Name: Jeevan Sudhakar Patil

Department: Computer

Class: SE Div: C

Roll no: 48

Subject: Object Oriented Programming

EXPERIMENT NO: - 4

**CODE: -**

*/\*Write a C++ program that creates an output file, write information to it, closes the file, open it again as an input file and read the information from file.\**/

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

fstream file; /*/object of fstream file*

file.open("sample.txt",ios::out|ios::app); /*/opening file “sample.txt” in out(write) mode*

if(!file)

{

cout<<"Error in creating file!!"<<endl;

return 0;

}

cout<<"File created successfully"<<endl;

file<<"ABCD ";  *//write text in the file*

file.close(); /*/closing the file*

file.open("sample.txt",ios::in); *//again open the file in the read mode*

if(!file)

{

cout<<"Error in opening file!!"<<endl;

return 0;

}

string line;  *//to read contents from file*

cout<<"File content : ";

while(!file.eof())

{

file>>line; //to read the single character from the file

cout<<line<<endl; *//read the content display on the screen*

}

file.close();  *//file close*

return 0;

}

**Output :**

comp@ubuntu:~/SEC48$ g++ Exp4.cpp

comp@ubuntu:~/SEC48$ ./a.out

File created successfully

File content : ABCD

ABCD

Name: Jeevan Sudhakar Patil

Department: Computer

Class: SE Div: C

Roll no: 48

Subject: Object Oriented Programming

**EXPERIMENT NO. 3**

**CODE: -**

/*\*Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications From this class derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float).Write a program that instantiates the book and tape class, allows user to Enter data and displays the data members. If an exception is caught, replace all the data member values with zero values. \**/

#include<iostream>

#include<stdio.h>

using namespace std;

class publication *//declaring the base class publications*

{

private:

string title;

float price;

public:

void add()

{

cout<<"Enter the Publication information : "<<endl;

cout<<"Enter Title of the publication : ";

cin>>title;

cout<<"Enter the Price of the publication : ";

cin>>price;

}

void display()

{

cout<<"Title of publication : "<<title<<endl;

cout<<"Publication price : "<<price<<endl;

}

}; /*/Base class publications closed*

//declaring class book which inherite class publication in public mode

class book:public publication

{

private:

int page\_count;

public:

void add\_book()

{

try

{

add(); *//calling the function from the parent class (publication)*

cout<<"Enter Page count of book : ";

cin>>page\_count;

if (page\_count<=0)

{

throw page\_count;

}

}  *//try block closed*

catch(...)

{

cout<<"Invalid page count!";

page\_count=0;

}

}  *//close function add\_book*

void display\_book()

{

display();  *//function from parent class*

cout<<"Page count : "<<page\_count<<endl;

}

}; *//book class closed(child of the publications)*

//declaring class tape which inherite class publication in public mode

class tape:public publication

{

private:

float play\_time;

public:

void add\_tape()

{

try

{

add(); *//calling the function from parent class*

cout<<"Enter play time of the tape : ";

cin>>play\_time;

if (play\_time<=0)

{

throw play\_time;

}

}

catch(...)

{

cout<<"Invalid play time!";

play\_time=0;

}

}

void display\_tape()

{

display();

cout<<"Play time : "<<play\_time<<endl;

}

};  *//class tape closed*

int main()

{

book b1[10]; *//object of class book*

tape t1[10]; *//object of class tape*

int ch, b\_count=0, t\_count=0;

do

{

cout<<"\n\n\*\*\*\*PUBLICATION DATABASE SYSTEM\*\*\*\*\*\*\*\*"<<endl;

cout<<"---------------MENU---------------"<<endl;

cout<<"\t1.Add information to books"<<endl;

cout<<"\t2.Add information to tape"<<endl;

cout<<"\t3.Display book information"<<endl;

cout<<"\t4.Dispay tape information"<<endl;

cout<<"\t5.Exit"<<endl;

cout<<"\nEnter your choice : ";

cin>>ch;

switch(ch)

{

case 1:

b1[b\_count].add\_book();

b\_count++;

break;

case 2:

t1[t\_count].add\_tape();

t\_count++;

break;

case 3:

cout<<"\n\*\*\*\*\*BOOK PUBLICATION DATABASE\*\*\*\*\*"<<endl;

for(int j=0; j<b\_count;j++)

{

b1[j].display\_book();

}

break;

case 4:

cout<<"\n\*\*\*\*\*TAPE PUBLICATION DATABASE\*\*\*\*\*"<<endl;

for(int j=0; j<t\_count;j++)

{

t1[j].display\_tape();

}

break;

case 5:

cout<<"End of the program"<<endl;

break;

}

}while (ch!=5);

return 0;

