Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра экономической информатики

Дисциплина: Объектно-ориентированное программирование

Отчет о лабораторной работе №3

«ПОТОКИ ВВОДА/ВЫВОДА В ЯЗЫКЕ С++»

Выполнил: ст.гр. 872303

Околотович А.Г.

Проверил: Богданова Е.А.

Минск 2020

ЦЕЛЬ РАБОТЫ: изучить понятие потока; организацию ввода из потока и вывода в поток; контроль состояния потока и исправление ошибок; неформатированный ввод-вывод; манипуляторы потоков (стандартные и определяемые пользователем).

ЗАДАНИЕ: На основе разработанной иерархии классов, реализованной во второй лабораторной работе, реализовать программу работы с данными, используя потоки ввода-вывода. Реализовать функций добавления, удаления,

редактирования и просмотра данных, поиска данных по необходимым параметрам.

В данной лабораторной работе реализованы функции добавления, просмотр, редактирование, и удаление данных, поиск по необходимому параметру. (Рисунок 1)

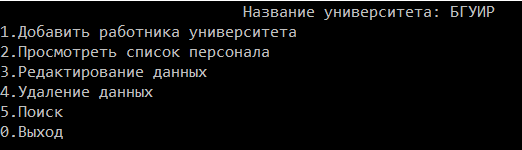


Рисунок 1 – Основные функции программы

На рисунке 2 происходит добавление персонала в университет.

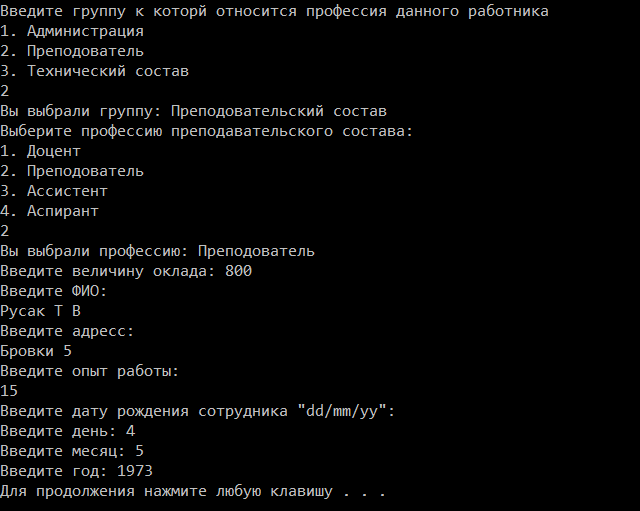


Рисунок 2 – Добавление информации

На рисунке 3 реализован вывод информации.

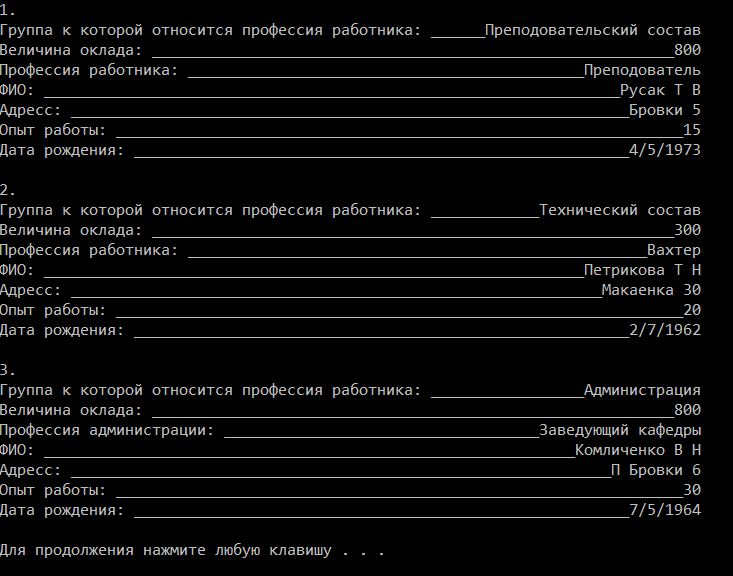


Рисунок 3 – Вывод информации

На рисунке 4 происходит редактирование информации.

