ЛАБОРАТОРНАЯ РАБОТА 10. МНОГОМОДУЛЬНЫЕ ПРИЛОЖЕНИЯ

1. Цель и содержание

Цель лабораторной работы: научиться проектировать консольные приложения на основе нескольких сборок.

Задачи лабораторной работы:

– научиться управлять несколькими разрабатываемыми проектами в одном решении;

– научиться организовывать взаимодействие нескольких модулей приложения.

2. Формируемые компетенции

Лабораторная работа направлена на формирование следующих компетенций:

– способность к проектированию базовых и прикладных информационных технологий (ПК-11);

– способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12).

3. Теоретическая часть

В терминологии технологической платформы .NET не используется понятие «исполняемый файл». Подобные файлы (*.dll или *.exe) характерны для компиляторов небезопасного кода.

В технологии .NET используется термин «сборка». Сборка – это модуль, в котором хранится компилированный управляемый код. Она похожа на классический исполняемый файл (*.exe) или dll-библиотеку, но имеет важное

свойство – она полностью себя описывает. Сборки содержат метаданные, которые включают в себя сведения о сборке и обо всех определенных внутри нее типах, методах и т.д. Сборка может быть частной (доступной только одному приложению) или разделяемой (доступной любому приложению Windows).

В предыдущих лабораторных работах студентам предлагалось спроектировать консольные приложения, которые представляют собой одномодульные приложения – exe-файлы.

В данной лабораторной работе требуется спроектировать приложение, которое состоит из нескольких сборок (один модуль – файл *.exe, остальные – dll).

4. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место

пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

Для выполнения лабораторной работы спроектируем следующее приложение:

Требуется разработать приложение-опрос. Пользователю последовательно задаются вопросы, за каждый вариант ответа начисляются определенные баллы. После завершения опросы выводится сумма набранных баллов и какое-либо резюме. Такая программа может использоваться для оценки остаточных знаний в форме тестирования, для составления опросов, для самообследования и т.п.

Перед началом создания приложения, то есть перед непосредственным программированием, необходимо определиться: какие объекты и явления предметной области и связи между ними должен выявить программист. Очевидно, что это:

– вопрос;

– ответ (несколько для каждого вопроса);

– балл соответствующий конкретному ответу;

– итоговые резюме или оценки, которые выводятся в зависимости от количества набранных баллов.

Второе, что необходимо сделать осуществить физическую декомпозицию данного приложения, то есть сколько отдельных модулей будет содержать приложение. В данном случае:

– 1-ый модуль в виде файла *.exe для запуска приложения;

– 2-ой модуль в виде файла *.dll, содержащий вопросы и ответы;

– 3-ий модуль в виде файла *.dll, содержащий сообщения о результатах тестирования (опроса).

Декомпозиция, логическая и физическая, завершена. Для реализации прложения необходимо выполнить следующие шаги:

1. Создайте консольное приложение в среде MS VS (проект назовите Quiz). После выполнения всех необходимых действий окно Solution Explorer примет следующий вид:

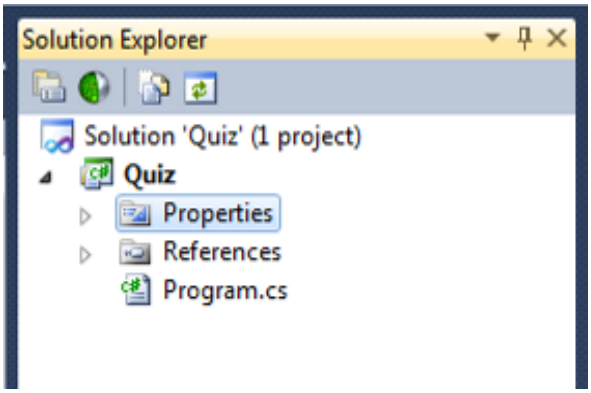


Рисунок 15.1 – Структура одномодульного приложения.

В окне отображено одно решение (Solution) с именем «Quiz» и один проект в рамках решения (с именем «Quiz»). Проект содержит файл кода Program.cs, в котором определена функция Main. Этот проект будет откомпилирован в файл Quiz.exe (то есть файл для запуска).