} *//main function closed*

**Output:**

comp@ubuntu:~/SEC48$ g++ Exp3.cpp

comp@ubuntu:~/SEC48$ ./a.out

\*\*\*\*PUBLICATION DATABASE SYSTEM\*\*\*\*\*\*\*\*

---------------MENU---------------

1.Add information to books

2.Add information to tape

3.Display book information

4.Dispay tape information

5.Exit

Enter your choice : 1

Enter the Publication information :

Enter Title of the publication : sunshine

Enter the Price of the publication : 800

Enter Page count of book : 1200

\*\*\*\*PUBLICATION DATABASE SYSTEM\*\*\*\*\*\*\*\*

---------------MENU---------------

1.Add information to books

2.Add information to tape

3.Display book information

4.Dispay tape information

5.Exit

Enter your choice : 2

Enter the Publication information :

Enter Title of the publication : tape

Enter the Price of the publication : 900

Enter play time of the tape : 212

\*\*\*\*PUBLICATION DATABASE SYSTEM\*\*\*\*\*\*\*\*

---------------MENU---------------

1.Add information to books

2.Add information to tape

3.Display book information

4.Dispay tape information

5.Exit

Enter your choice : 3

\*\*\*\*\*BOOK PUBLICATION DATABASE\*\*\*\*\*

Title of publication : sunshine

Publication price : 800

Page count : 1200

\*\*\*\*PUBLICATION DATABASE SYSTEM\*\*\*\*\*\*\*\*

---------------MENU---------------

1.Add information to books

2.Add information to tape

3.Display book information

4.Dispay tape information

5.Exit

Enter your choice : 4

\*\*\*\*\*TAPE PUBLICATION DATABASE\*\*\*\*\*

Title of publication : tape

Publication price : 900

Play time : 212

\*\*\*\*PUBLICATION DATABASE SYSTEM\*\*\*\*\*\*\*\*

---------------MENU---------------

1.Add information to books

2.Add information to tape

3.Display book information

4.Dispay tape information

5.Exit

Enter your choice : 5

End of the program

Name: Jeevan Sudhakar Patil

Department: Computer

Class: SE Div: C

Roll no: 48

Subject: Object Oriented Programming

# EXPERIMENT NO. 1

**CODE: -**

*/\* Implement a class Complex which represents the Complex Number data type. Implement the*

*following*

*1. Constructor (including a default constructor which creates the complex number 0+0i).*

*2. Overload operator+ to add two complex numbers.*

*3. Overload operator\* to multiply two complex numbers.*

*4. Overload operators << and >> to print and read Complex Numbers. \*/*

#include <iostream>

using namespace std;

class complex

{

public:

float x,y;

public:

complex() *//default constructor*

{

x=0;

y=0;

}

complex operator+(complex);  *//declaration of function to overload + operator*

complex operator\*(complex);  *//function to overload \* operator*

complex operator-(complex);  *//function to overload – operator*

friend istream &operator>>(istream &input,complex &t)  *//friend def*

{

cout<<"Enter the real part : ";

input>>t.x;

cout<<"Enter the imaginary part : ";

input>>t.y;

}

friend ostream &operator<<(ostream &output,complex &t)  *//friend def*

{

output<<t.x<<" + "<<t.y<<"i"<<endl;

}

};  *//class closing*

complex complex::operator+(complex c)

{

complex temp;

temp.x=x+c.x;

temp.y=y+c.y;

return temp;

}

complex complex::operator-(complex c)

{

complex temp;

temp.x=x-c.x;

temp.y=y-c.y;

return temp;

}

complex complex::operator\*(complex c)

{

complex temp2;

temp2.y=(y\*c.x)+(x\*c.y);

temp2.x=(x\*c.x)-(y\*c.y);

return temp2;

}

int main()

{

complex x1,cn1,cn2,cn3,cn4,cn5;  *//object of the class*

cout<<"Default constructor value = "<<x1<<endl;  *//calling the friend operator<<function*

cout<<"Enter the first number : "; *//Pass first complex no through object C1*

cin>>cn1;  *//calling the friend operator function>>function*

cout<<"Enter the second number : ";  *//Pass first complex no through object C2*

cin>>cn2;  *//calling the friend operator function>>function*

cn3=cn1+cn2;  *//calling of operator + function (Binary Operator Overloading)*

cn4=cn1\*cn2;  *//calling of operator \* function (Binary Operator Overloading)*

cn5=cn1-cn2;  *//calling of operator - function (Binary Operator Overloading)*

cout<<"The first number is : "<<cn1<<endl; *//calling the friend operator <<function*

cout<<"The second number is : "<<cn2<<endl;

cout<<"The addition is : "<<cn3<<endl;  *//calling the friend operator <<function*

cout<<"The multiplication is : "<<cn4<<endl;

cout<<"The substraction is : "<<cn5<<endl;

return 0;