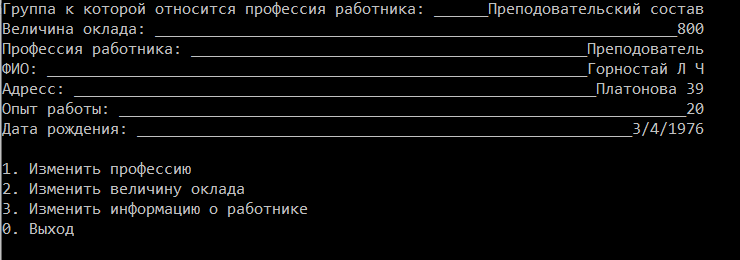


Рисунок 4 – Редактирование информации

На рисунке 5 и 6 происходит поиск информации о сотруднике по величине оклада.

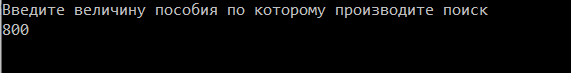


Рисунок 5 – Ввод величины пособия, по которому производится поиск

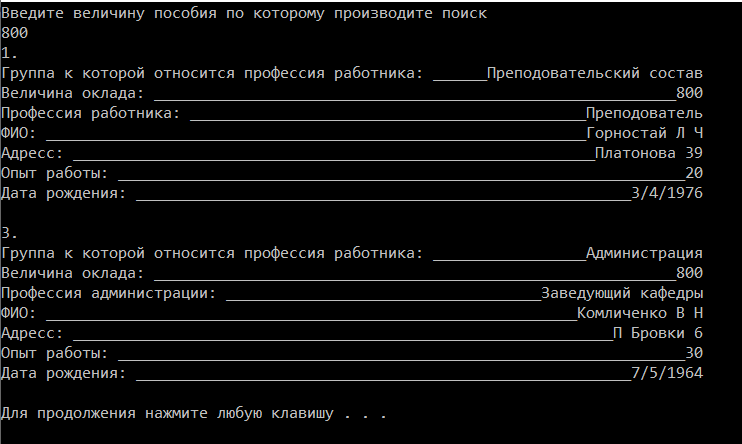


Рисунок 6 – Вывод информации

На рисунке 7 и 8 происходит удаление определенного сотрудника.