2. Создадим еще один файл в рамках проекта Quiz, который будет содержать класс Testing, позволяющий реализовывать логику тестирования, то есть вывод вопросов в определенной последовательности, ввод выбора пользователя и т.д. Для этого вызовем контекстное меню проекта (рис. 13.2) и создадим новый файл с именем Testing.cs

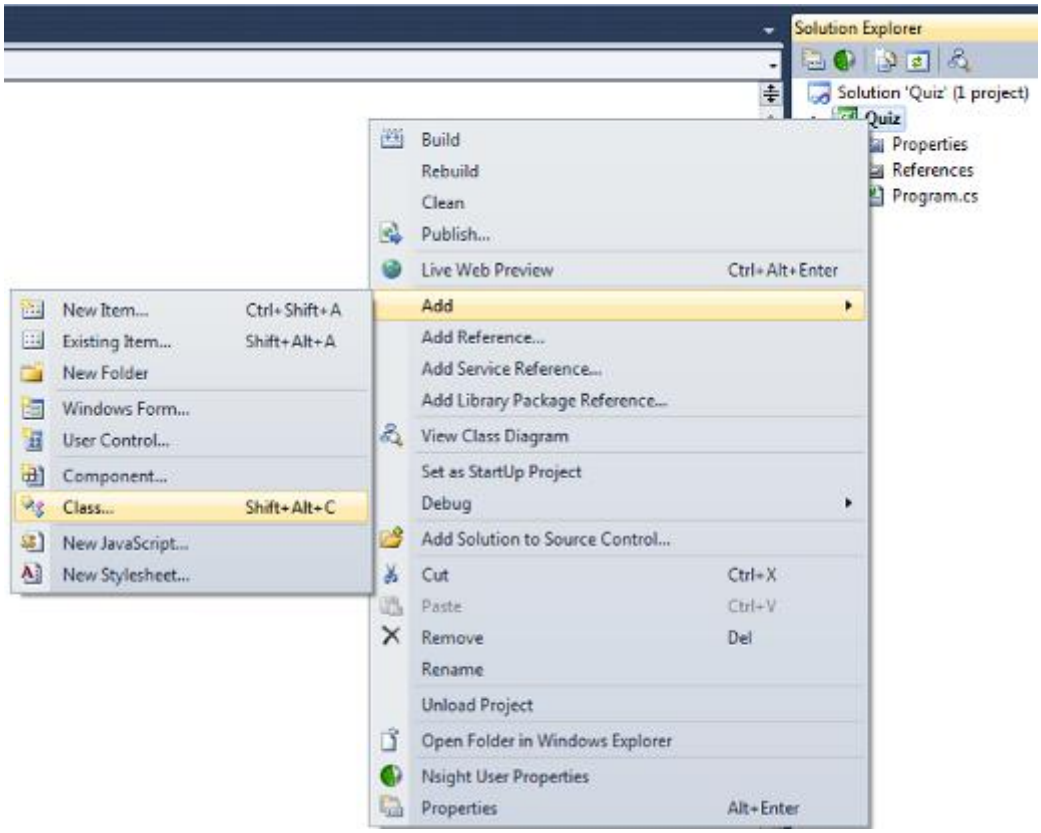


Рисунок 15.2 – Добавления нового класса к проекту.

После выполнения данного действия в окне Solution Explorer состав проекта изменится (рис. 15.3).

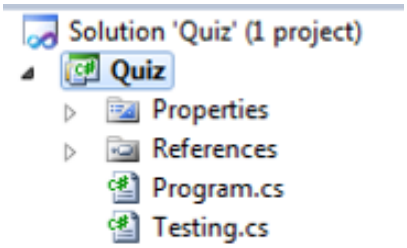


Рисунок 15.3 – Проект с несколькими файлами исходного кода.

На данном этапе класс Testing не содержит никакой логики.

3. Создадим второй модуль в виде dll-библиотеки, которая будет содержать вопросы и ответы. Для этого вызовем команду контекстного меню решения Quiz (рис. 15.4).