} *//main closed*

**Output :**

comp@ubuntu:~/SEC48$ g++ Exp1.cpp

comp@ubuntu:~/SEC48$ ./a.out

Default constructor value = 0 + 0i

Enter the first number : Enter the real part : 8

Enter the imaginary part : 3

Enter the second number : Enter the real part : 5

Enter the imaginary part : 1

The first number is : 8+ 3i

The second number is : 5 + 1i

The addition is : 13 + 4i

The multiplication is : 43+23i

The substraction is : 3+ 2i

Name: Jeevan Sudhakar Patil

Department: Computer

Class: SE Div: C

Roll no: 48

Subject: Object Oriented Programming

# EXPERIMENT NO. 6

**CODE: -**

*/\*Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone number etc) using vector container. \*/*

#include <iostream> // Standard input-output stream header file

#include <algorithm> // STL algorithms for operations on data structures

#include <string>

#include <vector> // Header file for STL vector library

using namespace std;

class person { // Class to store person details

public:

char name[10]; // Name of person

string DOB; // Date of Birth of person (string for flexibility)

long pno; // Phone number of person

int id; // Unique ID for each person

bool operator==(const person& p1) {

return id == p1.id;

}

bool operator<(const person& p1) {

return id < p1.id;

}

};

vector<person> o1; // Global vector to store multiple persons

void print(person &p1); // Function to print details of a person

void display(); // Function to display all persons in the vector

void insert(); // Function to insert a new person

void search(); // Function to search for a person by id

void dlt(); // Function to delete a person by id

// Comparison function for sorting persons by id

bool compare(const person &p1, const person &p2) {

return p1.id < p2.id;

}

int main() {

int ch;

do {

// Menu for user input

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

cout << "\n1.Insert";

cout << "\n2.Display";

cout << "\n3.Search";

cout << "\n4.Sort";

cout << "\n5.Delete";

cout << "\n6.Exit";

cout << "\nEnter your choice: ";

cin >> ch;

// Handling user's menu selection

switch (ch) {

case 1:

insert(); // Insert a new person

break;

case 2:

display(); // Display all persons

break;

case 3:

search(); // Search for a person by id

break;

case 4:

sort(o1.begin(), o1.end(), compare); // Sort persons by id

cout << "\n\nSorted by id: ";

display();

break;

case 5:

dlt(); // Delete a person by id

break;

case 6:

exit(0); // Exit the program

}

} while (ch != 6); // Loop until user chooses to exit

return 0;

}

// Insert a new person into the vector

void insert() {

person p1;

cout << "\nEnter person Name: ";

cin >> p1.name;

cout << "\nEnter person DOB: ";

cin >> p1.DOB;

cout << "\nEnter person mobile number: ";

cin >> p1.pno;

cout << "\nEnter person id: ";

cin >> p1.id;

o1.push\_back(p1); // Add person to vector

}

// Display all persons stored in the vector

void display() {

if (o1.empty()) { // Check if vector is empty

cout << "\nNo records to display!\n";

return;

}

for\_each(o1.begin(), o1.end(), print); // Use for\_each to print each person

}

// Print details of a person

void print(person &p1) {

cout << "\n";

cout << "\nPerson Name: " << p1.name;

cout << "\nPerson DOB: " << p1.DOB;

cout << "\nPerson mobile number: " << p1.pno;

cout << "\nPerson id: " << p1.id;

cout << "\n\n";

}

// Search for a person by id

void search() {

if (o1.empty()) { // Check if vector is empty

cout << "\nNo records available for searching!\n";

return;

}

vector<person>::iterator p;

person p1;

cout << "\nEnter person id to search: ";

cin >> p1.id;

p = find(o1.begin(), o1.end(), p1); // Find person by id

if (p == o1.end()) {

cout << "\nNot found!!!"; // If not found

} else {

cout << "\nFound!!!"; // If found

}

}

// Delete a person by id

void dlt() {

if (o1.empty()) { // Check if vector is empty

cout << "\nNo records to delete!\n";

return;

}

vector<person>::iterator p;

person p1;

cout << "\nEnter person id to delete: ";

cin >> p1.id;

p = find(o1.begin(), o1.end(), p1); // Find person by id

if (p == o1.end()) {

cout << "\nNot found!!!"; // If not found

} else {

o1.erase(p); // Erase person from vector

cout << "\nDeleted!!!";

}

}

***OUTPUT: -***

comp@ubuntu:~/SEC48$ g++ Exp6.cpp

comp@ubuntu:~/SEC48$ ./a.out

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 1

Enter person Name: jeevan

Enter person DOB: 27/07/2005

Enter person mobile number: 7276605175

Enter person id: 48

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 1

Enter person Name: hello

Enter person DOB: 57-8-1947

Enter person mobile number: 678489289276

Enter person id: 12

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 3

Enter person id to search: 12

Found!!!

