LAB MANUAL

Course : CP1244 OOPS Lab

Class : Bachelor in Computer Applications

Semester : 2

Prepared By : SYAMA M NAIR

COLLEGE OF APPLIED SCIENCE

PERISSERY

TABLE OF CONTENTS

1.LARGEST EVEN AND ODD

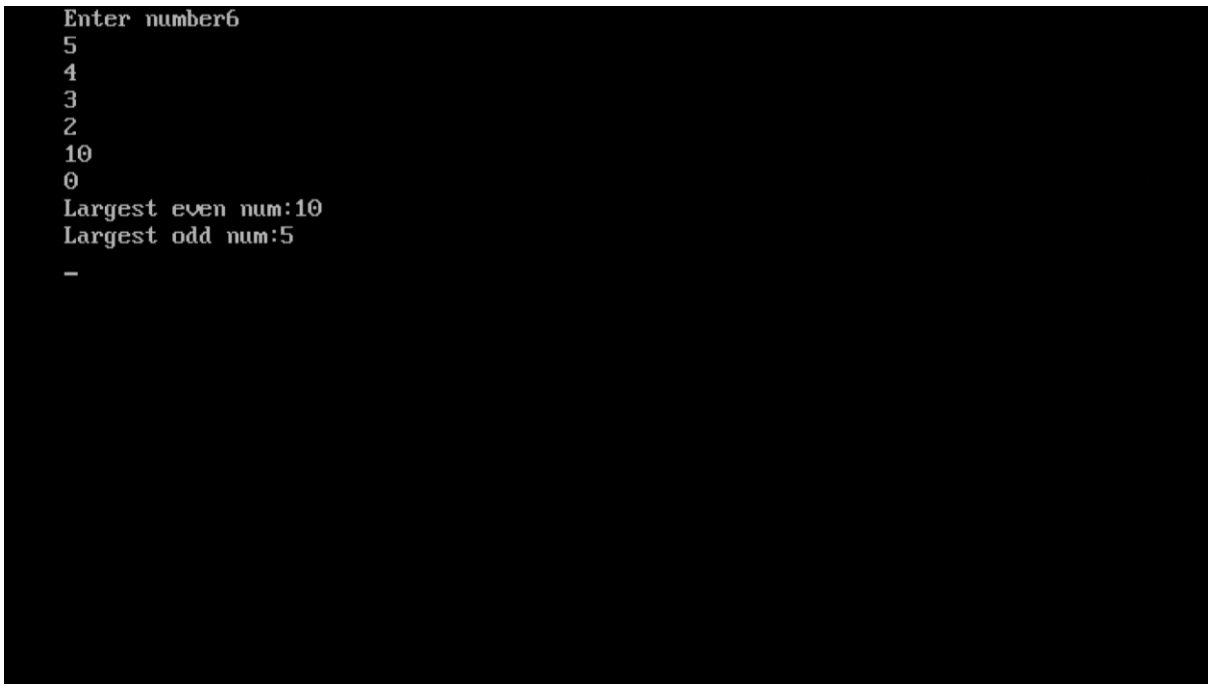
- **Aim:**

  To write a program to find largest even and odd numbers.

- **Program:**

```
#include <iostream>
using namespace std;
int main() {
    int num, meven= 0, modd = 1;
    // clrscr();
    cout<<"Enter number";
    do {
            cin>>num;
            if(num % 2 == 0) {
                    if(num > meven) {
                            meven = num;
                    }
            } else {
                    if(num > modd)
                            modd = num;
            }
    } while(num);
    return 0;
}
```

