|  |  |  |
| --- | --- | --- |
| American University of SharjahCollege of Engineering Department of Computer Science & Engineering  P. O. Box 26666  Sharjah, UAE |  | **Lab Instructor:** Eng. Sameer Alawnah  **Office:** EB2-101  **Phone**: 971-6-5152974  **e-mail**: salawnah@aus.edu  **Semester**: Fall 16 |

**CMP 220L – Introduction to Computer Science II**

**Lab 8**

**Note: The good programmer adds comments to his/her code. Add comments to your code.**

**Question 1:**

You have the following class definition:

#include <iostream>

using namespace std;

class Array { // Class declaration

friend ostream & operator <<(ostream & os, const Array & a);//print all elements in arr to the output stream os

friend istream & operator >>(istream & is, const Array & a);// read all elements of the array from the input stream is.

public:

Array(int = 10); //Initialize the array with 0 values, default size =10

~Array();//Destructor

int getSize();// return the size of the array.

void operator++();//pre increment, increment all elements in the arr

int getAt(int index)

{

if(index<0||index>size)

{

cout<<"Index out of range error!\n";

exit(-1);

}

return arr[index];

}

private:

int size; // size of the created array

int \* arr;

};

void printArray(Array a)

{

for(int i=0;i<a.getSize();i++){

cout<<a.getAt(i)<<" ";

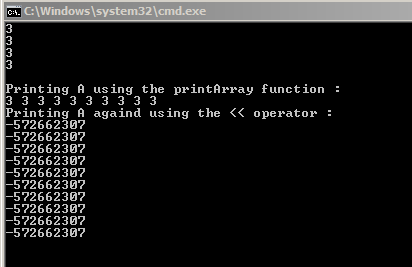
}

cout<<endl;

}

1. Implement all unimplemented functions and run the main() program provided at the end of the question. Will the program run without any problems? what caused the problem?

No, a copy constructor was missing which lead to this problem (printed garbage).



1. Add the copy constructor to the class and run the main() again. Is the problem resolved? why ?

Yes, because I’ve added the copy constructor.

int main()

{

Array a(5), b(5);

cout << "Please enter the content of the first array (A):\n";

cin >> a;

cout << "Please enter the content of the second array (B):\n";

cin >> b;

cout<<"Printing A and B using the << operator :\n ";

cout << "A = " << a << " B = " << b << endl;

++a;

cout << "++A\n";

cout << "A after ++A = " << a << endl;

cout<<"Printing A using the printArray function : \n";

printArray(a);

cout<<"Printing A againd using the << operator :\n";

cout<<a<<endl;

return 0;

}

#include <iostream>

using namespace std;

class Array { // Class declaration

friend ostream & operator <<(ostream & os, const Array & a);//print all elements in arr to the output stream os

friend istream & operator >>(istream & is, const Array & a);// read all elements of the array from the input stream is.

public:

Array(int i = 10)//Initialize the array with 0 values, default size =10

{

size = i;

arr = new int[size];

for (int i = 0; i < size; i++)

{

arr[i]=0;

}

}

Array(Array &x)

{

size = x.size;

arr = new int[size];

for (int i = 0; i < size; i++)

{

arr[i] =x.arr[i] ;

}

}

~Array()//Destructor

{

delete[]arr;

arr = NULL;

}

int getSize()// return the size of the array.

{

return (size);

}

void operator++()//pre increment, increment all elements in the arr

{

for (int i = 0; i < size; i++)

{

arr[i] = ++arr[i];

}

}

int getAt(int index)

{

if (index<0 || index>size)

{

cout << "Index out of range error!\n";

exit(-1);

}

return arr[index];

}

private:

int size; // size of the created array

int \* arr;

};

void printArray(Array a)

{

for (int i = 0; i<a.getSize(); i++){

cout << a.getAt(i) << " ";

}

cout << endl;

}

ostream & operator <<(ostream & os, const Array & a)

{

for (int i = 0; i < a.size; i++)

{

os << a.arr[i]<<endl;

}

return (os);

}

istream & operator >>(istream & is, const Array & a)

{

for (int i = 0; i < a.size; i++)

{

is >> a.arr[i];

}

return (is);

}

int main()

{

Array a(5), b(5);

cout << "Please enter the content of the first array (A):\n";

cin >> a;

cout << "Please enter the content of the second array (B):\n";

cin >> b;

cout << "Printing A and B using the << operator :\n ";

cout << "A = " << a << " B = " << b << endl;

++a;

cout << "++A\n";

cout << "A after ++A = " << a << endl;

cout << "Printing A using the printArray function : \n";