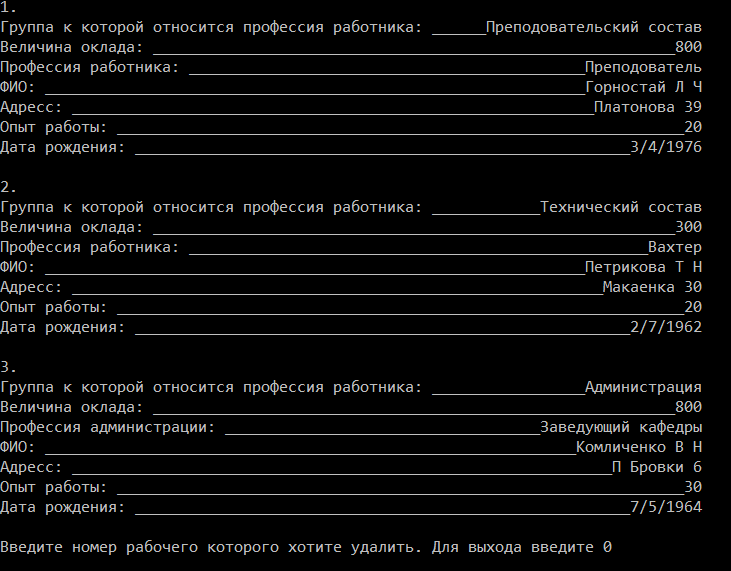


Рисунок 7 – Выбор сотрудника для удаления

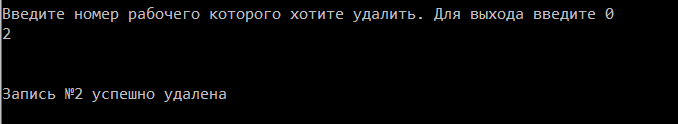


Рисунок 8 - Удаление сотрудника

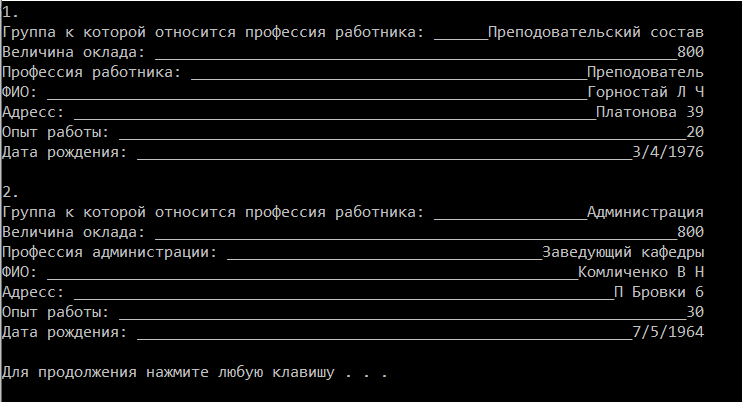


Рисунок 9 – Вывод данных

Листинг кода

Header.h

#pragma once

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class University {

private:

string name;

public:

University() {};

University(int n) {

if (n) throw 1;

}

~University() {};

void get\_name();

void show\_information();

};

class Information {

public:

virtual void show() = 0; //Вывести информацию

virtual void change() = 0; //Редактированние данных

virtual bool search(float) = 0; //Поиск данных

Information() {};

};

class Staff : public Information{

private:

enum ptype { administration, lectuter, technicalStaff};

ptype temp;

float posobie;

public:

Staff\* st;

Staff() {};

void get\_staff();

float get\_posobie();

void change(); //Редактирование данных

void show(); //Вывести информацию

bool search(float); //Поиск данных

ptype get\_st() { return temp; };

class Administration {

private:

enum atype { rector, viceRector, dean, deputyDean, headOfDepartment };

atype temp;

public:

void get\_administration();

void show\_admin();

atype get\_admin() { return temp; };

};

Administration\* admin= new Administration;

class Lectuter {

private:

enum ltype { associateProfessor, professor, assistant, postgraduate };

ltype temp;

public:

void get\_lectuter();

void show\_lectuter();

ltype get\_lec() { return temp; };

};

Lectuter\* lec=new Lectuter;

class TechnicalStaff {

private:

enum ttype { laboratoryAssistant, watchman, cleaner };

ttype temp;

public:

void get\_technicalstaff();

void show\_technicalstaff();

ttype get\_tstaff() { return temp; };

};

TechnicalStaff\* tstaff= new TechnicalStaff;

class Employee : public Administration, Lectuter,TechnicalStaff {

private:

string fio;

string address;

int workExperience;

public:

Employee() {};

void get\_employee();

void show\_employee();

class Date

{

private:

int day, mons, year;

public:

Date() {};

void get\_date()

{

cout << "Введите дату рождения сотрудника \"dd/mm/yy\":" << endl;

while (true) {

cout << "Введите день: ";

cin.setf(ios::skipws);

cin >> day;

if (day > 31 || day < 1) {

cout<< "День должен быть между 1 и 31\n ";

cin.clear(ios::failbit);//установить флаг ошибки

}

if (cin.good()) {

cin.ignore(10, '\n');

break; // Ввод корректный, выйти из 'while'

} cin.clear(); // ошибка ввода; очистить статус ошибки

cin.ignore(10, '\n');

cout << "Неверно введен день \n";

cin.unsetf(ios::skipws);

}

while (true) {

cout << "Введите месяц: ";

cin.setf(ios::skipws);

cin >> mons;

if (mons > 12 || mons < 1) {

cout << "Месяц должен быть между 1 и 12\n ";

cin.clear(ios::failbit);//установить флаг ошибки

}

if (cin.good()) {

cin.ignore(10, '\n');

break; // Ввод корректный, выйти из 'while'

} cin.clear(); // ошибка ввода; очистить статус ошибки

cin.ignore(10, '\n');

cout << "Неверно введен месяц \n";

cin.unsetf(ios::skipws);

}

while (true) {

cout << "Введите год: ";

cin.setf(ios::skipws);// не пропускать разделители

cin >> year;

if (year > 2002 || year < 1940) {

//cout << "Год должен быть между 1940 и 2002\n ";

cin.clear(ios::failbit);//установить флаг ошибки

}

if (cin.good()) {

cin.ignore(10, '\n');

break; // Ввод корректный, выйти из 'while'

} cin.clear(); // ошибка ввода; очистить статус ошибки

cin.ignore(10, '\n');

cout << "Неверно введен год \n";

cin.unsetf(ios::skipws);

