSIDE THE CLASS

#include <iostream>

using namespace std;

class greatest

{

public:

    int greatest1(int a, int b, int c)

    {

        if (a > b && a > c)

            return a;

        else if (b > a && b > c)

            return b;

        else

            return c;

    }

} G;

int main()

{

    int a, b, c;

    cout << "\nEnter three number to find grestest : ";

    cin >> a >> b >> c;

   int g = G.greatest1(a, b, c);

    cout<<"\n GREATEST NUMBER IS :"<<g;

    return 0;

}

**OUTPUT :**

**Text

Description automatically generated**

**EXPERIMENT -3**

**AIM:**

**C++ program to find reverse of number by defining functions outside class**

**DESCRIPTION:**

A class is a blueprint for the object.  
  
We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, we need to create objects.

**SOURCE CODE :**

// REVERSE NUMBER BY DEFINING FUNTION OUTSIDE THE CLASS

#include <iostream>

using namespace std;

class reverse

{

public:

    void reverse1(int x[]);

} R;

void reverse :: reverse1(int x[])

    {

        cout << "reverse of your array is :";

        for (int i = 4; i >= 0; i--)

        {

            cout << x[i]<<" ";

        }

    }

int main()

{

    int a[5];

    cout << "Enter element of array : ";

    for (int i = 0; i < 5; i++)

    {

        cin >> a[i];

    }

    R.reverse1(a);

    return 0;

}

**OUTPUT :**

**Graphical user interface, text

Description automatically generated**

**EXPERIMENT -4**

**AIM:**

**C++ program to add two complex number passing objects as arguments**

**DESCRIPTION:**

A class is a blueprint for the object.  
  
We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, we need to create objects.

**SOURCE CODE :**

//C++ program to add two complex number passing objects as arguments

#include<iostream>

using namespace std;

class complex

{

        int re,im;

        public:

                void get()

                {

                    cout<<"\nEnter Real Part :: ";

                        cin>>re;

                        cout<<"\nEnter Imag. Part :: ";

                        cin>>im;

                }

                void disp()

                {

                        cout<<re<<"+"<<im<<"i"<<"\n";

                }

                void sum(complex,complex);

};

void complex::sum(complex c1,complex c2)

{

        re=c1.re+c2.re;

        im=c1.im+c2.im;

}

int main()

{

        complex c1,c2,c3;

        cout<<"Enter 1st complex no.: \n";

        c1.get();

        cout<<"\nEnter 2nd complex no.: \n";

        c2.get();

        cout<<"\nThe 1st complex no. is :: ";

        c1.disp();

        cout<<"\nThe 2nd complex no. is :: ";

        c2.disp();

        c3.sum(c1,c2);

        cout<<"\nThe Sum of two complex no.s are :: ";

        c3.disp();

        return 0;

}

**OUTPUT :**

**Text

Description automatically generated**

**EXPERIMENT - 5**

**AIM:**

**C++ program to add two complex number passing objects as arguments**

**DESCRIPTION:**

A class is a blueprint for the object.  
  
We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, we need to create objects.

**SOURCE CODE :**

//

#include <iostream>

using namespace std;

class sum

{

    int x = 100;

public:

    int sum1(int a)

    {

        int x = a;

        int add = 0;

        for (int i = 1; i < 100; i++)

        {

            if (i % 2 != 0)

            {

                add = add + i;

            }

        } cout<<"\nSUM of first 100 odd natural number is :"<<add;

    }

} S;

int main()

{

    int x = 100;

    S.sum1(x);

    return 0;

}

**OUTPUT :**

**OUTPUT :**

**Text

Description automatically generated**

**NUMBER AND STRING**

**EXPERIMENT -1**

**AIM :**

**C++ program to find the square root of a number.**

**Description :**

There are two primary categories of numbers in the C++ programming language: integers and floating points, with the latter supporting decimal places. Integers can be a short int, a regular int, or a long or double int. There are three types of floating point numbers in C++ float, double, and long double.

