Министерство науки и высшего образования РФ
Пензенский государственный университет
Кафедра “Вычислительная техника”

Отчёт

по лабораторной работе №1
по курсу “Объектно-ориентированное программирование”
на тему “Основы работы с классами и объектами на языке C++”
Вариант 1

Выполнил студент гр. 22ВВВ3:
Кулахметов С.И.

Приняли:
к.т.н., доцент Евсеева Ю.И.
к.т.н., доцент Гудков А.А.

Пенза 2024

Цель работы

Изучить основы работы с классами и объектами на языке C++.

Лабораторное задание

Разработать класс для объекта **Student**. Включить в класс конструктор по умолчанию, конструктор с параметрами, конструктор копирования и деструктор. Поля класса определить с модификаторо доступа private, для доступа к ним реализовать методы Set- и Get- с модификатором доступа private.

Класс **Student** должен содержать следующие поля: Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс. Создать массив объектов. Вывести:

а) список студентов заданного факультета;

б) списки студентов для каждого факультета и курса;

в) список студентов, родившихся после заданного года.

Пояснительный текст к прорамме

Программа разделена на 3 файла, которые включают в себя различные модули лабораторной работы: 1 файл main.cpp — содержит главную часть программы и функцию main(); 2 файл Student.h — содержит описание основного класса **Student** и прототипов методов, содержащихся в нём; 3 файл Student.cpp — содержит реализацию методов класса **Student**.

Результаты работы программы

В начале работы программы необходимо ввести количество студентов, которые будут внесены в список объектов (рис. 1).

Рисунок 1 — Ввод количества студентов для создания массива объектов

После ввода числа студентов генерируется массив объектов класса **Student** данного размера и предлагается ввести данные для каждого студента (рис. 2).

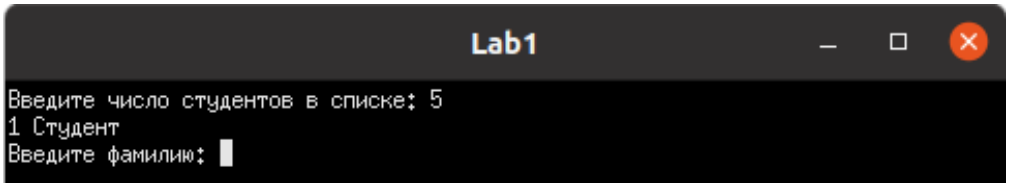

Рисунок 2 — Начало ввода данных о каждом студенте

После чего следует заполнить всю необходимую информацию о студентах (рис. 3).