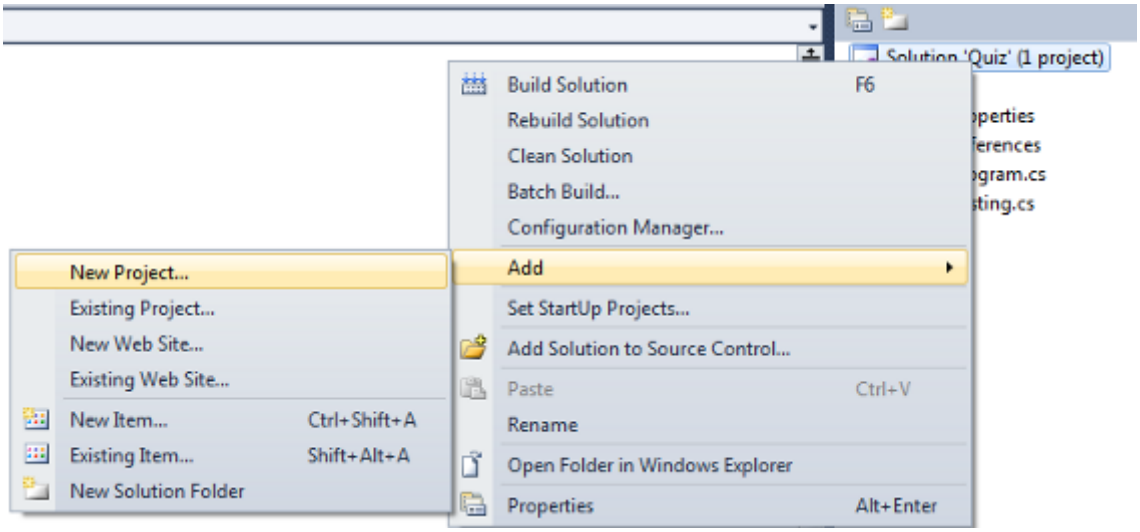


Рисунок 15.4 – Добавление проекта к решению VS.

4. В появившемся диалоговом окне выберем тип проекта «Class Library», в качестве имени проекта укажем «Task» (рис. 15.5)

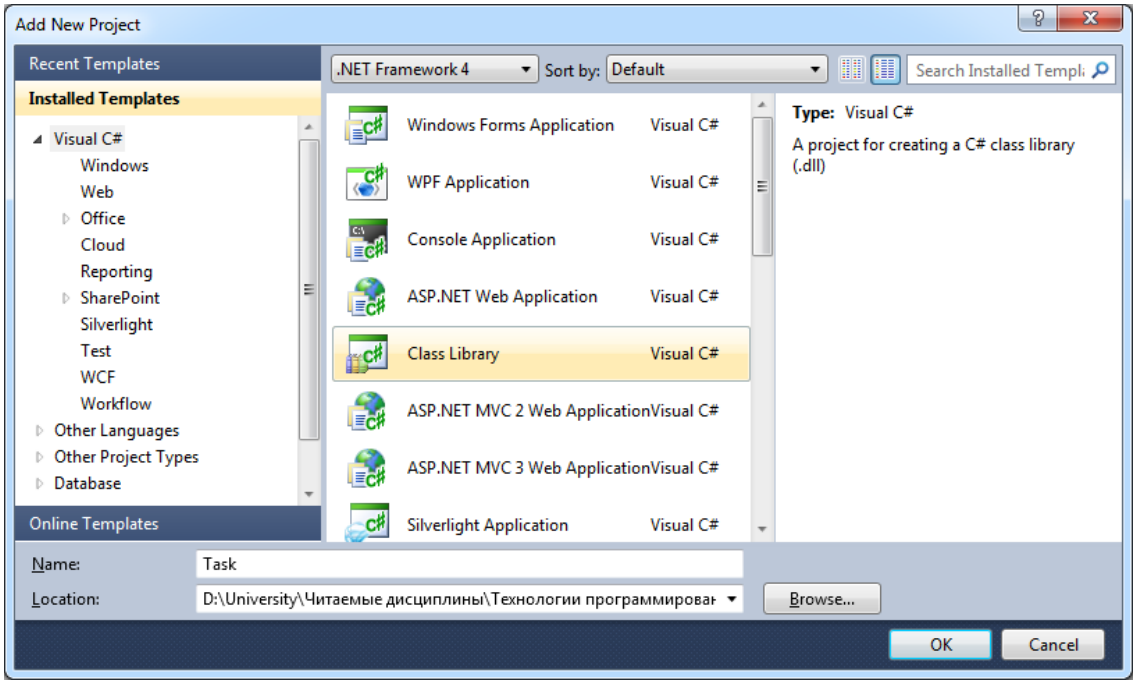


Рисунок 15.5 –Добавление библиотеки классов к решению Quiz.