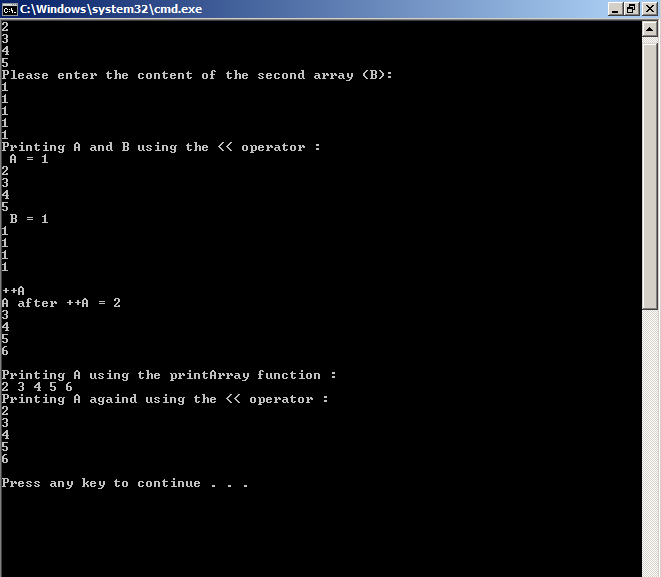
printArray(a);

cout << "Printing A againd using the << operator :\n";

cout << a << endl;

return 0;

}



**Question 2:**

In the previous lab do the following modification to the lab solution:

1. Add the default constructor to the Student class, the constructor should initialize the student id to “-1”.
2. Add the following function to the StudentsList class :

Student findStudentById(string id);

This function should return the found student or student initialized with the default constructor (id is “-1”).

1. In the main generate the following report and print it to the user:

Program\_1\_name

Student\_1\_ID Student\_1\_FirstName Student\_1\_LastName

. . . .

Student\_m\_ID Student\_m\_FirstName Student\_m\_LastName

. . . .

. . . .

Program\_n\_name

Student\_1\_ID Student\_1\_FirstName Student\_1\_LastName

. . . .

Student\_m\_ID Student\_m\_FirstName Stdent\_m\_LastName

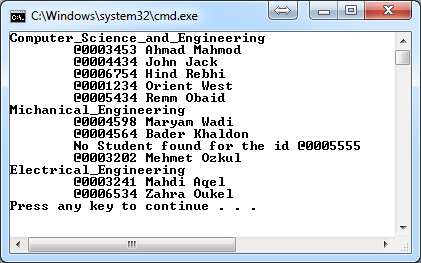
If the ID for the ID is not found in the student list, you should print “Student are not found” instead

of the student information.

Notes:

* You have to use the findStudentByID function and check the returned student id before printing.
* You have to use the attached “programs.txt” file instead of the one attached with the lab7.

Example Run:



#include<iostream>

#include<string>

#include<vector>

#include<fstream>

using namespace std;

class Course{

private:

string id;

string name;

public:

string getID(){ return id; }

string getName(){ return name; }

void setID(string idd){ id = idd; }

void setName(string namee){ name = namee; }

friend istream& operator>>(istream &in, Course &c);

friend ostream& operator<<(ostream &out, Course c);

};

istream& operator>>(istream &in, Course &c)

{

in >> c.id;

getline(in, c.name);

return in;

}

ostream& operator<<(ostream &out, Course c)

{

out << c.id << "," << c.name;

return out;

}

class Student{

private:

string id;

string fname;

string lname;

vector<string> course\_ID\_list;

public:

Student(){ id = "-1"; }

string getID(){ return id; }

string getFNAme(){ return fname; }

string getLNAme(){ return lname; }

void setID(string idd){ id = idd; }

void setFName(string namee){ fname = namee; }

void setLName(string namee){ lname = namee; }

string getCourseIDAt(int index)

{

if (index<0 || index>course\_ID\_list.size() - 1)

{

cout << "Index out of range";

exit(1);

}

return course\_ID\_list[index];

}

int getCoursesCount()

{

return course\_ID\_list.size();

}

friend istream& operator>>(istream &in, Student &std);

friend ostream& operator<<(ostream &out, Student std);

};

istream& operator>>(istream &in, Student &std)

{

in >> std.id;

in >> std.fname;

in >> std.lname;

int size;

in >> size;

std.course\_ID\_list.clear();

for (int i = 0; i < size; i++)

{

string t;

in >> t;

std.course\_ID\_list.push\_back(t);

}

return in;

}

ostream& operator<<(ostream &out, Student std)

{

out << std.id << "," << std.fname << "," << std.lname << "," << std.course\_ID\_list.size();

for (int i = 0; i < std.course\_ID\_list.size(); i++)

{

out << "," << std.course\_ID\_list[i];

}

return out;

}

class Program{

private:

string id;

string name;

vector<string> student\_ID\_list;

public:

string getID(){ return id; }

string getName(){ return name; }

void setID(string idd){ id = idd; }

void setName(string namee){ name = namee; }

string getStudentIDAt(int index)

{

if (index<0 || index>student\_ID\_list.size() - 1)

{

cout << "Index out of range";

exit(1);

}

return student\_ID\_list[index];

}

int getStudetnsCount()

{

return student\_ID\_list.size();

}

friend istream& operator>>(istream &in, Program &prog);

friend ostream& operator<<(ostream &out, Program prog);

};

istream& operator>>(istream &in, Program &prog)

{

in >> prog.id;

in >> prog.name;

int s;

in >> s;

prog.student\_ID\_list.clear();

for (int i = 0; i < s; i++)

{

string t;

in >> t;

prog.student\_ID\_list.push\_back(t);