- **Output:**



2.MATRIX

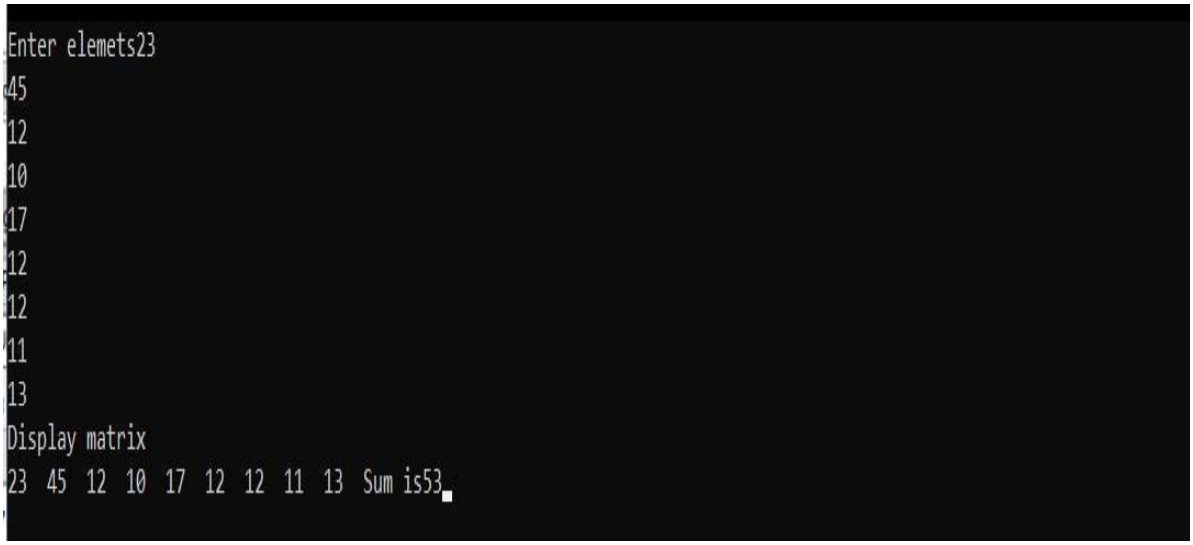
- **Aim:**

  To write a program to find the sum of diagonal elements in a matrix.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
using namespace std; // not needed in turbo

int main() {
    int i, j, a[3][3], sum = 0;
    // clrscr();
    cout<<"Enter elemets";
    for(i = 0; i < 3; i++)
        for(j = 0; j < 3; j++)
            cin>>a[i][j];
    cout<<"Display matrix"<<endl;

    for(i = 0; i < 3; i++)
        for(j = 0; j < 3; j++)
            cout<<a[i][j]<<"  ";

    for(i = 0; i < 3; i++)
        for(j = 0; j < 3; j++)
            if(i == j)
                sum += a[i][j];

    cout<<"Sum is"<<sum;
    _getch(); // in turbo getch()
    return 0;
}
```

- **Output:**


Enter elemets23
45
12
10
17
12
12
11
13
Display matrix
23 45 12 10 17 12 12 11 13 Sum is53

3.LARGEST OF THREE NUMBERS

- **Aim:**

  To write a program to find the largest of three numbers.

- **Program:**

```
#include <iostream> // in turbo iostream.h
#include <conio.h>
using namespace std; // not needed in turbo
int main() {
    int a, b, c;
    // clrscr();
    cout<<"Enter 3 numbers:"<<endl;
    cin>>a>>b>>c;
    if(a > b && a > c)
            cout<<a<<"is largest"<<endl;
    else if(b > a && b > c)
            cout<<b<<"is largest"<<endl;
    else
            cout<<c<<"is largest"<<endl;
    _getch(); // getch() in turbo
    return 0;
}
```

- **Output:**

4.SORT ELEMENTS

- **Aim:**

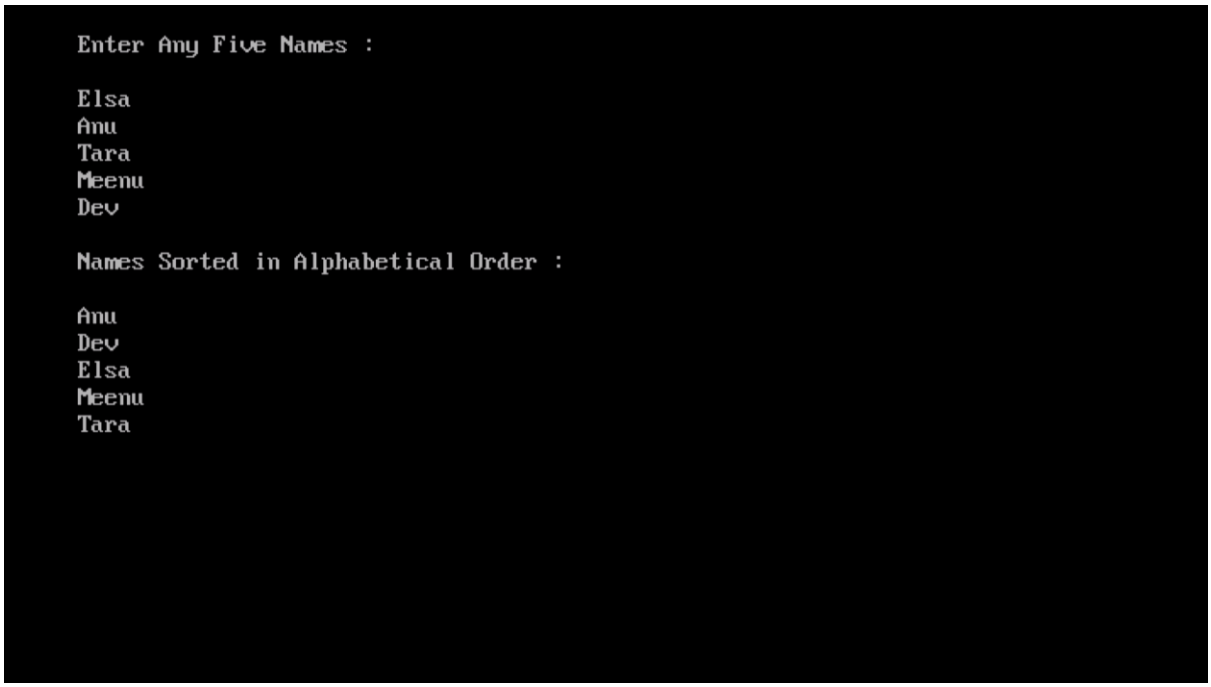
  To write a program to sort elements in ascending order using string functions.

- **Program:**