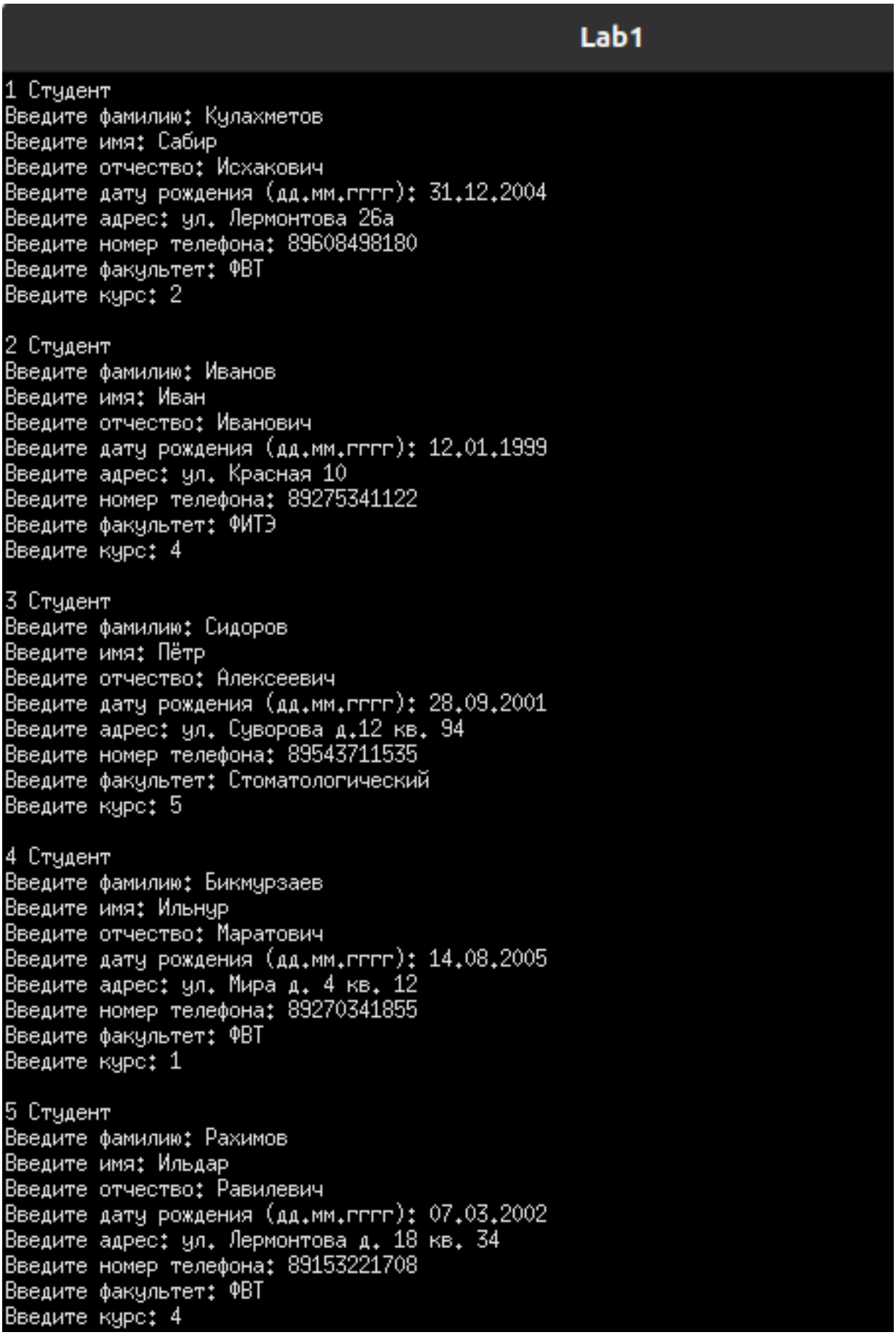

Рисунок 3 — Ввод данных о студентах

Как только все поля будут заполнены, программа выводит меню, в котором мы можем выбрать любой из пунктов, выполняющих задание данной лабораторной работы (рис. 4).

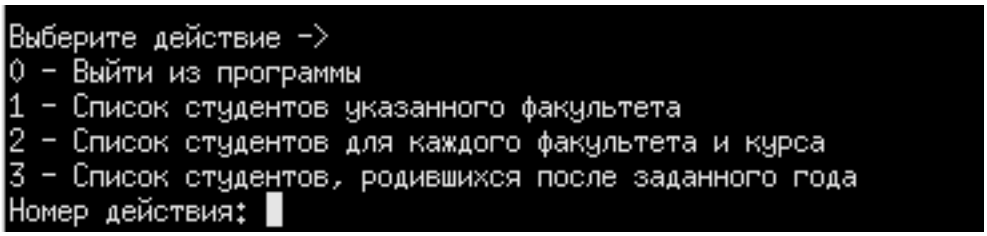

Рисунок 4 — Меню для выбора пункта задания лабораторной работы

Результаты выполнения программы представлены на иллюстрациях 5 — 7.

