**ПРАКТИЧЕСКАЯ РАБОТА №12**

Разработка классов и их элементов в языке С#.

* **ЦЕЛЬ РАБОТЫ**

Научиться разрабатывать классы и их элементы в языке С#.

* **ОБЕСПЕЧЕННОСТЬ ЗАНЯТИЯ**

Оборудование лаборатории и рабочих мест лаборатории:

компьютеры, принтер, сканер, проектор, программное обеспечение общего и профессионального назначения, комплект учебно-методической документации.

Реализация профессионального модуля предполагает обязательную учебную практику.

Оборудование и технологическое оснащение рабочих мест:

1.Комплект ТС компьютера IBM-PC

2.Методические указания для выполнения практических работ

4. Microsoft Visio.

5. Microsoft Visual Studio.

6.Microsoft Office.

* **ЛИТЕРАТУРА**

Хейлсберг А. Язык программирования C#. / Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Классика Computers Science. 4-е изд. — СПб.: Питер, 2012. — 784 с.

* **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.**

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.

Мы не можем напрямую обращаться к закрытым данными 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.

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.

* **ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ**

1. Изучите методические указания и конспект лекций.
2. Проанализируйте задание по своему варианту.
3. Создайте приложение, отвечающее запросу задания.

* **МЕТОДИКА АНАЛИЗА РЕЗУЛЬТАТОВ, ОБРАЗЕЦ ОТЧЕТА.**

1. Отчет должен содержать цель работы.
2. Содержание индивидуального задания.
3. Перечень классов, выбранных для создания приложения.