```
#include<iostream.h>
#include<string.h>
#include<stdio.h>
#include<conio.h>
int main()
{
clrscr();
        char str[5][20], t[20];
        int i, j;
        cout<<"\n Enter Any Five Names : \n\n";
        for(i=0; i<5; i++)
        {
                cout<<" ";
                cin>>str[i];
        }
        for(i=1; i<5; i++)
        {
                for(j=1; j<5; j++)
                {
                        if(strcmp(str[j-1], str[j])>0)
                        {
                                strcpy(t, str[j-1]);
                                strcpy(str[j-1], str[j]);
                                strcpy(str[j], t);
                        }
                }
        }
        cout<<"\n Names Sorted in Alphabetical Order : \n\n";
```

```
        for(i=0; i<5; i++)
        {
                cout<<" ";
                cout<<str[i]<<"\n";
        }
        getch();
        return 0;

}
```

- **Output:**



5.PROGRAM TO FIND DIGIT BY DIGIT

- **Aim:**

  To write a program to read a number n and display it in word.

- **Program:**

```
#include <iostream> // in turbo iostream.h
#include <conio.h>
using namespace std; // not needed in turbo

class Number {
public:
        int z, b, r[10], x, j, i;
        void getdata();
        void words();
};

void Number::getdata() {
        cout<<"\n Enter the data:";
        cin>>z;
        if(z == 0)
                cout<<"Zero";
}

void Number::words() {
        int i = 0;
        while( z > 0) {
                b = z % 10;
                r[i] = b;
                z = z / 10;
                i++;
        }
        j = i - 1;
        for(i = j; i >= 0; i--) {
                int c;
```

```
c = r[i];
switch(c) {
        case 0:
                cout<<"ZERO"<<endl;
                break;
        case 1:
                cout<<"ONE"<<endl;
                break;
        case 2:
                cout<<"TWO"<<endl;
                break;
        case 3:
                cout<<"Three"<<endl;
                break;
        case 4:
                cout<<"FOUR"<<endl;
                break;
        case 5:
                cout<<"FIVE"<<endl;
                break;
        case 6:
                cout<<"SIX"<<endl;
                break;
        case 7:
                cout<<"SEVEN"<<endl;
                break;
        case 8:
                cout<<"EIGHT"<<endl;
                break;
        case 9:
                cout<<"NINE"<<endl;
                break;
        default:
```

```
                                    cout<<"Something went wrong";
                                    _getch(); // getch() in turbo
                                    return;
                            }
                    }
            }

            int main() {
                    // clrscr();
                    Number n;
                    n.getdata();
                    n.words();
                    _getch(); // getch() in turbo
                    return 0;
            }
```

- **Output:**

6.ARITHMATIC OPERATIONS

- **Aim:**

  To write a program to perform arithematic operations using function.

- **Program:**

```
#include <iostream> // in turbo iostream.h
#include <conio.h>
using namespace std; // not needed in turbo

class Arithematic {
public:
        int ans;
        void sum(int, int);
        void sub(int, int);
        void mul(int, int);
        void div(int, int);
};

void Arithematic::sum(int a, int b) {
        ans = a + b;
        cout<<a<<"+"<<b<<"="<<ans<<endl;
}

void Arithematic::sub(int a, int b) {
        ans = a - b;
        cout<<a<<"-"<<b<<"="<<ans<<endl;
}

void Arithematic::mul(int a, int b) {
        ans = a * b;
        cout<<a<<"*"<<b<<"="<<ans<<endl;
}
```

```
void Arithematic::div(int a, int b) {
	ans = a / b;
	cout<<a<<"/"<<b<<"="<<ans<<endl;
}

int main() {
	Arithematic a;
	int choice, x, y;
	cout<<"Enter your choice:\n";
	cout<<" 1. Addition\n 2. Multiplication\n 3. Division\n 4. Subtraction\n";
	cin>>choice;
	cout<<"Enter two numbers:\n";
	cin>>x>>y;
	switch(choice) {
		case 1:
			a.sum(x, y);
			break;
		case 2:
			a.mul(x, y);
			break;
		case 3:
			a.div(x, y);
			break;
		case 4:
			a.sub(x, y);
			break;
		default:
			cout<<"Something went wrong";
			_getch(); // getch() in turbo
			return 0;
	}
	_getch(); // getch() in turbo
	return 0;
```