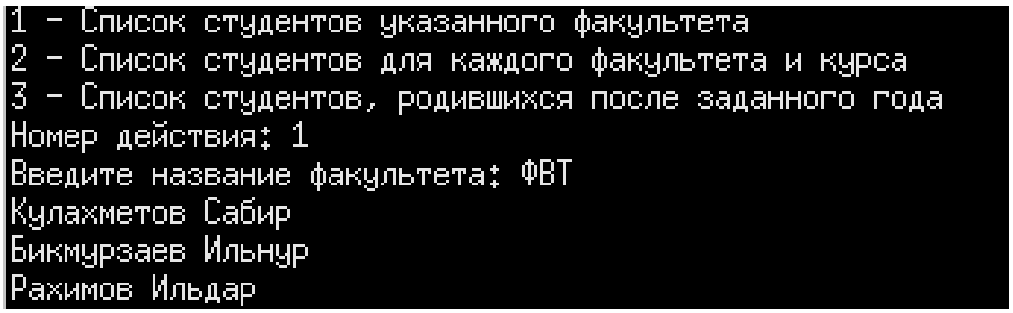

Рисунок 5 — Вывод списка студентов, обучающихся на заданном факультете

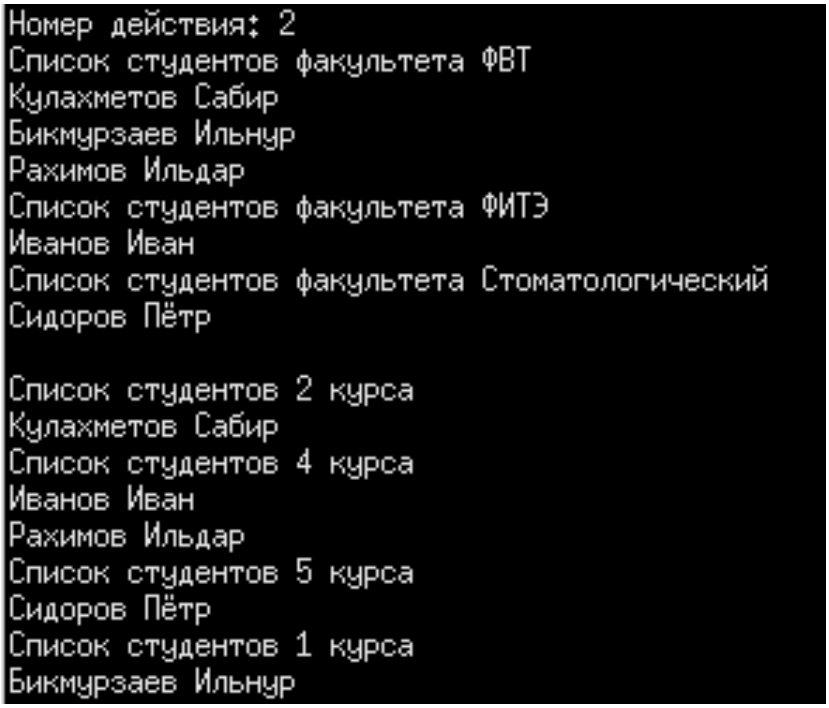

Рисунок 6 — Вывод списков студентов для каждого курса и факультета

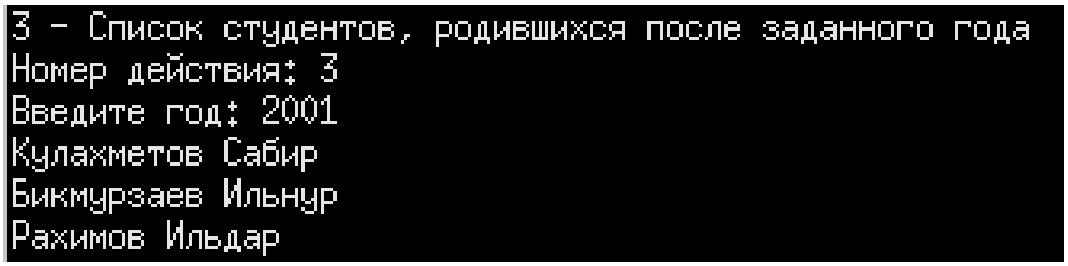

Рисунок 7 — Вывод списка студентов, родившихся после заданного года

Вывод

В ходе выполнения данной лабораторной работы были получены базовые навыки работы с классами и объектами в языке программирования C++.