22:23

**SOURCE CODE :**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

    int a;

    cout << "\nEnter the number to find square root :";

    cin >> a;

    cout << "\n square root of " << a << " is  :" << sqrt(a);

    return 0;

}

**Output :**

**Graphical user interface, text

Description automatically generated**

**EXPERIMENT -2**

**AIM :**

**C++ program to find the sum of digits of number.**

**Description :**

There are two primary categories of numbers in the C++ programming language: integers and floating points, with the latter supporting decimal places. Integers can be a short int, a regular int, or a long or double int. There are three types of floating point numbers in C++ float, double, and long double.

**SOURCE CODE :**

#include<iostream>

using namespace std;

int main()

{     int n,sum=0;

    cout<<"\nenter number to find sum";

    cin>>n;

    while( n>0)

    {

      int a= n%10;

      sum = sum+a;

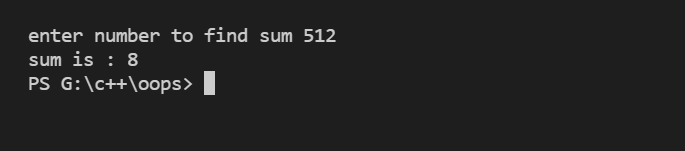
      n=n/10;

    }

    cout<<"sum is : "<<sum<<endl;

}

**Output :**

****

**EXPERIMENT -3**

**AIM :**

**C++ program to find to length of string**

**Description :**

There are two primary categories of numbers in the C++ programming language: integers and floating points, with the latter supporting decimal places. Integers can be a short int, a regular int, or a long or double int. There are three types of floating point numbers in C++ float, double, and long double.

**SOURCE CODE :**

#include<iostream>

#include<string>

using namespace std ;

int main()

{         string s;

          cout<<"\n enter the string to find its lenght :";

          getline(cin,s);

          cout<<"\n Length of given string is  :"<<s.length();

    return 0 ;

}

**Output:**

**Text

Description automatically generated**

**EXPERIMENT -4**

**AIM :**

**C++ program to copy one string to another .**

**Description :**

There are two primary categories of numbers in the C++ programming language: integers and floating points, with the latter supporting decimal places. Integers can be a short int, a regular int, or a long or double int. There are three types of floating point numbers in C++ float, double, and long double.

**SOURCE CODE :**

#include <iostream>

#include <string.h>

using namespace std;

int main()

{

  char str1[50], str2[50];

  cout << "Enter the string: ";

  cin >> str1;

  // copy the string

  strcpy(str2, str1);

  cout << "Copied String (str2): " << str2;

  return 0;

}

**Output:**

**Text

Description automatically generated**

**EXPERIMENT - 5**

**AIM :**

**C++ program to.**

**Description :**

There are two primary categories of numbers in the C++ programming language: integers and floating points, with the latter supporting decimal places. Integers can be a short int, a regular int, or a long or double int. There are three types of floating point numbers in C++ float, double, and long double.

**SOURCE CODE :**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

    int a;

    cout << "\nEnter the number to find  cube root :";

    cin >> a;

    cout << "\n cube root of " << a << " is  :" << cbrt(a);

    return 0;

}

**Output:**

Text

Description automatically generated

**CONSTRUCTOR AND DESTRUCTOR**  **EXPERIMENT -1**

**AIM :**

**Find the volume of cube using constructor**

**Description :**

**As the name suggest to construct space, or in direct words, Constructors are a unique class functions that do the job of initialising every object. Whenever, any object is created the constructor is called by the compile**

**We worked with initialising the object values, but what when the scope of the object ends, we must also destroy the object, right !. For this we use Destructors in C++. As soon as the object goes out of scope the compiler automatically destructs the object.**

**SOURCE CODE :**

//volume of cube using construtor

#include<iostream>

using namespace std ;

class cube{

        int a;

      public:

        cube();

        cube(int x){

          int a =x;

          cout<<"\nvolume of cube is "<<x\*x\*x;

          }

};

int main()