```
}
```

- **Output:**


Enter your choice:
1. Addition
2. Multiplication
3. Division
4. Subtraction
1
Enter two numbers:
1
2
1+2=3

7.SWAPPING NUMBERS

- **Aim:**

  To write a c++ program to swap two numbers.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
#include <stdio.h>
using namespace std;

class Numbers {
public:
        void swap();
};

void Numbers::swap() {
        int temp, a, b;
        cout<<"Enter numbers to swap"<<endl;
        cin>>a>>b;
        cout<<"Before swapping num1 ="<<a<<" num2 = "<<b<<endl;
        temp = a;
        a = b;
        b = temp;
        cout<<"After swapping num1 = "<<a<<" num2 = "<<b<<endl;
}

int main() {
        Numbers ob;
        ob.swap();
        _getch(); // getch() in turbo
        return 0;
}
```

- **Output:**

8.CLASS DISTANCE

- **Aim:**

  To write a program to find the distance which contains feet and inches as data members and compute the distance.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h> using namespace std;
class Distance {
public:
float feet, inches; void
getData(); void
dispalyData();
};
void Distance::getData() {
cout<<"\n Enter the value of feet";
cin>>feet;
cout<<"\n Enter the value of inches";
cin>>inches;
}
void Distance::dispalyData() {
cout<<"\n Distance = "<<feet<<"\n Inches = "<<inches;
}
int main() {
Distance d;
// clrscr();
d.getData();
d.dispalyData();
_getch(); // getch() in turbo
return 0;
}
```

- **Output:**

9.DYNAMIC MEMORY ALLOCATION

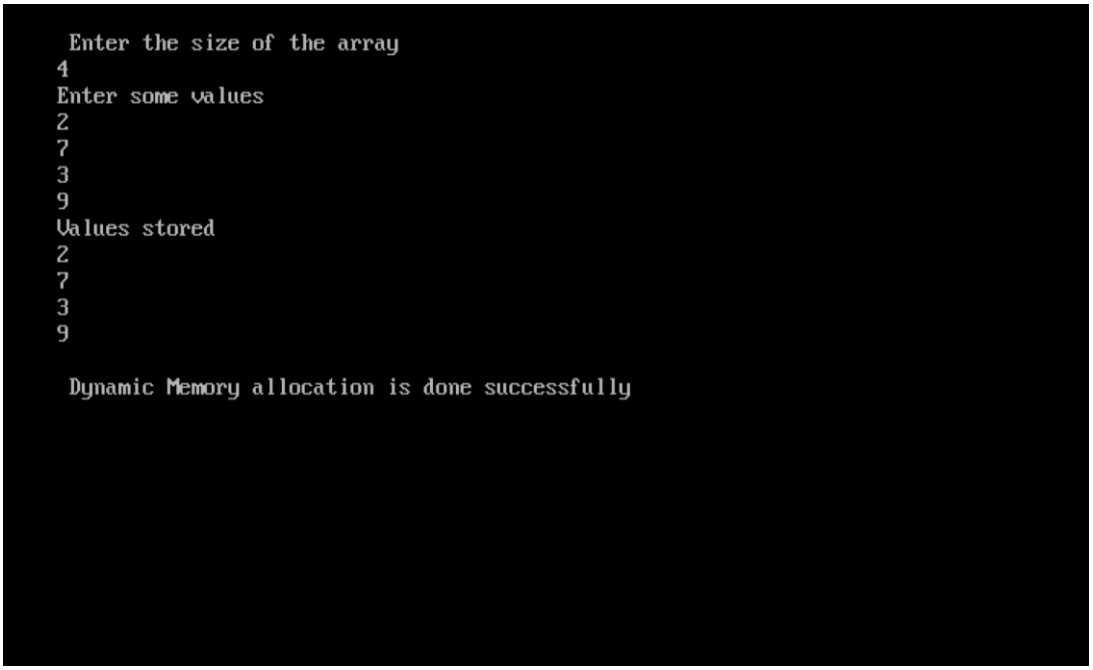
- **Aim:**

  To write a program to illustrate dynamic memory allocation using new and delete operator.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
#include <stdlib.h>
using namespace std; // not needed in turbo
int main() {
        int *ptr, n;
        // clrscr();
        cout<<"\n Enter the size of the array \n";
        cin>>n;
        ptr = new int[n];
        if(!ptr) {
                cout<<"Allocation field\n";
                exit(1);
        } else {
                cout<<"Enter some values"<<endl;
                for(int i = 0; i < n; i++)
                        cin>>ptr[i];
                cout<<"Values stored"<<endl;
                for(int i = 0; i < n; i++)
                        cout<<ptr[i]<<endl;
        }
        cout<<"\n Dynamic Memory allocation is done successfully";
        delete[] ptr;
        _getch(); // getch() in turbo
        return 0;
}
```

- **Output:**


Enter the size of the array
4
Enter some values
2
7
3
9
Values stored
2
7
3
9
Dynamic Memory allocation is done successfully

10.FIBINOCCI SERIES

- **Aim:**

  To write a program to implement Fibinocci series.

