Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №8**

По дисциплине «Проектирование программ в ИС»

**“Потоки ввода-вывода”**

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Парфеевец И.А.

**Проверил:**

Монтик Н.С.

Брест 2022

ЦЕЛЬ РАБОТЫ

Знакомство с объектно-ориентированной библиотекой ввода-вывода C++ (иерархия классов iostream).

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с объектно-ориентированной библиотекой ввода-вывода C++. Изучить классы, иерархии классов, методы.

2. Выполнить примеры с использованием методов get(), функции getline().

3. Выполнить примеры с использованием метода put().

4. Выполнить примеры с использованием методов организации файлового ввода-вывода.

5. Выполнить примеры с использованием с использованием манипуляторов.

6. Реализовать сохранение и загрузку данных (массива объектов) из файла для индивидуальных заданий.

Код Программы

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <stdlib.h>

#include <ctime>

#include<string>

#include<Windows.h>

#include <thread>

#define DEBUG

using namespace std;

string path1 = "Transaction.txt";

string path2 = "Student.txt",pathg2="Studentinfo.txt";

string path3 = "Employee.txt", path3obj="EmployeeOBJ.txt",pathg3="EmployeeList.txt";

//done

class Transaction {

private :

static int operationid;

string senderacc;

string recipientacc;

string sendername;

string recipientname;

int sum;

friend void clear(int);

friend bool exists\_test0(string& name);

friend void FileContent(string path);

public:

Transaction() {

senderacc = "you did not specify a sender account";

recipientacc = "you did not specify a recipient account ";

sendername = "you haven't specify sender name";

recipientname = "you haven't specify recipient name";

sum = 0;

operationid++;

}

Transaction(Transaction& sent) {

this->recipientacc = sent.recipientacc;

this->recipientname = sent.recipientacc;

this->senderacc = sent.senderacc;

this->sendername = sent.sendername;

this->sum = sent.sum;

operationid++;

}

Transaction(string senderacc, string recipientacc, string sendername, string recipientname, int sum) {

this->recipientacc = recipientacc;

this->recipientname = recipientname;

this->senderacc = senderacc;

this->sendername = sendername;

this->sum = sum;

operationid++;

}

void To\_Show() {

cout << "Information about operation No " << operationid << ": \n";

cout << "recipient name : " << recipientname << endl;

cout << "sender name : " << sendername << endl;

cout << "recipient account number : " << recipientacc << endl;

cout << "sender account number : " << senderacc << endl;

cout << "sum : " << sum << endl;

}

void To\_Save() {

fstream FS;

FS.open("Transaction.txt", ios::app);

if (!FS.is\_open()) { cout << "File wasn't opened \n"; }

else {

FS<< "Information about operation No " << operationid << ": \n";

FS << "recipient name : " << recipientname << endl;

FS << "sender name : " << sendername << endl;

FS << "recipient account number : " << recipientacc << endl;

FS << "sender account number : " << senderacc << endl;

FS << "sum : " << sum << endl<<endl;

cout << "\nYour data has been saved " << endl;

}

FS.close();

}

};

//done

class Student {

private:

string name;

string age;

string Qualification;

char\* TS;

static int studentid;

friend void clear(int);

friend bool exists\_test0(string& name);

friend void FileContent(string path);

public:

Student() {

name = "You haven't specify name ";

Qualification = "You haven't specify Qualification ";

age ="age wasn't specify";

studentid++;

}

Student(char\* s) {

TS = s;

int i = 0;

while (s[i] != ',') {

name += s[i];

i++;

}

i++;

while (s[i] != ',') {

Qualification += s[i];

i++;

}

i++;

for (i; i < 40; i++) {

if (s[i] == '.') { break; }

age += TS[i];

}

studentid++;

}

Student(Student& Sample) {

this->age = Sample.age;

this->name = Sample.name;

this->Qualification = Sample.Qualification;

studentid++;

