

[[Начало](#)][[Далее](#)]

Цель работы: изучение и практическое освоение основ объектно-ориентированного программирования в C++.

Лабораторная работа №1. Объекты и классы в C++

Содержание

1. Теоретические сведения

- [Введение в объектно-ориентированное программирование](#)
- [Модификаторы доступа public и private](#)
- [Программа учета успеваемости студентов](#)
- [Отделение данных от логики](#)
- [Создание объекта через указатель](#)
- [Конструктор и деструктор класса](#)
- [Конструктор Students](#)
- [Сохранение оценок в файл](#)
- [Деструктор Students](#)

2. Индивидуальные задания

- [Задание №1](#)
- [Задание №2](#)
- [Задание №3](#)

Введение в объектно-ориентированное программирование

Весь реальный мир состоит из объектов. Города состоят из районов, в каждом районе есть свои названия улиц, на каждой улице находятся жилые дома, которые также состоят из объектов.

Практически любой материальный предмет можно представить в виде совокупности объектов, из которых он состоит. Допустим, что нам нужно написать программу для учета успеваемости студентов. Можно представить группу студентов, как класс языка C++. Назовем его *Students*.

```
class Students {  
    // Имя студента  
    std::string name;  
    // Фамилия  
    std::string last_name;  
    // Пять промежуточных оценок студента  
    int scores[5];  
    // Итоговая оценка за семестр  
    float average_ball;  
};
```

Классы в программировании состоят из свойств и методов. Свойства — это любые данные, которыми можно характеризовать объект класса. В нашем случае, объектом класса является студент, а его свойствами — имя, фамилия, оценки и средний балл.

У каждого студента есть имя — *name* и фамилия *last_name*. Также, у него есть промежуточные оценки за весь семестр. Эти оценки мы будем записывать в целочисленный массив из пяти элементов. После того, как все пять оценок будут проставлены, определим средний балл успеваемости студента за весь семестр — свойство *average_ball*.

Методы — это функции, которые могут выполнять какие-либо действия над данными (свойствами) класса. Добавим в наш класс функцию *calculate_average_ball()*, которая будет определять средний балл успеваемости ученика.

Методы класса — это его функции.

Свойства класса — его переменные.

```
class Students {  
    public:  
        // Функция, считающая средний балл  
        void calculate_average_ball()  
        {  
            int sum = 0; // Сумма всех оценок
```

```

        for (int i = 0; i < 5; ++i) {
            sum += scores[i];
        }
        // считаем среднее арифметическое
        average_ball = sum / 5.0;
    }

    // Имя студента
    std::string name;
    // Фамилия
    std::string last_name;
    // Пять промежуточных оценок студента
    int scores[5];

private:
    // Итоговая оценка за семестр
    float average_ball;
};

```

Функция *calculate_average_ball()* просто делит сумму всех промежуточных оценок на их количество.

[[В начало](#)]

Модификаторы доступа *public* и *private*

Все свойства и методы классов имеют права доступа. По умолчанию, все содержимое класса является доступным для чтения и записи только для него самого. Для того, чтобы разрешить доступ к данным класса извне, используют модификатор доступа *public*. Все функции и переменные, которые находятся после модификатора *public*, становятся доступными из всех частей программы.

Закрытые данные класса размещаются после модификатора доступа *private*. Если отсутствует модификатор *public*, то все функции и переменные, по умолчанию являются закрытыми (как в первом примере).

Обычно, приватными делают все свойства класса, а публичными — его методы. Все действия с закрытыми свойствами класса реализуются через его методы. Рассмотрим следующий код.

```

class Students {
public:
    // Установка среднего балла
    void set_average_ball(float ball)
    {
        average_ball = ball;
    }
    // Получение среднего балла
    float get_average_ball()
    {
        return average_ball;
    }
    std::string name;
    std::string last_name;
    int scores[5];

private:
    float average_ball;
};

```

Мы не можем напрямую обращаться к закрытым данным класса. Работать с этими данными можно только посредством методов этого класса. В примере выше, мы используем функцию *get_average_ball()* для получения средней оценки студента, и *set_average_ball()* для выставления этой оценки.

Функция *set_average_ball()* принимает средний балл в качестве параметра и присваивает его значение закрытой переменной *average_ball*. Функция *get_average_ball()* просто возвращает значение этой переменной.