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 4

Sorted by id:

Person Name: hello

Person DOB: 57-8-1947

Person mobile number: 678489289276

Person id: 12

Person Name: jeevan

Person DOB: 27/07/2005

Person mobile number: 7276605175

Person id: 48

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 5

Enter person id to delete: 48

Deleted!!!

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 2

Person Name: hello

Person DOB: 57-8-1947

Person mobile number: 678489289276

Person id: 12

\* \* \* \* \* Menu \* \* \* \* \*

1.Insert

2.Display

3.Search

4.Sort

5.Delete

6.Exit

Enter your choice: 6

Name: Jeevan Sudhakar Patil

Department: Computer

Class: SE Div: C

Roll no: 48

Subject: Object Oriented Programming

# EXPERIMENT NO. 5

***Code:-***

*/\*Write a function template for selection sort that inputs, sorts and outputs an integer array and a float array.\*/*

#include<iostream>

using namespace std;

int n; // Number of elements

#define size 10 // Max array size

template <typename T> // Template function for Selection Sort

void selection(T A[size])

{

int i, j, min;

T temp;

// Selection sort logic

for(i = 0; i < n-1; i++)

{

min = i;

for(j = i+1; j < n; j++)

{

if(A[j] < A[min])

min = j;

}

temp = A[i]; // Swap current element with minimum

A[i] = A[min];

A[min] = temp;

}

// Output sorted array

cout << "\nSorted array:";

for (i = 0; i < n; i++)

{

cout << " " << A[i];

}

}

int main()

{

int X[size]; // Integer array

float Y[size]; // Float array

int i;

cout << "\n Enter total no of int elements:"; // Input for integer array

cin >> n;

cout << "\n Enter int elements:";

for(i = 0; i < n; i++)

{

cin >> X[i];

}

selection(X); // Sort integers

cout << "\n Enter total no of float elements:"; // Input for float array

cin >> n;

cout << "\n Enter float elements:";

for(i = 0; i < n; i++)

{

cin >> Y[i];

}

selection(Y); // Sort floats

}

OUTPUT:-

comp@ubuntu:~/SEC48$ g++ Exp5.cpp

comp@ubuntu:~/SEC48$ ./a.out

Enter total no of int elements:4

Enter int elements:34

41

12

9

Sorted array: 9 12 34 41

Enter total no of float elements:3

Enter float elements:2.5

2.1

2.9

Sorted array: 2.1 2.5 2.9

Name: Jeevan Sudhakar Patil

Department: Computer

Class: SE Div: C

Roll no: 48

Subject: Object Oriented Programming

# EXPERIMENT NO. 7

***Code:-***

*/\*Write a program in C++ to use map associative container. The keys will be the*

*names of states and the values will be the populations of the states. When the*

*program runs, the user is prompted to type the name of a state. The program then*

*looks in the map, using the state nameas an index and returns the population of*

*the state. \*/*

#include<iostream>

#include<map>

#include<string>

using namespace std;

int main()

{

typedef map<string,int>mapType; // Map declaration

mapType populationMap; // creation of map datamenber as object

// insert records of indian states along with its population

populationMap.insert(pair<string, int>("Maharashtra", 7026357));

populationMap.insert(pair<string, int>("Rajasthan", 6578936));

populationMap.insert(pair<string, int>("Karanataka", 6678993));

populationMap.insert(pair<string, int>("Punjab", 5789032));

populationMap.insert(pair<string, int>("West Bengal", 6676291));

// creation of iterator for seaching entry from Map

mapType::iterator iter;

cout<<"===Population of states in India==\n";

// Return no of entries in map

cout<<"\n Size of populationMap:"<<populationMap.size()<<"\n";

string state\_name;

cout<<"\n Enter name of the state :";

cin>>state\_name;

// seach record

iter = populationMap.find(state\_name);

if( iter!= populationMap.end() )

cout<<state\_name<<"'s population is "

<<iter->second ; // Second mean return a value from map

else

cout<<"Key is not populationMap"<<"\n";

populationMap.clear();

}

**OUTPUT: -**

comp@ubuntu:~/SEC48$ g++ Exp7.cpp

comp@ubuntu:~/SEC48$ ./a.out

===Population of states in India==

Size of populationMap:5

Enter name of the state :Maharashtra

Maharashtra's population is 7026357