}

void TO\_Show() {

cout << "There is info about student No " << studentid<<":" << endl;

cout << "Name: " << name << endl;

cout << "Qualification : " << Qualification << endl;

cout << "age : " << age << endl<<endl;

}

int To\_Get() {

cout << endl;

return puts(TS);

}

void save() {

fstream FS;

FS.open("Student.txt",ios::app );

!FS.is\_open() ? cout << "file wasn't opened ! :( \n" : cout << "File is opened ! \n";

FS << "There is info about student No " << studentid << ":" << endl;

FS << "Name: " << name << endl;

FS << "Qualification : " << Qualification << endl;

FS << "age : " << age << endl << endl;

FS.close();

}

};

//done

class Employee {

public:

friend string getname();

friend void clear(int);

friend bool exists\_test0(string& name);

friend void FileContent(string path);

string name;

string age;

string speciality;

static int Employeeid;

char\* GL;

public:

Employee(Employee& Sample) {

this->name = Sample.name;

this->age = Sample.age;

this->speciality = Sample.speciality;

Employeeid++;

}

Employee() {

name = "name wasn't specify";

age = "0";

Employeeid++;

speciality = "speciality wasn't specify";

}

Employee(string name, string age, string speciality) {

this->name = name;

this->speciality = speciality;

this->age = age;

Employeeid++;

}

Employee(char\* sample) {

GL= sample;

int i = 0;

while (GL[i] != ',') {

name += GL[i];

i++;

}

i++;

while (GL[i] != ',') {

age += GL[i];

i++;

}

i++;

for (i; i < 50; i++) {

if (GL[i] == '.') { break; }

age += GL[i];

}

Employeeid++;

}

void To\_Get() { puts(GL); }

void To\_Show() {

cout << "There is info about id " << Employeeid << ":" << endl;

cout << "Name: " << name << endl;

cout << "Speciality : " << speciality << endl;

cout << "age : " << age << endl << endl;

}

void save() {

fstream FS;

FS.open("Employee.txt", ios::app);

!FS.is\_open() ? cout << "file wasn't opened ! :( \n" : cout << "File is opened ! \n";

FS << "There is info about student No " << Employeeid << ":" << endl;

FS << "Name: " << name << endl;

FS << "Qualification : " << speciality << endl;

FS << "age : " << age << endl << endl;

FS.close();

}

void objectsave(Employee& sample) {

ofstream FS;

FS.open("EmployeeOBJ.txt", ofstream::app);

FS.write((char\*)&sample, sizeof(Employee));

}

void objectshow() {

ifstream getobj;

getobj.open(path3obj);

Employee sample;

while (getobj.read((char\*)&sample, sizeof(Employee))) {

sample.To\_Show();

}

}

};

string getname() {

srand(time(NULL));

string names[19] = { "Anna","Dmitriy","Zhenya","Andrey","Nick","John","Ivan","Vanya","Danila",

"Denis","Denchik","Vova","Garik","Anton","Masha","Katya","Nika","Zhenya","Nikita"};

string surnames[19] = { "Kovalev","Yarmolenko","Syhoi","Dolgiy","Puchincky",

"Karagodin","Dubina","Yasuk","Novy","Star","Delphy","Lacarte","Hudick","Romanko","Kirilovich","Shlyk",

"Majort","Dandy","PUcher"};

int gN = rand() % 18 + 0;

int gS = rand() % 18 + 0;

string name = names[gN] + " " + surnames[gS];

return name;

};

string getage() {

srand(time(NULL));

int age = rand() % 40 + 18;

return to\_string(age);

}

string getspeciality() {

srand(time(NULL));

string specialities[7] = {"Taxi Man","Producer","Toiler Cleaner","Mentor",

"Manager","Actor","Clown"};

int get = rand() % 6 + 0;

return specialities[get];

}

bool exists\_test0(string& name) {

ifstream f(name.c\_str());

f.close();

return f.good();