}

}

void show\_date()const {

cout <<std::right << std::setw(56) << day << "/" << mons << "/" << year << endl;

}

};

Date D1;

};

Employee\* employee=new Employee;

};

template <class T>

T Check()

{

T i;

try {

while (!(cin >> i) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

throw "Ошибка ввода. Повторите попытку";

}

return i;

}

catch (const char\* str)

{

cout << str << endl;

}

}

class NegativeNumberExeption : public std::exception {

public:

string m\_error;

NegativeNumberExeption(string error) :m\_error(error) {}

const char\* what() const noexcept { return m\_error.c\_str(); }

~NegativeNumberExeption() {};

};

Source.cpp

#include "Header.h"

string proverka() {

string str;

while (1) {

try {

rewind(stdin);

cin >> str;

for (char c : str) {

if (c >= '0' && c <= '9') cin.clear(ios::failbit);

}

if (cin.good()) {

cin.clear();

return str;

}

else {

cin.clear();

throw 1;

}

}

catch (int) {

cout << "Введено число. Повторите попытку" << endl;

}

}

}

void University::get\_name(){

cout << "Введите название универистета: " << endl;

name = proverka();

}

void University::show\_information() {

cout.setf(ios::right);

cout.width(50);

cout << "Название университета: " << name<<endl;

cout.unsetf(ios::right);

}

void Staff::get\_staff() {

int choice;

cout << "Введите группу к которй относится профессия данного работника " << endl

<<"1. Администрация "<<endl

<<"2. Преподователь "<<endl

<<"3. Технический состав "<<endl;

choice = Check<int>();

while (choice < 0 || choice > 3) {

choice = Check<int>();

};

switch (choice)

{

case 1: temp = administration; break;

case 2: temp = lectuter; break;

case 3: temp = technicalStaff; break;

}

cout << "Вы выбрали группу: ";

switch (temp)

{

case 0: cout<< "Администрация" << endl; break;

case 1: cout << "Преподовательский состав" << endl; break;

case 2: cout << "Технический состав" << endl; break;

}

}

float Staff::get\_posobie() {

cout << "Введите величину оклада: ";

while (!(cin >> posobie))

{

cin.clear();

while (cin.get() != '\n');

cout << "Неверный ввод. " << endl;

cout << "Введите число: ";

}

return posobie;

}

void Staff::show() {

cout << "Группа к которой относится профессия работника: ";

cout.fill('\_');

switch (temp)

{

cout.fill('\_');

case 0: cout << std::right << std::setw(30) << "Администрация" << endl; break;

case 1: cout << std::right << std::setw(30) << "Преподовательский состав" << endl; break;

case 2: cout << std::right << std::setw(30) << "Технический состав" << endl; break;

}

cout << "Величина оклада: " << std::right << std::setw(61)<<std::setprecision(4) << posobie << endl;

if (get\_st() == 0) {

admin->show\_admin();

}

if (get\_st() == 1) {

lec->show\_lectuter();

}

if (get\_st() == 2) {

tstaff->show\_technicalstaff();

}

employee->show\_employee();

}

void Staff::change() {

while (true)

{

int choice;

system("cls");

this->show();

cout << "1. Изменить профессию\n2. Изменить величину оклада\n3. Изменить информацию о работнике\n0. Выход\n";

choice = Check<int>();

while (choice < 0 || choice > 3) {

cout << "Ошибка ввода. Повторите попытку\n";

choice = Check<int>();

};

system("cls");

switch (choice)

{

case 1:

{

cout << "Выберите новую профессию"<<endl;

get\_staff();

if (get\_st() == 0) {

admin->get\_administration();

}

if (get\_st() == 1) {

lec->get\_lectuter();

}

if (get\_st() == 2) {

tstaff->get\_technicalstaff();

}

}; break;

case 2:

{

cout << "Введите новую величину оклада"<<endl;

get\_posobie();

}; break;

case 3:

{

cout << "Введите новые данные о работнике"<<endl;

employee->get\_employee();

}; break;

case 0:

{

return;

}; break;

default:

{

cout << "Неверный ввод";

}; break;

}

system("cls");

}

}

bool Staff::search(float searchpos) {

if (searchpos == posobie) {

return true;