- **Program:**

```
#include <iostream> // iostream in turbo
#include <conio.h>
using namespace std; // not needed in turbo

int main() {
        int first = 0, second = 1, next, n, i;
        // clrscr();
        cout<<"\n Enter the limit"<<endl;
        cin>>n;
        cout<<first<<"\t"<<second;
        for(i = 3; i <= n; i++) {
                next = first + second;
                cout<<"\t"<<next;
                first = second;
                second = next;
        }
        _getch(); // getch() in turbo
        return 0;
}
```

- **Output:**

11.STATIC FUNCTIONS

- **Aim:**

  To write a c++ program to implement static function.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
using namespace std; // not need in turbo

class Item {
        static int count;
        int number;
public:
        void setData(int a) {
                number = a;
                count++;
        }

        void getCount(){
                cout<<"count:"<<count<<endl;
        }
};

int Item::count = 0;

int main() {
        Item a, b, c;
        a.getCount();
        b.getCount();
        a.setData(100);
        b.setData(200);
        c.setData(300);
        cout<<"After reading data"<<endl;
        a.getCount();
```

```
        b.getCount();
        c.getCount();
        _getch(); // getch() in turbo
        return 0;

}
```

- **Output:**

12.INHERITENCE

- **Aim:**

  To write a program to show single inheritence between two classes.

- **Program:**

```
#include<iostream.h>
#include<conio.h>
class Person
{
protected:
char name[15];
int age;
};
class desc: public Person
{
float h,w;
public:
void getdata()
{
cout<<"Enter name and age\n";
cin>>name>>age;
cout<<"Enter height and weight\n";
cin>>h>>w;
}
void show()
{
cout<<"\nName:"<<name<<"\nAge:"<<age;
cout<<"\nHeight:"<<h<<"Feet"<<"\nWeight:"<<w<<"Kg";
}
};
void main()
{
clrscr();
desc x;
```

```
x.getdata();
x.show();
getch();
}
```

- **Output:**

13.MULTILEVEL INHERITANCE

- **Aim:**

  To write a program to implement Multilevel Inheritence.

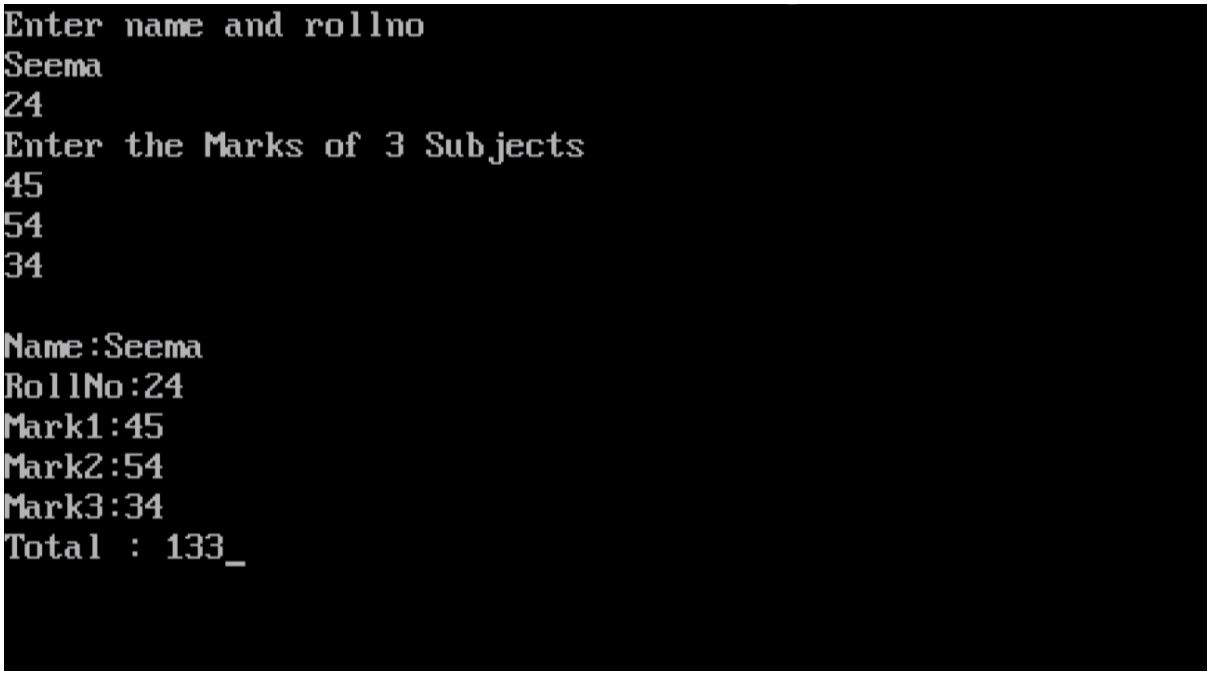
- **Program:**