[[В начало](#)]

Программа учета успеваемости студентов

Создадим программу, которая будет заниматься учетом успеваемости студентов в группе. Создайте заголовочный файл *students.h*, в котором будет находиться класс *Students*.

```
/* students.h */
#include <string>

class Students {
public:
    // Установка имени студента
    void set_name(std::string student_name)
    {
        name = student_name;
    }
    // Получение имени студента
    std::string get_name()
    {
        return name;
    }
    // Установка фамилии студента
    void set_last_name(std::string student_last_name)
    {
        last_name = student_last_name;
    }
    // Получение фамилии студента
    std::string get_last_name()
    {
        return last_name;
    }
    // Установка промежуточных оценок
    void set_scores(int student_scores[])
    {
        for (int i = 0; i < 5; ++i) {
            scores[i] = student_scores[i];
        }
    }
    // Установка среднего балла
    void set_average_ball(float ball)
    {
        average_ball = ball;
    }
    // Получение среднего балла
    float get_average_ball()
    {
        return average_ball;
    }

private:
    // Промежуточные оценки
    int scores[5];
    // Средний балл
    float average_ball;
    // Имя
    std::string name;
    // Фамилия
    std::string last_name;
};
```

Мы добавили в наш класс новые методы, а также сделали приватными все его свойства. Функция *set_name()* сохраняет имя студента в переменной *name*, а *get_name()* возвращает значение этой переменной. Принцип работы функций *set_last_name()* и *get_last_name()* аналогичен.

Функция *set_scores()* принимает массив с промежуточными оценками и сохраняет их в приватную переменную *int scores[5]*.

Теперь создайте файл *main.cpp* со следующим содержимым.

```
/* main.cpp */
#include <iostream>
```

```

#include "students.h"

int main()
{
    // Создание объекта класса Student
    Students student;

    std::string name;
    std::string last_name;

    // Ввод имени с клавиатуры
    std::cout << "Name: ";
    getline(std::cin, name);

    // Ввод фамилии
    std::cout << "Last name: ";
    getline(std::cin, last_name);

    // Сохранение имени и фамилии в объект класса Students
    student.set_name(name);
    student.set_last_name(last_name);

    // Оценки
    int scores[5];
    // Сумма всех оценок
    int sum = 0;

    // Ввод промежуточных оценок
    for (int i = 0; i < 5; ++i) {
        std::cout << "Score " << i+1 << ": ";
        std::cin >> scores[i];
        // суммирование
        sum += scores[i];
    }

    // Сохраняем промежуточные оценки в объект класса Student
    student.set_scores(scores);
    // Считаем средний балл
    float average_ball = sum / 5.0;
    // Сохраняем средний балл в объект класса Students
    student.set_average_ball(average_ball);
    // Выводим данные по студенту
    std::cout << "Average ball for " << student.get_name() << " "
        << student.get_last_name() << " is "
        << student.get_average_ball() << std::endl;

    return 0;
}

```

В самом начале программы создается объект класса *Students*. Дело в том, что сам класс является только описанием его объекта. Класс *Students* является описанием любого из студентов, у которого есть имя, фамилия и возможность получения оценок.

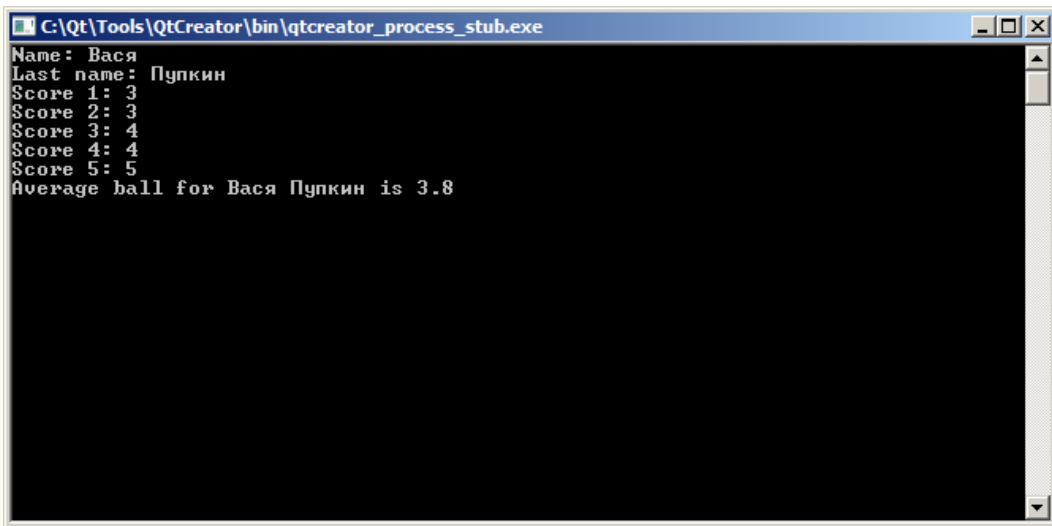
Объект класса *Students* характеризует конкретного студента. Если мы захотим выставить оценки всем ученикам в группе, то будем создавать новый объект для каждого из них. Использование классов очень хорошо подходит для описания объектов реального мира.