Ссылка на *GitHub* репозиторий с лабораторной работой

https://github.com/KulakhmetovS/OOP

Приложение А

Листинг программы

Файл main.cpp

```
#include <iostream>
#include <string>
#include <iomanip>
#include <vector>
#include <Student.h>

using namespace std;

void SetData(Student& student, int index);  //Получение данных от пользователя
//Вывод списка студентов по факультетам
void SortByFaculty(Student *stud_array, string faculty, int size);
//Вывод списков студентов по каждому факультету и курсу
void SortByEachFaculty(Student *stud_array, int size);
//Вывод списка студентов, родившихся после указанного года
void AfterCertainYear(Student *stud_array, int size, int Year);

int main()
{
   /*Переменные строки для работы с функцией getline()
   сама же функция является костылём чтобы читать строки с пробелами*/
   string s_size, s_choice, faculty, s_year;
   int size = 0, i, choice = 0, year = 0;

   cout << "Введите число студентов в списке: ";
   getline(cin, s_size);
   size = stoi(s_size);    //Преобразование строки в число
   Student *stud_array = new Student[size];    //Массив объектов класса Student

   //Заполнение каждого объекта массива данными
   for(i = 0; i < size; i++)
      SetData(stud_array[i], i);

   //Цикл для множественного выполнения операций над массивом объектов
   while(1)
   {
      cout << "Выберите действие ->\n";
      cout << "0 - Выйти из программы\n";
      cout << "1 - Список студентов указанного факультета\n";
      cout << "2 - Список студентов для каждого факультета и курса\n";
      cout << "3 - Список студентов, родившихся после заданного года\n";
      cout << "Номер действия: ";
      getline(cin, s_choice);
      choice = stoi(s_choice);

      if(choice == 0) break;
      else if(choice == 1)
      {
         cout << "Введите название факультета: ";
         getline(cin, faculty);

         SortByFaculty(stud_array, faculty, size);
      }
      else if(choice == 2) SortByEachFaculty(stud_array, size);
      else if(choice == 3)
```

```
            {
                cout << "Введите год: ";
                getline(cin, s_year);
                year = stoi(s_year);

                AfterCertainYear(stud_array, size, year);
            }
        }

        delete[] stud_array;
        return 0;
    }


    void SetData(Student& student, int index)
    {
        string surname, name, patronymic, date, address, phone_number, faculty, s_course;
        int course;

        cout << ++index << " Студент" << endl;
        cout << "Введите фамилию: ";
        getline(cin, surname);
        cout << "Введите имя: ";
        getline(cin, name);
        cout << "Введите отчество: ";
        getline(cin, patronymic);

        student.setName(surname, name, patronymic); //Ввод фамилии, имени и отчества

        cout << "Введите дату рождения (дд.мм.гггг): ";
        getline(cin, date);

        student.setDateOfBirth(date);   //Ввод даты рождения

        cout << "Введите адрес: ";
        getline(cin, address);
        cout << "Введите номер телефона: 8";
        getline(cin, phone_number);
        cout << "Введите факультет: ";
        getline(cin, faculty);

        //Ввод контактной информаци: адрес, номер телефона, факультет
        student.setContacts(address, phone_number, faculty);

        cout << "Введите курс: ";
        getline(cin, s_course);
        course = stoi(s_course);

        student.setCourse(course);  //Ввод курса

        cout << endl;
    }

    void SortByFaculty(Student *stud_array, string faculty, int size)
    {
        int count = 0;   //Счётчик для проверки студентов, соответствующих требованиям
```

```
        for(int i = 0; i < size; i++)
                if(stud_array[i].getFaculty() == faculty)    //Сравнение поля объекта и заданного факультета
            {
               //Вывод имени и фамилии соответствующего студента
                cout << stud_array[i].getSurname() << " " << stud_array[i].getName() << endl;
                    count++;      //Увеличение счётчика, если студент соответствует факультету
            }

        if(count == 0)  //Вывод предупреждения, если соответствующих студентов не найдено
        {
        cout << "Студенты данного факультета отсутствуют в списке\n" << \
        "(Или неправильно введено название факультета!)"<< endl;
        }
      }

      void SortByEachFaculty(Student *stud_array, int size)
      {
        //visited_courses - массив для пометки уже указанного курса
        int i, j, *visited_courses = new int[size], int_cmp;
        vector<string> list;    //Вектор, хранящий названия факультетов
        vector<int> Courses;    //Вектор, хранящий курсы студентов
           string *visited_str = new string[size], cmp;      //Массив для пометки уже указанного факультета

        //Запись названий факультетов в массив
        for(i = 0; i < size; i++)
           visited_str[i] = stud_array[i].getFaculty();

        //Составление списка факультетов при помощи вектора
        for(j = 0; j < size; j++)
        {
           if(visited_str[j] != "null")    //Если факультет ещё не помечен
           {
            cmp = visited_str[j];   //Сохранение названия факультета для дальнейшего сравнения
           list.push_back(cmp);    //Помещение названия факультета в вектор

            //Если факультет уже посещён, то заменить все аналогичные факультеты строкой "null"
           for(i = 0; i < size; i++)
              if(visited_str[i] == cmp) visited_str[i] = "null";
           }
        }

          for(i = 0; i < list.size(); i++)      //Киличество итераций цикла равно числу элементов вектора
        {
           cout << "Список студентов факультета " << list[i] << endl;

           //Вывод списков студентов по факультетам
           for(j = 0; j < size; j++)
               if(stud_array[j].getFaculty() == list[i])   //Если поле объекта соответствует текущему факультету
                   cout << stud_array[j].getSurname() << " " << stud_array[j].getName() << endl;
        }
```

```
        cout << "\n";

        delete[] visited_str;   //Удаление использованного массива


        //Составление списка студентов по курсам аналогичено составлению списка по факультетам
        for(i = 0; i < size; i++)
            visited_courses[i] = stud_array[i].getCourse();

        for(j = 0; j < size; j++)
        {
            if(visited_courses[j] != 0)
            {
            int_cmp = visited_courses[j];
            Courses.push_back(int_cmp);

            for(i = 0; i < size; i++)
                if(visited_courses[i] == int_cmp) visited_courses[i] = 0;
            }
        }

        for(i = 0; i < Courses.size(); i++)
        {
            cout << "Список студентов " << Courses[i] << " курса"   << endl;

            for(j = 0; j < size; j++)
                if(stud_array[j].getCourse() == Courses[i])
                      cout << stud_array[j].getSurname() << " " << stud_array[j].getName() << endl;
            }

            delete[] visited_courses;
        }

        void AfterCertainYear(Student *stud_array, int size, int Year)
        {
            for(int i = 0; i < size; i++)
                if(stud_array[i].getYear() > Year)
                     cout << stud_array[i].getSurname() << " " << stud_array[i].getName() << endl;
        }
```