```
#include<iostream.h>
#include<conio.h>
class student
{
protected:
char name[15];
int rollno;
public:
void get();
void put();
};
void student::get()
{
cout<<"Enter name and rollno\n";
cin>>name>>rollno;
}
void student::put()
{
cout<<"\nName:"<<name;
cout<<"\nRollNo:"<<rollno;
}
class test: public student
{
protected:
float m1,m2,m3;
public:
void gets();
```

```
void puts();
};
void test::gets()
{
cout<<"Enter the Marks of 3 Subjects\n";
cin>>m1>>m2>>m3;
}
void test::puts()
{
cout<<"\nMark1:"<<m1;
cout<<"\nMark2:"<<m2;
cout<<"\nMark3:"<<m3;
}
class result:public test
{
float total;
public:
void display();
};
void result::display()
{
total=m1+m2+m3;
put();
puts();
cout<<"\nTotal : "<<total;

}
void main()
{
clrscr();
result s1;
s1.get();
s1.gets();
```

```
s1.display();
}
```

- **Output:**



14.FRIEND FUNCTION

- **Aim:**

  To write a program tocalculate the sum of integers of both classes using friend function.

- **Program:**

```
#include<iostream.h>
#include<conio.h>
class test;
class sum
{
 int a;
 public:
 void get()
 {
 cout <<"\nEnter a number:";
 cin >>a;
 }
 friend void add(sum,test);
};
class test
{
int b;
public:
void get()
{
cout<<"\nEnter a number:";
cin>>b;
}
friend void add(sum, test);
};
void add(sum x, test y)
{
```

```
 cout<<"\nSum of two numbers :"<<x.a+y.b;
 }

void main()
{
clrscr();
sum s;
test t;
s.get();
t.get();
add(s,t);
getch();
}
```

- **Output:**

15.UNARY OPERATOR OVERLOADING

- **Aim:**

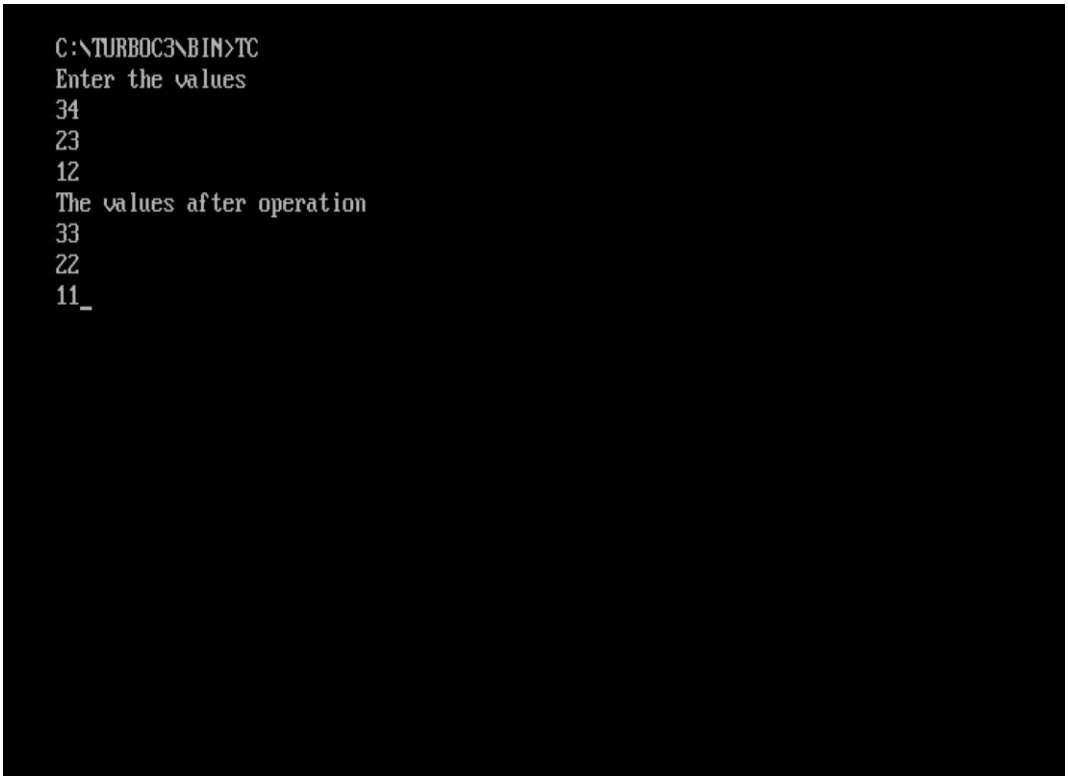
  To write a c++ program to illustrate unary operator overloading.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
using namespace std;

class Test {
public:
        int a, b, c;
        void readData() {
                cout<<"Enter the values"<<endl;
                cin>>a>>b>>c;
        }

        void operator-() {
                --a;
                --b;
                --c;
        }

        void dispaly() {
                cout<<"The values after operation"<<endl;
                cout<<a<<endl<<b<<endl<<c;
        }
};

int main() {
        Test obj;
        obj.readData();
        -obj;
```

```
        obj.dispaly();
        _getch(); // getch() in turbo
        return 0;
}
```