}

}

void Staff::Administration::get\_administration() {

int choice;;

cout << "Выберите профессию работника " << endl

<<"1. Ректор "<<endl

<<"2. Проректор "<<endl

<<"3. Декан "<<endl

<<"4. Зам декана "<<endl

<<"5. Заведующий кафедры "<<endl;

choice = Check<int>();

while (choice < 0 || choice > 5) {

cout << "Ошибка ввода. Повторите попытку\n";

choice = Check<int>();

};

switch (choice)

{

case 1: temp = rector; break;

case 2: temp = viceRector; break;

case 3: temp = dean; break;

case 4: temp = deputyDean; break;

case 5: temp = headOfDepartment; break;

}

cout << "Вы выбрали профессию: ";

switch (temp)

{

case 0: cout << "Ректор" << endl; break;

case 1: cout << "Проректор" << endl; break;

case 2: cout << "Декан" << endl; break;

case 3: cout << "Зам декана" << endl; break;

case 4: cout << "Заведующий кафедры" << endl; break;

}

}

void Staff::Administration::show\_admin() {

cout << "Профессия администрации: ";

switch (temp)

{

case 0: cout << std::right << std::setw(53) << "Ректор" << endl; break;

case 1: cout << std::right << std::setw(53) << "Проректор" << endl; break;

case 2: cout << std::right << std::setw(53) << "Декан" << endl; break;

case 3: cout << std::right << std::setw(53) << "Зам декана" << endl; break;

case 4: cout << std::right << std::setw(53) << "Заведующий кафедры" << endl; break;

}

}

void Staff::Lectuter::get\_lectuter() {

int choice;

cout << "Выберите профессию преподавательского состава: " << endl

<< "1. Доцент " << endl

<< "2. Преподователь " << endl

<< "3. Ассистент " << endl

<< "4. Аспирант " << endl;

choice = Check<int>();

while (choice < 0 || choice > 4) {

cout << "Ошибка ввода. Повторите попытку\n";

choice = Check<int>();

};

switch (choice)

{

case 1: temp = associateProfessor; break;

case 2: temp = professor; break;

case 3: temp = assistant; break;

case 4: temp = postgraduate; break;

}

cout << "Вы выбрали профессию: ";

switch (temp)

{

case 0: cout << "Доцент" << endl; break;

case 1: cout << "Преподователь" << endl; break;

case 2: cout << "Ассистент" << endl; break;

case 3: cout << "Аспирант" << endl; break;

}

}

void Staff::Lectuter::show\_lectuter() {

cout << "Профессия работника: ";

switch (temp)

{

case 0: cout << std::right << std::setw(57) << "Доцент" << endl; break;

case 1: cout << std::right << std::setw(57) << "Преподователь" << endl; break;

case 2: cout << std::right << std::setw(57) << "Ассистент" << endl; break;

case 3: cout << std::right << std::setw(57) << "Аспирант" << endl; break;

}

}

void Staff::TechnicalStaff::get\_technicalstaff() {

int choice;

cout << "Введите профессию технического персонала: " << endl

<< "1. Лаборант " << endl

<< "2. Вахтер " << endl

<< "3. Уборщица " << endl;

choice = Check<int>();

while (choice < 0 || choice > 3) {

cout << "Ошибка ввода. Повторите попытку\n";

choice = Check<int>();

};

switch (choice)

{

case 1: temp = laboratoryAssistant; break;

case 2: temp = watchman; break;

case 3: temp = cleaner; break;

}

cout << "Вы выбрали профессию: ";

switch (temp)

{

case 0: cout << "Лаборант" << endl; break;

case 1: cout << "Вахтер" << endl; break;

case 2: cout << "Уборщица" << endl; break;

}

}

void Staff::TechnicalStaff::show\_technicalstaff () {

cout << "Профессия работника: ";

switch (temp)

{

case 0: cout << std::right << std::setw(57) << "Лаборант" << endl; break;

case 1: cout << std::right << std::setw(57) << "Вахтер" << endl; break;

case 2: cout << std::right << std::setw(57) << "Уборщица" << endl; break;

}

}

void Staff::Employee::get\_employee() {

cout << "Введите ФИО: " << endl;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

getline(cin, fio);

cout << "Введите адресс: " << endl;

getline(cin, address);

cout << "Введите опыт работы: " << endl;

while (!(cin >> workExperience))

{

cin.clear();

while (cin.get() != '\n');

cout << "Неверный ввод. " << endl;

cout << "Введите число: ";