Файл Student.h

```
        #ifndef STUDENT_H
        #define STUDENT_H
        #include <iostream>

        using namespace std;

        class Student   //Главный класс
        {
        /*Так как не было понятно для чего можно использовать конструкторы и деструкторы в данной задаче,
            принято решение просто закомментировать их с последующим объяснением принципов их работы преподавателю

        public:
            Student(const Student& other)
```

```
        {
            cout << "Вызов конструктора копирования" << this << endl;
            this->mark = other.mark;
            this->date = other.date;
        }
         Student() {cout << "Вызов конструктора " << this << endl;} //Конструктор поумолчанию
        Student(int mark, int date) //Конструктор с параметрами
        {
            cout << "Вызов конструктора с парамеирами " << this << endl;
            this->mark = mark;  //this указывает на поле объекта
            this->date = date;
        }
        ~Student() {cout << "Вызов деструктора " << this << endl;} //Деструктор
    */

    private:    //Поля класса
        //int mark, date;
        string surname = "Нет данных"; //Фамилия
        string name = "Нет данных";    //Имя
        string patronymic = "Нет данных";  //Отчество
        string date = "Нет данных";    //Дата рождения
        string address = "Нет данных"; //Адрес
        string phone_number = "Нет данных";    //Номер телефона
        string faculty = "Нет данных"; //Факультет
        int course = 0;  //Курс

    public: //Прототипы методов
            void setName(string surname, string name, string patronymic);     //Запись фамилии, имени и отчества
         void setContacts(string address, string phone_number, string faculty);  //Запись адреса, номера и факультета
        void setDateOfBirth(string date);   //Запись даты рождения
        void setCourse(int course); //Запись курса

        string getSurname() {return surname;};  //Получение фамилии
        string getName() {return name;};    //Получение имени
        string getPatrinymic() {return patronymic;};    //Получение отчества
        string getDateOfBirth() {return date;}; //Получение даты рождения
        string getAddress() {return address;};  //Получение адреса
             string getPhoneNumber() {return phone_number;}; //Получение номера телефона
        string getFaculty() {return faculty;};  //Получение факультета
        int getCourse() {return course;};   //Получение курса
        int getYear();
    };

    #endif // STUDENT_H
```

Файл Student.cpp

```
    #include "Student.h"
    #include <iomanip>
    #include <string>
    //Реализация самих методов
    void Student::setName(string surname, string name, string patronymic)
    {
        //this указывает на поле объекта
        this->surname = surname;
```

```
    this->name = name;
    this->patronymic = patronymic;
}

void Student::setContacts(string address, string phone_number, string faculty)
{
    this->address = address;

    if(phone_number.size() == 10)   //Проверка на количество символов
    {
        this->phone_number = phone_number;

        for(int i = 0; i < 10; i++)
        {
            //Проверка на соответствие символа цифре
            if((phone_number[i] < 48) || (phone_number[i] > 57))
            {
                this->phone_number = "Некорректно введённые данные";
            }
        }
    }
    else this->phone_number = "Некорректно введённые данные";

    this->faculty = faculty;
}

void Student::setDateOfBirth(string date)
{
    if(date.size() == 10)   //Проверка на количество символов
    {
        this->date = date;

        for(int i = 0; i < 10; i++)
        {
            //Проверка на соотвествие символа цифре и на правильность формата
            if((date[i] < 46) || (date[i] > 57))
            {
                this->date = "Некорректно введённые данные";
            }
             if((date[2] != 46) || (date[5] !=46)) this->date = "Некорректно введённые
данные";
        }
    }
    else this->date = "Некорректно введённые данные";
}

void Student::setCourse(int course)
{
    if((course > 0) && (course < 7))
        this->course = course;
}

int Student::getYear()
{
    string s_year;
    int year = 0;

    for(int i = 0; i < 4; i++)
        s_year[i] = date[i+6];
    //Обработка исключаний
```

```
        try
        {
           year = stoi(s_year);    //Если невозможно преобразовать данные в число
        }
        catch(const invalid_argument &e)
        {
           year = 0;
             return year;    //То возвращаем вместо года просто 0 чтобы программа не
дампнулась
        }

         return year;     //Если проверка прошла успешно, то возвращается год в виде
числа
      }
```