После создания объекта *student*, мы вводим с клавиатуры фамилию, имя и промежуточные оценки для конкретного ученика. Пускай это будет Вася Пупкин, у которого есть пять оценок за семестр — две тройки, две четверки и одна пятерка.

Введенные данные мы передаем *set*-функциям, которые присваивают их закрытым переменным класса. После того, как были введены промежуточные оценки, мы высчитываем средний балл на основе этих оценок, а затем сохраняем это значение в закрытом свойстве *average_ball*, с помощью функции *set_average_ball()*.

Скомпилируйте и запустите программу.

Пример работы программы *Students*.

A screenshot of a Qt Creator console window. The title bar shows the path 'C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe'. The console output is as follows:

```
Name: Вася
Last name: Пупкин
Score 1: 3
Score 2: 3
Score 3: 4
Score 4: 4
Score 5: 5
Average ball for Вася Пупкин is 3.8
```

[[В начало](#)]

Отделение данных от логики

Вынесем реализацию всех методов класса в отдельный файл *students.cpp*.

```
/* students.cpp */
#include <string>
#include "students.h"

// Установка имени студента
void Students::set_name(std::string student_name)
{
    Students::name = student_name;
}

// Получение имени студента
std::string Students::get_name()
{
    return Students::name;
}

// Установка фамилии студента
void Students::set_last_name(std::string student_last_name)
{
    Students::last_name = student_last_name;
}

// Получение фамилии студента
std::string Students::get_last_name()
{
    return Students::last_name;
}

// Установка промежуточных оценок
void Students::set_scores(int scores[])
{
    for (int i = 0; i < 5; ++i) {
        Students::scores[i] = scores[i];
    }
}

// Установка среднего балла
void Students::set_average_ball(float ball)
{
    Students::average_ball = ball;
}

// Получение среднего балла
float Students::get_average_ball()
{

```

```
    return Students::average_ball;
}
```

А в заголовочном файле *students.h* оставим только прототипы этих методов.

```
/* students.h */
#pragma once /* Защита от двойного подключения заголовочного файла */
#include <string>

class Students {
public:
    // Установка имени студента
    void set_name(std::string);
    // Получение имени студента
    std::string get_name();
    // Установка фамилии студента
    void set_last_name(std::string);
    // Получение фамилии студента
    std::string get_last_name();
    // Установка промежуточных оценок
    void set_scores(int []);
    // Установка среднего балла
    void set_average_ball(float);
    // Получение среднего балла
    float get_average_ball();

private:
    // Промежуточные оценки
    int scores[5];
    // Средний балл
    float average_ball;
    // Имя
    std::string name;
    // Фамилия
    std::string last_name;
};
```

Такой подход называется абстракцией данных — одного из фундаментальных принципов объектно-ориентированного программирования. К примеру, если кто-то другой захочет использовать наш класс в своем коде, ему не обязательно знать, как именно высчитывается средний балл. Он просто будет использовать функцию *calculate_average_ball()* из второго примера, не вникая в алгоритм ее работы.

Над крупными проектами обычно работает несколько программистов. Каждый из них занимается написанием определенной части продукта. В таких масштабах кода, одному человеку практически нереально запомнить, как работает каждая из внутренних функций проекта. В нашей программе, мы используем оператор потокового вывода *cout*, не задумываясь о том, как он реализован на низком уровне. Кроме того, отделение данных от логики является хорошим тоном программирования.

В начале обучения мы говорили о пространствах имен (*namespaces*). Каждый класс в C++ использует свое пространство имен. Это сделано для того, чтобы избежать конфликтов при именовании переменных и функций. В файле *students.cpp* мы используем оператор принадлежности *::* перед именем каждой функции. Это делается для того, чтобы указать компилятору, что эти функции принадлежат классу *Students*.