{

        int a;

        cout<<"\n Enter the side of cube :";

        cin>>a;

        cube c(a);

    return 0 ;

}

**OUTPUT :**

**Graphical user interface

Description automatically generated**

**EXPERIMENT -2**

**AIM :**

**Determine area of rectangle using constructor.**

**Description :**

**As the name suggest to construct space, or in direct words, Constructors are a unique class functions that do the job of initialising every object. Whenever, any object is created the constructor is called by the compile**

**We worked with initialising the object values, but what when the scope of the object ends, we must also destroy the object, right !. For this we use Destructors in C++. As soon as the object goes out of scope the compiler automatically destructs the object.**

**SOURCE CODE :**

//area of rectangle

#include<iostream>

using namespace std ;

class cube{

        int a,b,c;

      public:

        cube();

        cube(int x,int y,int z){

          int a =x;

          int b =y;

          int c =z;

          cout<<"\nArea  of rectangle is "<<x\*y\*z;

          }

};

int main()

{

        int l,b,h;

        cout<<"\n Enter the lenght , breadth and height :";

        cin>>l>>b>>h;

        cube c(l,b,h);

    return 0 ;

}

**Output :**

**Text

Description automatically generated**

**EXPERIMENT -3**

**AIM :**

**Write a program to show counter using constructer**

**Description :**

**As the name suggest to construct space, or in direct words, Constructors are a unique class functions that do the job of initialising every object. Whenever, any object is created the constructor is called by the compile**

**We worked with initialising the object values, but what when the scope of the object ends, we must also destroy the object, right !. For this we use Destructors in C++. As soon as the object goes out of scope the compiler automatically destructs the object.**

**SOURCE CODE :**

#include<iostream>

using namespace std;

class counter

{

    private:

        int count;

    public:

        counter ()  {

            count=0;

        }

        void inc\_count ()

        {

            count++;

        }

        int get\_count ()

        {

            return count;

        }

};

int main ()

{

    counter c1;

    cout<<"\nBefore calling Counter Function, Count = ";

    cout<<c1.get\_count ();

    c1.inc\_count ();

    cout<<"\n\nAfter calling Counter Function, Count = ";

    cout<<c1.get\_count ();

    cout<<"\n";

    return 0;

}

**Output :**

**Graphical user interface, text

Description automatically generated**

**EXPERIMENT -4**

**AIM :**

**Write a program to display date using constructor.**

**Description :**

**As the name suggest to construct space, or in direct words, Constructors are a unique class functions that do the job of initialising every object. Whenever, any object is created the constructor is called by the compile**

**We worked with initialising the object values, but what when the scope of the object ends, we must also destroy the object, right !. For this we use Destructors in C++. As soon as the object goes out of scope the compiler automatically destructs the object.**

**Source code :**

#include<iostream>

using namespace std ;

class date{

    public:

    date(){

        cout<<"\n TODAY DATE IS 4-10-2021";   }

}d;

int main()

{

    return 0 ;

}

**Output :**

**Text

Description automatically generated**

**EXPERIMENT -5**

**AIM :**

**C++ program of overload constructor**

**Description :**

**As the name suggest to construct space, or in direct words, Constructors are a unique class functions that do the job of initialising every object. Whenever, any object is created the constructor is called by the compile**

**We worked with initialising the object values, but what when the scope of the object ends, we must also destroy the object, right !. For this we use Destructors in C++. As soon as the object goes out of scope the compiler automatically destructs the object.**

**Source code :**

// C++ program to demonstrate constructor overloading

#include <iostream>

using namespace std;

class Person {

   private:

    int age;

   public:

    Person() {

        age = 20;

    }

    Person(int a) {

        age = a;

    }

    int getAge() {

        return age;

    }

};

int main() {

    Person person1, person2(45);

    cout << "Person1 Age = " << person1.getAge() << endl;

    cout << "Person2 Age = " << person2.getAge() << endl;

    return 0;

}

**Output :**

**Text

Description automatically generated**