**Programs:**

1. Write a C++ program to overload function called ‘square’ to calculate the square of an int variable and the square of a double variable.

#include <iostream>

using namespace std;

int square(int x){

cout <<"Square of integer number is "<< x\*x <<endl;

return x\*x;

}

double square(double x){

cout <<"square of double number is "<< x\*x <<endl;

return x\*x;

}

int main()

{

int x; double y;

cout << "Enter the Int Number: ";

cin >> x;

square(x);

cout << "Enter the double Number: ";

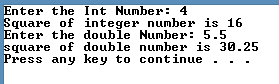
cin >> y;

square(y);

return 0;

}

**Output:**



1. Write a C++ program to overload function called ‘area’ to find the area of square rectangle and circle.

#include <iostream>

using namespace std;

const double pi = 3.14;

int area(int side){

int areaS = side \* side;

cout <<"The area of the Square is "<< side \* side <<endl;

return side\*side;

}

double area(double radius){

double areaC = pi\*radius\*radius;

cout << "The area of the Circle is "<< areaC <<endl;

return areaC;

}

int area(int length, int breadth){

int areaR = length \* breadth;

cout <<"The area of the rectangle is "<< areaR <<endl;

return areaR;

}

int main()

{

int side, length, breadth;

double radius;

cout << "Enter the side of the Square: ";

cin >> side;

area(side);

cout << "Enter the length of the rectangle: ";

cin >> length;

cout << "Enter the breadth of the rectangle: ";

cin >> breadth;

area(length, breadth);

cout << "Enter the radius of the circle: ";

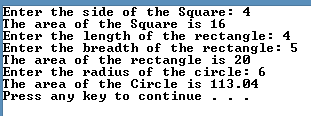
cin >> radius;

area(radius);

return 0;

}

**Output:**



1. Write a C++ program to overload function called ‘swap’ to swap two variables of integer, float and char types.

#include <iostream>

using namespace std;

void swap(int a, int b){

int temp;

temp = a;

a = b;

b = temp;

cout << "The value of int 'a' is " << a << " and the value of int 'b' is " << b <<endl;

return;

}

void swap(float a, float b){

float temp;

temp = a;

a = b;

b = temp;

cout << "The value of float 'c' is " << a << " and the value of float 'd' is " << b <<endl;

return;

}

void swap(char a, char b){

char temp;

temp = a;

a = b;

b = temp;

cout << "The value of char 'e' is " << a << " and the value of char 'f' is " << b <<endl;

return;

}

int main()

{

int a, b;

float c, d;

char e, f;

cout << "Enter the int value of a : ";

cin >> a;

cout << "Enter the int value of b: ";

cin >> b;

swap(a,b);

cout << "\nEnter the float value of c: ";

cin >> c;

cout << "Enter the float value of d: ";

cin >> d;

swap(c, d);

cout << "\nEnter the char value of e: ";

cin >> e;

cout << "Enter the char value of f: ";

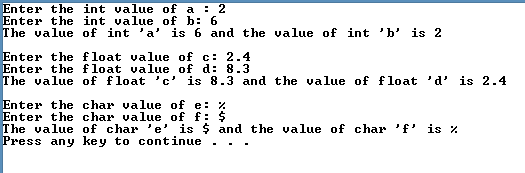
cin >> f;

swap(e, f);

return 0;

}

**Output:**



1. Write a C++ program to overload function called ‘sum’ that adds the elements of two multi-dimensional arrays for 2 integer arrays and 2 double arrays.