- **Output:**



16.BINARY OPERATOR OVERLOADING

- **Aim:**

  To write a c++ program to implement binary operator overloading.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
using namespace std;

class Complex {
public:
        int a, b;
        // clrscr();

        void readdata() {
                cout<<"Enter the numbers: \n";
                cin>>a>>b;
        }

        Complex operator+(Complex ob) {
                Complex temp;
                temp.a = a + ob.a;
                temp.b = b + ob.b;
                return temp;
        }

        Complex operator-(Complex ob) {
                Complex temp;
                temp.a = a - ob.a;
                temp.b = b - ob.b;
                return temp;
        }

        void display() {
```

```
                cout<<a<<"\t"<<b;
        }
};

int main() {
        Complex o1, o2, o3, o4;
        // clrscr();
        cout<<"Input \n";
        o1.readdata();
        o2.readdata();
        o3 = o1 + o2;
        o4 = o1 - o2;
        cout<<"The output"<<endl;
        o3.display();
        cout<<endl;
        o4.display();
        _getch(); // getch() in turbo
        return 0;
}
```

- **Output:**

17.VIRTUAL FUNCTIONS

- **Aim:**

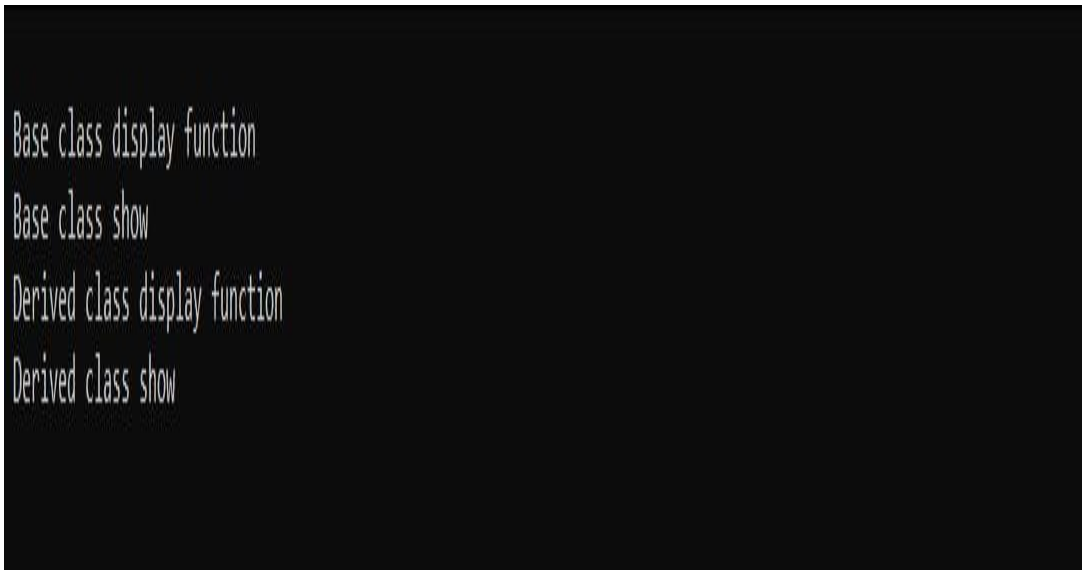
  To write a c++ program to implement virtual function.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
using namespace std; // not needed in turbo

class First {
public:
        virtual void display(void) {
                cout<<"\n Base class display function";
        }

        virtual void show(void) {
                cout<<"\n Base class show";
        }
};

class Second: public First {
public:
        void display() {
                cout<<"\n Derived class display function";
        }

        void show() {
                cout<<"\n Derived class show";
        }
};

int main() {
        // clrscr();
        First obj, *s;
```

```
        Second obj1;
        s = &obj;
        s-> display();
        s-> show();
        s = &obj1;
        s-> display();
        s-> show();
        _getch(); // getch() in turbo
        return 0;
}
```