[[В начало](#)]

Создание объекта через указатель

При создании объекта, лучше не копировать память для него, а выделять ее в куче с помощью указателя. И освобождать ее после того, как мы закончили работу с объектом. Реализуем это в нашей программе, немного изменив содержимое файла *main.cpp*.

```
/* main.cpp */
#include <iostream>
#include "students.h"
```

```

int main()
{
    // Выделение памяти для объекта Students
    Students *student = new Students;

    std::string name;
    std::string last_name;

    // Ввод имени с клавиатуры
    std::cout << "Name: ";
    getline(std::cin, name);

    // Ввод фамилии
    std::cout << "Last name: ";
    getline(std::cin, last_name);

    // Сохранение имени и фамилии в объект класса Students
    student->set_name(name);
    student->set_last_name(last_name);

    // Оценки
    int scores[5];
    // Сумма всех оценок
    int sum = 0;

    // Ввод промежуточных оценок
    for (int i = 0; i < 5; ++i) {
        std::cout << "Score " << i+1 << ": ";
        std::cin >> scores[i];
        // суммирование
        sum += scores[i];
    }
    // Сохраняем промежуточные оценки в объект класса Student
    student->set_scores(scores);

    // Считаем средний балл
    float average_ball = sum / 5.0;
    // Сохраняем средний балл в объект класса Students
    student->set_average_ball(average_ball);
    // Выводим данные по студенту
    std::cout << "Average ball for " << student->get_name() << " "
        << student->get_last_name() << " is "
        << student->get_average_ball() << std::endl;
    // Удаление объекта student из памяти
    delete student;
    return 0;
}

```

При создании статического объекта, для доступа к его методам и свойствам, используют операция прямого обращения — «.» (символ точки). Если же память для объекта выделяется посредством указателя, то для доступа к его методам и свойствам используется оператор косвенного обращения — «->».

[[В начало](#)]

Конструктор и деструктор класса

Конструктор класса — это специальная функция, которая автоматически вызывается сразу после создания объекта этого класса. Он не имеет типа возвращаемого значения и должен называться также, как класс, в котором он находится. По умолчанию, заполним двойками массив с промежуточными оценками студента.

```

class Students {
public:
    // Конструктор класса Students
    Students(int default_score)
    {
        for (int i = 0; i < 5; ++i) {

```

```

        scores[i] = default_score;
    }
}

private:
    int scores[5];
};

int main()
{
    // Передаем двойку в конструктор
    Students *student = new Students(2);
    return 0;
}

```

Мы можем исправить двойки, если ученик будет хорошо себя вести, и вовремя сдавать домашние задания. А на «нет» и суда нет :-)

Деструктор класса вызывается при уничтожении объекта. Имя деструктора аналогично имени конструктора, только в начале ставится знак тильды ~. Деструктор не имеет входных параметров.

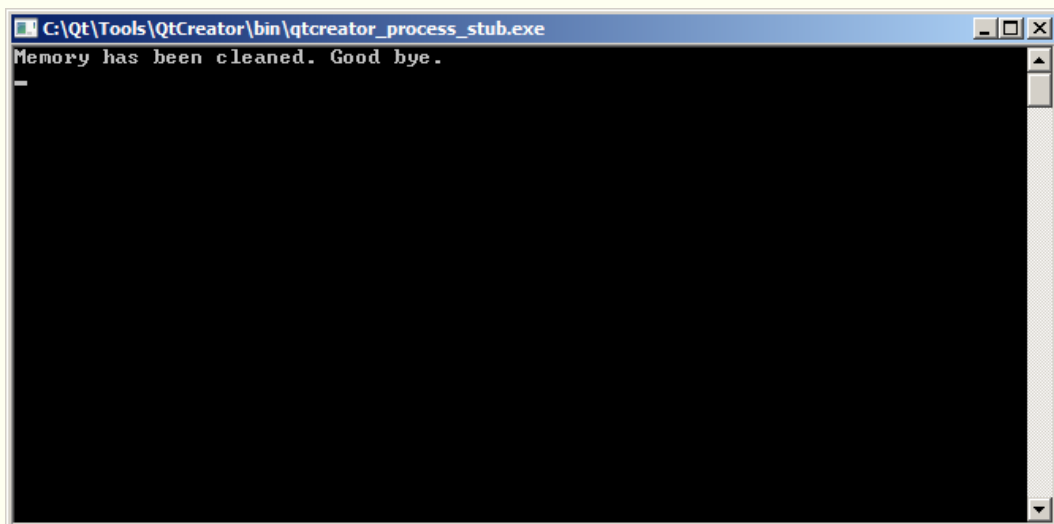
```

#include <iostream>

class Students {
public:
    // Деструктор
    ~Students()
    {
        std::cout << "Memory has been cleaned. Good bye." << std::endl;
    }
};

int main()
{
    Students *student = new Students;
    // Уничтожение объекта
    delete student;
    return 0;
}

```



[[В начало](#)]

Конструктор *Students*

Добавим в класс *Students* конструктор, который будет принимать имя и фамилию ученика, и сохранять эти значения в соответствующих переменных класса.

```

// Конструктор Students
Students::Students(std::string name, std::string last_name)
{
    Students::set_name(name);
}