}

D1.get\_date();

}

void Staff::Employee::show\_employee() {

cout << "ФИО: " << std::right << std::setw(73) << fio << endl

<< "Адресс: " << std::right << std::setw(70) << address << endl

<< "Опыт работы: " << std::right << std::setw(65) << workExperience << endl

<< "Дата рождения: "; D1.show\_date();

cout << endl;

cout.fill(' ');

}

Main.cpp

#include <vector>

#include <windows.h>

#include "Header.h"

void exx() {

cout << "Программа завершилась из-за ошибки";

system("pause");

exit(1);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "rus");

vector<Information\*> inf;

University university;

try {

University\* university = new University;

}

catch (bad\_alloc exept) {

cout << "Возникло исключение " << exept.what() << endl;}

catch (int) { cout << "int exception hendler" << endl; }

university.get\_name();

system("pause");

system("cls");

while (true)

{

int choice;

university.show\_information();

cout << "1.Добавить работника университета\n2.Просмотреть список персонала\n3.Редактирование данных\n4.Удаление данных\n5.Поиск\n0.Выход\n";

choice = Check<int>();

while (choice < 0 || choice > 5) {

choice = Check<int>();

};

system("cls");

try{

Staff\* st = new Staff;

switch (choice)

{

case 1: //Добавление информации

{

st->get\_staff();

if (st->get\_st() == 0) {

st->admin->get\_administration();

}

if (st->get\_st() == 1) {

st->lec->get\_lectuter();

}

if (st->get\_st() == 2) {

st->tstaff->get\_technicalstaff();

}

while (1) {

try {

if (st->get\_posobie() < 0) { throw NegativeNumberExeption("Оклад не может быть отрицательным"); }

else

{

break;

}

}

catch (const NegativeNumberExeption e) {

cout << e.m\_error << endl;

system("pause");

}

}

st->employee->get\_employee();

inf.push\_back(st);

system("pause");

}; break;

case 2: //Вывод информации

{

if (!inf.empty()) {

system("cls");

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

cout << i + 1 << ". "<<endl;

inf[i]->show();

}

}

else cout << "Вы ничего не добавили" << endl;

system("pause");

}; break;

case 3: //Редактирование данных

{

if (!inf.empty()) {

system("cls");

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

cout << i + 1 << ". "<<endl;

inf[i]->show();

}

cout << "Введите номер изменяемого работника. Для выхода введите 0\n";

int choice;

choice = Check<int>();

while (choice < 0 || choice > inf.size()) {

cout << "Ошибка ввода. Повторите попытку\n";

choice = Check<int>();

};

if (choice == 0) break;

else inf[choice - 1]->change();

}

else cout << "Вы ничего не добавили" << endl;

system("pause");

}; break;

case 4: //Удаление данных

{

if (!inf.empty()) {

system("cls");

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

cout << i + 1 << ". "<<endl;

inf[i]->show();

}

cout << "Введите номер рабочего которого хотите удалить. Для выхода введите 0\n";

int choice;

choice = Check<int>();

while (choice < 0 || choice > inf.size()) {

cout << "Ошибка ввода. Повторите попытку\n";

choice = Check<int>();

};

if (choice == 0) break;

else {

inf.erase(inf.begin() + choice - 1); //Удаление из контейнера vector

cout << "\n\nЗапись №" << choice << " успешно удалена\n\n" << endl;

}

}

else cout << "Записей нет" << endl;

system("pause");

}; break;

case 5: //Поиск данных

{

if (!inf.empty()) {

system("cls");

float searchpos;

int count=0;

cout << "Введите величину пособия по которому производите поиск" << endl;

searchpos = Check<float>();

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

if (inf[i]->search(searchpos)==true) {

cout << i + 1 << ". "<<endl;

inf[i]->show();

count++;

}

}

if (count == 0) cout << "Не найдено" << endl;

}

else cout << "Вы ничего не добавили" << endl;

system("pause");

}; break;

case 0:

{

return 0;

}; break;

default:

{

cout << "Неверный ввод";

}; break;

}

}

catch (bad\_alloc exept) { cout << "Возникло исключение " << exept.what() << endl; exx(); }

catch (...) { cout << "Сработал абсолютный обработчик" << endl; }

system("cls");

}

return 0;

}