- **Output:**



18.FUNCTION OVERLOADING

- **Aim:**

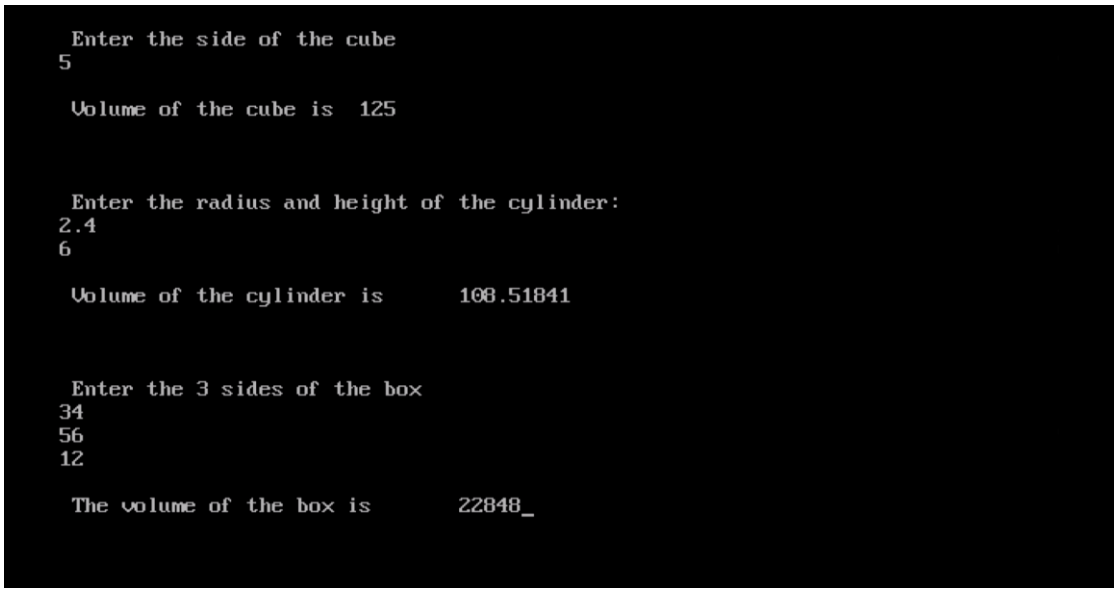
  To write a Program to implement function overloading.

- **Program:**

```
#include <iostream> // iostream.h in turbo
#include <conio.h>
using namespace std; // not needed in turbo

class Volume {
public:
        void getVolume(int);
        void getVolume(float, float);
        void getVolume(int, int, int);
};

void Volume::getVolume(int a) {
        int v = a * a * a;
        cout<<"\n Volume of the cube is \t"<<v;
}

void Volume::getVolume(float x, float y) {
        float v = 3.14 * (x * x) * y;
        cout<<"\n Volume of the cylinder is \t"<<v;
}

void Volume::getVolume(int a, int b, int c) {
        int v = a * b * c;
        cout<<"\n The volume of the box is \t"<<v;
}

int main(int argc, char const *argv[])
{
```

```
    int a, b, c;
    float x, y;
    Volume obj;
    cout<<"\n Enter the side of the cube"<<endl;
    cin>>a;
    obj.getVolume(a);
    cout<<endl<<endl<<endl;
    cout<<"\n Enter the radius and height of the cylinder:"<<endl;
    cin>>x>>y;
    obj.getVolume(x, y);
    cout<<endl<<endl<<endl;
    cout<<"\n Enter the 3 sides of the box"<<endl;
    cin>>a>>b>>c;
    obj.getVolume(a, b, c);
    _getch();  // getch() in turbo
    return 0;
}
```

- **Output:**



19.EXCEPTION HANDLING

- **Aim:**

  To write a program to implement Exception Handling in C++.

- **Program:**

```
#include <iostream>
using namespace std;
void sqr()
{
   int s;
cout<<"\nEnter a number";
cin>>s;
if(s>0)
{
   cout<<"Square ="<<s*s;
}
else
{
   throw(s);
}
}
int main()
{
   try
   {
      sqr();
      sqr();
   }
   catch(int j)
   {
      cout<<"\nCaught the exception";
   }
   return 0;
}
```

- **Output:**


Enter a number5
Square =25
Enter a number
0
Caught the exception