```



```
Students::set_last_name(last_name);  
}
```

При создании нового объекта, мы должны передать конструктору имя и фамилию студента. Иначе компиляция программы завершится с ошибкой.

```
std::string name = "Василий";  
std::string last_name = "Пупкин";  
Students *student = new Students(name, last_name);
```

Теперь добавим прототип конструктора в файл *students.h*.

```
/* students.h */  
#pragma once /* Защита от двойного подключения заголовочного файла */  
#include <string>  
  
class Students {  
public:  
    // Конструктор класса Students  
    Students(std::string, std::string);  
  
    // Установка имени студента  
    void set_name(std::string);  
    // Получение имени студента  
    std::string get_name();  
  
    // Установка фамилии студента  
    void set_last_name(std::string);  
    // Получение фамилии студента  
    std::string get_last_name();  
  
    // Установка промежуточных оценок  
    void set_scores(int []);  
  
    // Установка среднего балла  
    void set_average_ball(float);  
    // Получение среднего балла  
    float get_average_ball();  
private:  
    // Промежуточные оценки  
    int scores[5];  
    // Средний балл  
    float average_ball;  
    // Имя  
    std::string name;  
    // Фамилия  
    std::string last_name;  
};
```

В файле *students.cpp* определим сам конструктор.

```
/* students.cpp */  
#include <string>  
#include <fstream>  
#include "students.h"  
  
// Конструктор Students  
Students::Students(std::string name, std::string last_name)  
{  
    Students::set_name(name);  
    Students::set_last_name(last_name);  
}  
  
// Установка имени студента  
void Students::set_name(std::string student_name)  
{  
    Students::name = student_name;  
}
```

```

// Получение имени студента
std::string Students::get_name()
{
    return Students::name;
}

// Установка фамилии студента
void Students::set_last_name(std::string student_last_name)
{
    Students::last_name = student_last_name;
}

// Получение фамилии студента
std::string Students::get_last_name()
{
    return Students::last_name;
}

// Установка промежуточных оценок
void Students::set_scores(int scores[])
{
    int sum = 0;
    for (int i = 0; i < 5; ++i) {
        Students::scores[i] = scores[i];
        sum += scores[i];
    }
}

// Установка среднего балла
void Students::set_average_ball(float ball)
{
    Students::average_ball = ball;
}

// Получение среднего балла
float Students::get_average_ball()
{
    return Students::average_ball;
}

```

В **main()** мы принимаем от пользователя имя и фамилию ученика, и сохраняем их во временных локальных переменных. После этого создаем новый объект класса **Students**, передавая его конструктору эти переменные.

```

/* main.cpp */
#include <iostream>
#include "students.h"

int main(int argc, char *argv[])
{
    // Локальная переменная, хранящая имя ученика
    std::string name;
    // И его фамилию
    std::string last_name;

    // Ввод имени
    std::cout << "Name: ";
    getline(std::cin, name);
    // И фамилии
    std::cout << "Last name: ";
    getline(std::cin, last_name);

    // Передача параметров конструктору
    Students *student = new Students(name, last_name);

    // Оценки
    int scores[5];
    // Сумма всех оценок
    int sum = 0;

```

```

// Ввод промежуточных оценок
for (int i = 0; i < 5; ++i) {
    std::cout << "Score " << i+1 << ": ";
    std::cin >> scores[i];
    // суммирование
    sum += scores[i];
}
// Сохраняем промежуточные оценки в объект класса Student
student->set_scores(scores);

// Считаем средний балл
float average_ball = sum / 5.0;
// Сохраняем средний балл
student->set_average_ball(average_ball);

// Выводим данные по студенту
std::cout << "Average ball for " << student->get_name() << " "
    << student->get_last_name() << " is "
    << student->get_average_ball() << std::endl;
// Удаление объекта student из памяти
delete student;
return 0;
}

```

[[В начало](#)]

Сохранение оценок в файл

Чтобы после завершения работы с программой, все данные сохранялись, мы будем записывать их в текстовый файл.

Оценки каждого студента будут находится в отдельной строке. Имя и фамилии будут разделяться пробелами. После имени и фамилии ученика ставится еще один пробел, а затем перечисляются все его оценки.