**#include <iostream>**

**using namespace std;**

**void sum(int x[2][2], int y[2][2]){**

**int sumArr[2][2] = {0};**

**cout<< "The sum of element of two integer multi-dimensional arrays are:"<<endl;**

**for(int i = 0; i<2; i++){**

**for(int j=0; j<2; j++){**

**sumArr[i][j] = x[i][j] + y[i][j];**

**}**

**}**

**for(int i = 0; i<2; i++){**

**for(int j=0; j<2; j++){**

**cout<<sumArr[i][j]<<"\t";**

**}**

**cout << "\n";**

**}**

**return;**

**}**

**void sum(float x[2][2], float y[2][2]){**

**float sumArr[2][2] = {0};**

**cout<< "The sum of element of two float multi-dimensional arrays are:"<<endl;**

**for(int i = 0; i<2; i++){**

**for(int j=0; j<2; j++){**

**sumArr[i][j] = x[i][j] + y[i][j];**

**}**

**}**

**for(int i = 0; i<2; i++){**

**for(int j=0; j<2; j++){**

**cout<<sumArr[i][j]<<"\t";**

**}**

**cout << "\n";**

**}**

**return;**

**}**

**int main()**

**{**

**int mdArr1[2][2] = {0};**

**int mdArr2[2][2] = {0};**

**float mdArr3[2][2] = {0};**

**float mdArr4[2][2] = {0};**

**int i=0, j=0;**

**cout<<"Enter the value of elements for integer array1 at the index:"<<endl;**

**for(i = 0; i<2; i++){**

**for(j=0; j<2; j++){**

**cout <<"["<<i<<"]"<< "["<<j<<"]"<<":\t";**

**cin >> mdArr1[i][j];**

**}**

**}**

**cout<<"Enter the value of elements for integer array2 at the index:"<<endl;**

**for(i = 0; i<2; i++){**

**for(j=0; j<2; j++){**

**cout <<"["<<i<<"]"<< "["<<j<<"]"<<":\t";**

**cin >> mdArr2[i][j];**

**}**

**}**

**sum(mdArr1,mdArr2);**

**cout<<"Enter the value of elements for integer array1 at the index:"<<endl;**

**for(i = 0; i<2; i++){**

**for(j=0; j<2; j++){**

**cout <<"["<<i<<"]"<< "["<<j<<"]"<<":\t";**

**cin >> mdArr3[i][j];**

**}**

**}**

**cout<<"Enter the value of elements for integer array1 at the index:"<<endl;**

**for(i = 0; i<2; i++){**

**for(j=0; j<2; j++){**

**cout <<"["<<i<<"]"<< "["<<j<<"]"<<":\t";**

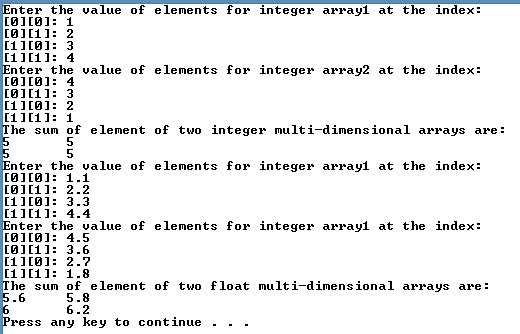
**cin >> mdArr4[i][j];**

**}**

**}**

**sum(mdArr3,mdArr4);**

**return 0;**

**}**

**Output:**

1. Write a recursive function power (base, exponent) that when invoked returns base^exponent. E.g. power (3,4) = 3\*3\*3\*3. Exponent has to be greater or equal to 1.

#include<iostream>

using namespace std;

int power(int base, int exponent){

if(exponent == 1){

return base;

}

return base\*power(base, exponent -1);

}

int main(){

int base = 0, exponent = 0;

cout << "Enter the base value:\t";

cin >> base;

cout << "Enter the exponent:\t";

cin >> exponent;

if(exponent < 1){

cout << "Please enter exponent greater or equal to 1";

}else{

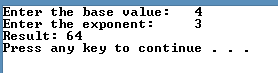
cout<<"Result: "<<power(base, exponent);

}

return 0;

}

**Output:**



1. Write a C++ Program to implement Linear Search using recursion.

#include<iostream>

using namespace std;

int linearSearch(int arr[], int searchElement, int index , int size){

if(index == size){

cout << "element not found."<<endl;

return 0 ;

}

if(searchElement == arr[index] ){

cout << "Element found at index: "<< index << endl;;

return index;

}

return linearSearch(arr, searchElement, index +1, size);

}

int main(){

int size = 5;

int arr[5] = { 1, 3, 5, 6, 9};

int searchElement = 0;

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

cin >> searchElement;

linearSearch(arr, searchElement, 0, size);

return 0;

}

**Output:**

**Case1: Element found.**



**Case2: Element not found.**