}

void clear(int num) {

if (num == 1) {

if (!remove("Transaction.txt")) {

cout << "the file has been cleared\n";

}

else cout << "file wasn't cleared ";

}

if (num == 2) {

if (!remove("Student.txt")) {

cout << "the file has been cleared\n";

}

else cout << "file wasn't cleared ";

}

if (num == 3) {

if (!remove("Employee.txt")) {

cout << "the file has been cleared\n";

}

else cout << "file wasn't cleared ";

}

if (num == 4) {

if (!remove("EmployeeOBJ.txt")) {

cout << "the file has been cleared\n";

}

else cout << "file wasn't cleared ";

}

};

void FileContent(string path) {

fstream FC;

string buff;

FC.open(path);

if (!FC.is\_open()) {

cout << "file wasn't opened " << endl;

}

else {

while (!FC.eof()) {

buff = "";

FC>>buff;

cout << buff << endl;

}

}

FC.close();

}

void ThTask() {

for (int i = 0; i < 100; i++) {

Employee Andre;

Andre.name = getname();

Andre.age = getage();

Andre.speciality = getspeciality();

#ifdef DEBUG

Andre.To\_Show();

#endif // DEBUG

Andre.save();

Sleep(1000);

}}

int Transaction::operationid = 0;

int Student::studentid = 0;

int Employee::Employeeid = 0;

int main() {

// ПЕРВЫЙ КЛАСС

cout << "|||||||||||| The first class(Transaction) :|||||||||||||||| " << endl<<endl;

Transaction T1("4305495893", "32030650560", "Ivan", "Andrew", 50);

Transaction T2("5495208954", "054869024035", "Masha", "Dasha", 100);

T1.To\_Show();

T1.To\_Save();

T2.To\_Show();

T2.To\_Save();

FileContent(path1);

#ifdef DEBUG

clear(1);

#endif // DEBUG

// ВТОРОЙ КЛАСС

cout << "\n\n|||||||||||The second class(Student): |||||||||||||\n\n ";

ifstream SS;

SS.open(pathg2);

while (!SS.eof()) {

char StudentArr[40];

SS.getline(StudentArr, 40);;

Student Template(StudentArr);

Template.TO\_Show();

Template.To\_Get();

Template.save();

}

SS.close();

#ifdef DEBUG

clear(2);

#endif // DEBUG

//ТРЕТИЙ КЛАСС

cout << "\n\n||||||||||The third class(Employee): ||||||||||||| \n\n ";

ifstream EFS;

EFS.open(pathg3);

while (!EFS.eof()) {

char getinfo[50];

EFS.getline(getinfo, 50);

Employee Sample(getinfo);

Sample.To\_Show();

Sample.To\_Get();

Sample.save();

}

//запись и вывод в файл объекта класса

Employee Andrusha("Andre", "20", "Taxi man");

Andrusha.objectsave(Andrusha);

Andrusha.objectshow();

cout << "let's create Employees by using our get functions: \n";

thread th1(ThTask);

th1.join();

#ifdef DEBUG

clear(3);

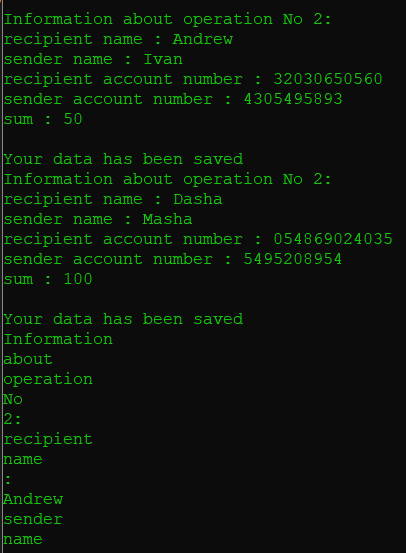
clear(4);

#endif // DEBUG

return 713;

}

Результат работы программы :



Вывод : научился работать с классами