Пример файла с оценками:

```

Василий Пупкин 5 4 5 3 3
Иван Сидоров   5 5 3 4 5
Андрей Иванов  5 3 3 3 3

```

Для работы с файлами мы воспользуемся библиотекой *fstream*, которая подключается в заголовочном файле с таким же именем.

```

#include <fstream>

// Запись данных о студенте в файл
void Students::save()
{
    std::ofstream fout("students.txt", std::ios::app);

    fout << Students::get_name() << " "
        << Students::get_last_name() << " ";

    for (int i = 0; i < 5; ++i) {
        fout << Students::scores[i] << " ";
    }

    fout << std::endl;
    fout.close();
}

```

Переменная *fout* — это объект класса *ofstream*, который находится внутри библиотеки *fstream*. Класс *ofstream* используется для записи каких-либо данных во внешний файл. Кстати, у него тоже есть конструктор. Он принимает в качестве параметров имя выходного файла и режим записи.

В данном случае, мы используем режим добавления — *std::ios::app* (англ. append). После завершения работы с файлом, необходимо вызвать метод *close()* для того, чтобы закрыть файловый дескриптор.

Чтобы сохранить оценки студента, мы будем вызывать только что созданный метод *save()*.

```
Students student = new Students("Василий", "Пупкин");
student->save();
```

[[В начало](#)]

Деструктор *Students*

Логично было бы сохранять все оценки после того, как работа со студентом закончена. Для этого создадим деструктор класса *Students*, который будет вызывать метод *save()* перед уничтожением объекта.

```
// Деструктор Students
Students::~~Students()
{
    Students::save();
}
```

Добавим прототипы деструктора и метода *save()* в *students.h*.

```
/* students.h */
#pragma once /* Защита от двойного подключения заголовочного файла */
#include <string>

class Students {
public:
    // Запись данных о студенте в файл
    void save();
    // Деструктор класса Students
    ~Students();

    // Конструктор класса Students
    Students(std::string, std::string);
    // Установка имени студента
    void set_name(std::string);
    // Получение имени студента
    std::string get_name();

    // Установка фамилии студента
    void set_last_name(std::string);
    // Получение фамилии студента
    std::string get_last_name();

    // Установка промежуточных оценок
    void set_scores(int []);
    // Получение массива с промежуточными оценками
    int *get_scores();
    // Получение строки с промежуточными оценками
    std::string get_scores_str(char);

    // Установка среднего балла
    void set_average_ball(float);
    // Получение среднего балла
    float get_average_ball();
private:
    // Промежуточные оценки
    int scores[5];
    // Средний балл
    float average_ball;
    // Имя
    std::string name;
    // Фамилия
    std::string last_name;
};
```

И определим эти функции в *students.cpp*.

```

/* students.cpp */
#include <string>
#include <fstream>

#include "students.h"

// Деструктор Students
Students::~Students()
{
    Students::save();
}

// Запись данных о студенте в файл
void Students::save()
{
    std::ofstream fout("students.txt", std::ios::app);

    fout << Students::get_name() << " "
          << Students::get_last_name() << " ";

    for (int i = 0; i < 5; ++i) {
        fout << Students::scores[i] << " ";
    }

    fout << std::endl;
    fout.close();
}

// Конструктор Students
Students::Students(std::string name, std::string last_name)
{
    Students::set_name(name);
    Students::set_last_name(last_name);
}

// Установка имени студента
void Students::set_name(std::string student_name)
{
    Students::name = student_name;
}

// Получение имени студента
std::string Students::get_name()
{
    return Students::name;
}

// Установка фамилии студента
void Students::set_last_name(std::string student_last_name)
{
    Students::last_name = student_last_name;
}

// Получение фамилии студента
std::string Students::get_last_name()
{
    return Students::last_name;
}

// Установка промежуточных оценок
void Students::set_scores(int scores[])
{
    int sum = 0;
    for (int i = 0; i < 5; ++i) {
        Students::scores[i] = scores[i];
        sum += scores[i];
    }
}