}

return in;

}

ostream& operator<<(ostream &out, Program prog)

{

out << prog.id << "," << prog.name << "," << prog.student\_ID\_list.size();

for (int i = 0; i < prog.student\_ID\_list.size(); i++)

{

out << "," << prog.student\_ID\_list[i];

}

return out;

}

class Courses\_list{

private:

vector<Course> list;

public:

int count(){ return list.size(); }

Course getCourseAt(int index)

{

if (index<0 || index>list.size() - 1)

{

cout << "Index out of range";

exit(1);

}

return list[index];

}

friend istream& operator>>(istream &in, Courses\_list &clist);

friend ostream& operator<<(ostream &out, Courses\_list clist);

};

istream& operator>>(istream &in, Courses\_list &clist)

{

int s;

in >> s;

clist.list.clear();

for (int i = 0; i < s; i++)

{

Course c;

in >> c;

clist.list.push\_back(c);

}

return in;

}

ostream& operator<<(ostream &out, Courses\_list clist)

{

out << clist.list.size();

for (int i = 0; i < clist.list.size(); i++)

{

out << ",";

out << clist.list[i];

}

return out;

}

class Students\_list {

private:

vector<Student> list;

public:

Student findStudentById(string id)

{

Student s;

for (int i = 0; i < list.size(); i++)

{

if (id == list[i].getID())

{

return(list[i]);

}

}

return (s);

}

int count(){ return list.size(); }

Student getStudentAt(int index)

{

if (index<0 || index>list.size() - 1)

{

cout << "Index out of range";

exit(1);

}

return list[index];

}

friend istream& operator>>(istream &in, Students\_list &slist);

friend ostream& operator<<(ostream &out, Students\_list slist);

};

istream& operator>>(istream &in, Students\_list &slist)

{

int s;

in >> s;

slist.list.clear();

for (int i = 0; i < s; i++)

{

Student c;

in >> c;

slist.list.push\_back(c);

}

return in;

}

ostream& operator<<(ostream &out, Students\_list slist)

{

out << slist.list.size();

for (int i = 0; i < slist.list.size(); i++)

{

out << ",";

out << slist.list[i];

}

return out;

}

class Programs\_list {

private:

vector<Program> list;

public:

int count(){ return list.size(); }

Program getProgramAt(int index)

{

if (index<0 || index>list.size() - 1)

{

cout << "Index out of range";

exit(1);

}

return list[index];

}

friend istream& operator>>(istream &in, Programs\_list &plist);

friend ostream& operator<<(ostream &out, Programs\_list plist);

};

istream& operator>>(istream &in, Programs\_list &plist)

{

int s;

in >> s;

plist.list.clear();

for (int i = 0; i < s; i++)

{

Program c;

in >> c;

plist.list.push\_back(c);

}

return in;

}

ostream& operator<<(ostream &out, Programs\_list plist)

{

out << plist.list.size();

for (int i = 0; i < plist.list.size(); i++)

{

out << "," << plist.list[i];

}

return out;

}

int main()

{

ifstream stds("students.txt");

if (stds.fail())

{

cout << "Can't open studnets file";

exit(1);

}

ifstream crs("ofcoureses.txt");

if (crs.fail())

{

cout << "Can't open courses file";

exit(2);

}

ifstream progs("programs.txt");

if (progs.fail())

{

cout << "Can't open programs file";

exit(3);

}

Student std;

Students\_list st\_list;

stds >> st\_list;

Courses\_list cr\_list;

crs >> cr\_list;

Programs\_list prgs;

progs >> prgs;

for (int i = 0; i < prgs.count(); i++)

{

Program p = prgs.getProgramAt(i);

cout << p.getID()<<" "<<p.getName() << endl;

for (int j = 0; j < p.getStudetnsCount(); j++)

{

std = st\_list.findStudentById(p.getStudentIDAt(j));

if (std.getID() == "-1"){ cout << "Student is not found." << endl; }

else{

cout << std.getID()<<" " << std.getFNAme() <<" "<< std.getLNAme() << endl;

}

}

}

stds.close();

crs.close();

progs.close();

ofstream stds\_out("students\_out.txt");

if (stds\_out.fail())

{

cout << "Can't open students output file";

exit(4);

}

ofstream crs\_out("ofcoureses\_out.txt");

if (crs\_out.fail())

{

cout << "Can't open courses output file";

exit(4);

}

ofstream progs\_out("programs\_out.txt");

if (progs\_out.fail())

{

cout << "Can't open programs output file";

exit(5);

}

stds\_out << st\_list;

crs\_out << cr\_list;

progs\_out << prgs;

stds\_out.flush();

crs\_out.flush();

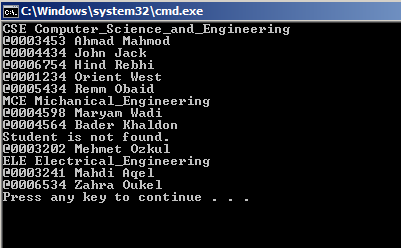
progs\_out.flush();

stds\_out.close();

crs\_out.close();

progs\_out.close();

}



Good Luck ☺