```

```
// Получение массива с промежуточными оценками
int *Students::get_scores()
{
    return Students::scores;
}

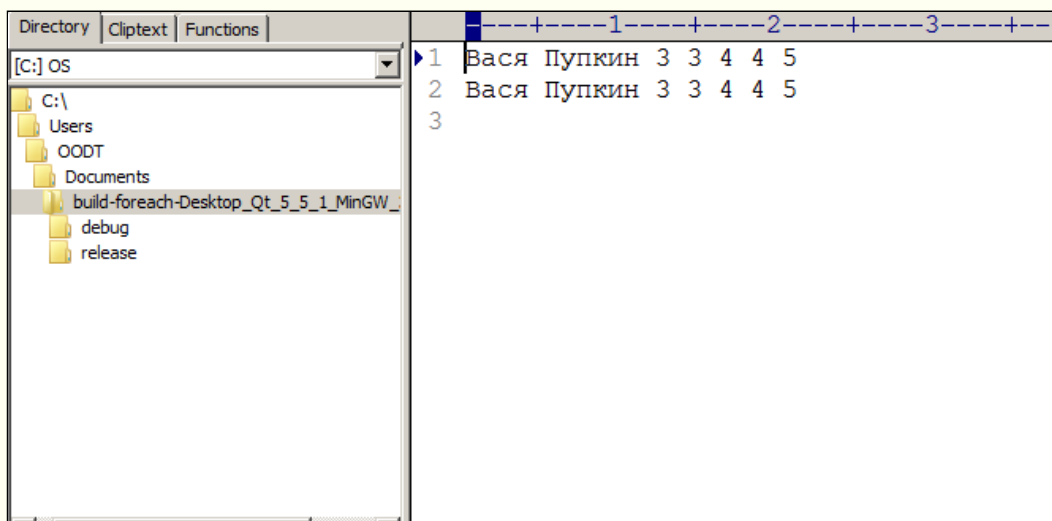
// Установка среднего балла
void Students::set_average_ball(float ball)
{
    Students::average_ball = ball;
}

// Получение среднего балла
float Students::get_average_ball()
{
    return Students::average_ball;
}
```

Содержимое **main.cpp** останется прежним. Скомпилируйте и запустите программу. Перед тем, как приложение завершит свою работу, в директории с исполняемым файлом будет создан новый текстовый файл с оценками — **students.txt**.

A screenshot of a Qt console window titled "C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe". The output text is as follows:

```
Name: Вася
Last name: Пупкин
Score 1: 3
Score 2: 3
Score 3: 4
Score 4: 4
Score 5: 5
Average ball for Вася Пупкин is 3.8
```



[[В начало](#)]

2 Варианты индивидуальных заданий

Задание: Разработать программу, реализующую логику в соответствии с вариантом:

№ варианта	Текст задания
------------	---------------

1	В текстовом файле фиксируются результаты измерения температуры окружающей среды. Результаты измерения температуры фиксируются четыре раза в сутки (утро, день, вечер, ночь). Каждая строка текстового файла начинается с даты в формате YYYY-MM-DD, после которой указываются результаты измерений. Создать 2 файла исходных данных на одинаковые даты двух лет (текущего и прошлого года) с результатами измерений за 14 дней.
2	В текстовом файле фиксируются результаты проведения экзаменационной сессии по четырем дисциплинам. В первой строке перечисляются наименования дисциплин. Количество дисциплин может варьироваться. В следующих строках указываются Фамилия И.О. студента и полученные им оценки: неудовлетворительно, удовлетворительно, хорошо и отлично. Количество оценок в строке должно совпадать с количеством дисциплин. Подготовить два файла данных для двух разных групп, сдававших экзамен в текущем и прошлом году. Количество и наименование дисциплин в файлах должны совпадать. Количество студентов может отличаться. ФИО студентов различны.
3	В текстовом файле фиксируются результаты измерения скорости и направления ветра. Результаты фиксируются четыре раза в сутки (утро, день, вечер, ночь). Каждая строка текстового файла начинается с даты в формате YYYY-MM-DD, после которой указываются результаты измерений в виде буквы (с, ю, з, в) и скорости в метрах в секунду. Создать 2 файла исходных данных на одинаковые даты двух лет (текущего и прошлого года) с результатами измерений за 14 дней.
4	В текстовом файле фиксируется дата и количество пассажиров купивших билет междугородний автобус (исходя из расписания, согласно которому каждый день автобус выполняет несколько рейсов в течение суток). Подготовить два файла данных для двух месяцев.
5	В текстовом файле фиксируется время начала движения и время выполнения маршрута городского автобуса для 4-х типовых дней недели (пн-чт, пт, сб, вс) . Подготовить два файла данных для двух разных лет, но с одинаковым начальным временем движения. В течение дня автобусы выполняют движения по маршруту не менее 14 раз: утром и вечером 2-3 раза в час, днем - 1-2 раза.
6	В первом текстовом файле ежедневно фиксируется дата и значение курса доллара в течение месяца для ЦБ РФ. В аналогичном текстовом файле фиксируется курс покупки и продажи валюты некоторого коммерческого банка.
7	В текстовом файле фиксируется продажа билетов на междугородний автобус. В начале строки указывается дата и время, затем количество пенсионеров, детей и обычных граждан, купивших билеты. Создать массив регистрации продажи билетов для автобуса ПАЗ, имеющего 24 посадочных места. До 20% мест могут быть незаполнены. Детский билет стоит 50%, взрослый 100%, пенсионерам предоставляется скидка 25%.
8	В текстовом файле фиксируется продажа билетов на электричку. В начале строки указывается номер вагона, затем количество пенсионеров, детей и обычных граждан, купивших билеты в данный вагон. Создать два файла регистрации продажи билетов на поезд для текущего и прошлого дней. В каждом вагоне электрички - 72 места. До 20% мест могут быть незаполнены. Детский билет стоит 50%, взрослый 100%, пенсионерам предоставляется скидка 25%. Для простоты считаем, что электричка ходит по маршруту из двух станций.
9	В файле фиксируются результаты голосования кафедры, состоящего из 15 сотрудников по принципу: в начале строки указывается Фамилия кандидата, потом результаты голосования по каждому сотруднику, выраженному словами "за", "против", воздержался, отсутствовал. Выборы проводились по четырем кандидатурам. Выборы считаются состоявшимися если «за» проголосовало свыше 51 % голосовавших и несостоявшимися в противном случае, при этом на голосовании присутствовало более 2/3 состава кафедры.
10	В массиве фиксируются результаты проведения зачетов по четырем дисциплинам по правилу: не зачтено, зачтено, не явился, не допущен. В группе 18 человек. Число студентов, сдавших зачет составляет 60%. На зачет не явилось – 2 человека. Не допущено к зачету – 3. Остальные студенты получили оценку «не зачтено».
11	Результаты измерения среднесуточной дневной и ночной температуры окружающей среды фиксируются в текстовом файле каждый день в течение месяца. Номер первого индекса в массиве соответствует времени суток: утро, день, вечер, ночь. Затем идут результаты измерений (от 28 до 31 в зависимости от числа дней в месяце. Необходимо создать два файла с данными за один и тот же месяц прошлого и текущего годов.
12	Результаты выступления команды в спортивных соревнованиях фиксируются в турнирной таблице в виде файла, В первой строке идет количество команд (не менее 8-ми), затем их наименования. В следующих строках в начале идет номер команды, а затем результаты ее встречи. Значение элемента в массиве соответствует набранным очкам: 0 – поражение команды I от команды J, 1 – ничья, 2 – победа команды I над командой J. Создать два файла - результаты прошлого и текущего сезонов."

2.1 Задание 1

Создать класс, реализующий:

1) чтение данных из файла; 2) вывод результатов на экран.

[[В начало](#)]

2.2 Задание 2

№ варианта	Задание	№ варианта	Задание
1	Создать методы класса, реализующие: 1)вычисление среднесуточной температуры;	2	Создать методы класса, реализующие: 1)вычисление среднего балла, набранного студентами исходя из того, что оценка неудовлетворительно приравнивается к 2, удовлетворительно к 3, хорошо к 4 и отлично к 5.

	2) вывод результатов на экран; 3) вывод результатов в файл;		2) вывод результатов на экран; 3) вывод результатов в файл;
3	Создать методы класса, реализующие: 1)вычисление среднесуточной скорости ветра; 2) вывод результатов на экран; 3) вывод результатов в файл;	4	Создать методы класса, реализующие: 1)вычисление среднего количества проданных билетов в сутки; 2) вывод результатов на экран; 3) вывод результатов в файл;
5	Создать методы класса, реализующие: 1)вычисление среднего времени движения по маршруту; 2) вывод результатов на экран; 3) вывод результатов в файл;	6	Создать методы класса, реализующие: 1)вычисление средневзвешенного курса валюты; 2) вывод результатов на экран; 3) вывод результатов в файл;
7	Создать методы класса, реализующие: 1)вычисление суммарного количества занятых мест; 2) вывод результатов на экран; 3) вывод результатов в файл;	8	Создать методы класса, реализующие: 1)вычисление суммарного количества незанятых мест; 2) вывод результатов на экран; 3) вывод результатов в файл;
9	Создать методы класса, реализующие: 1)вычисление результатов голосования по каждой кандидатуре; 2) вывод результатов на экран; 3) вывод результатов в файл;	10	Создать методы класса, реализующие: 1)вычисление доли студентов, сдавших все зачеты; 2) вывод результатов на экран; 3) вывод результатов в файл;
11	Создать методы класса, реализующие: 1)вычисление среднесуточной температуры; 2) вывод результатов на экран; 3) вывод результатов в файл;	12	Создать методы класса, реализующие: 1)вычисление количества набранных баллов каждой командой; 2) вывод результатов на экран; 3) вывод результатов в файл;

[[В начало](#)]

2.3 Задание 3

Разработать функцию, сопоставляющую результаты текущего и прошлого периода (данные исходных файлов №1 и №2.).

[[В начало](#)]