После добавления нового проекта окно Solution Explorer примет вид, показанный на рис. 15.6

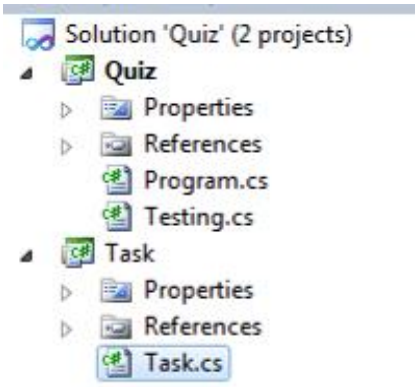


Рисунок 15.6 –Решение с несколькими проектами.

Два проекта в одном решении Quiz, при этом проект Quiz помечен полужирным шрифтом. Это означает, что при вызове команды Run будет запущен именно этот проект. Следует обратить внимание, что файл Task.cs проекта Task содержит только пустой класс без кода. Проект Task в целом не содержит функции Main, то есть не может быть запущен на выполнение.

5. Аналогичным образом добавим к решению еще один проект с именем «Result» и типом «Class Library». Окно Solution Explorer примет вид:

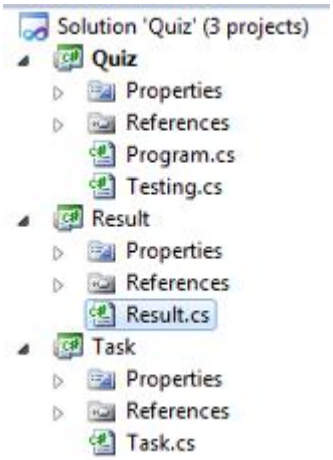


Рисунок 15.7 – Все требуемые проекты, в соответствие с физической декомпозицией, добавлены к решению.

Далее переходим к наполнению классов требуемым функционвалом.

6. Модифицируем файл Task.cs проекта Task следующим образом:

```
namespace Task
{
    // Класс для представления одного тестового задания
    public class SingleTest
    {
        public string Question {get; set;} // вопрос
        public List<string> Answers; // Ответы
        public List<int> Balls; // Баллы за ответы
    }

    // Статический класс для представления списка тестовых заданий
    public static class Task
    {
        // Список отдельных вопросов с ответами
        public static List<SingleTest> AllQuestions;

        // Статический конструктор для инициализации коллекции
        static Task()
        {
            AllQuestions = new List<SingleTest>()
            {
                new SingleTest()
                {
                    Question = "Main - точка входа программы ...",
                    Answers = new List<string>() { "1", "7", "Не знаю" },
                    Balls = new List<int>() { 20, 5, 0 }
                },
                new SingleTest()
                {
                    Question = "Какой оператор не является оператором цикла?",
                    Answers = new List<string>() { "while", "for", "if" },
                    Balls = new List<int>() { -5, 0, 10 }
                },
                new SingleTest()
                {
                    Question = "Как обозначить составной оператор?",
                    Answers = new List<string>() { "{ ... }", "( ... )", "< ... >" },
                    Balls = new List<int>() { 50, 0, 0 }
                },
                new SingleTest()
                {
                    Question = "Что обозначает символ ; ",
                    Answers = new List<string>() { "Пустой оператор", "Конец программы", "Не знаю" },
                    Balls = new List<int>() { 50, 20, 0 }
                }
            };
        }
    }
}
```

Рисунок 15.8 – Классы проекта Task.

Следует обратить внимание на то, что класс Task статический, то есть не требует создания объектов.

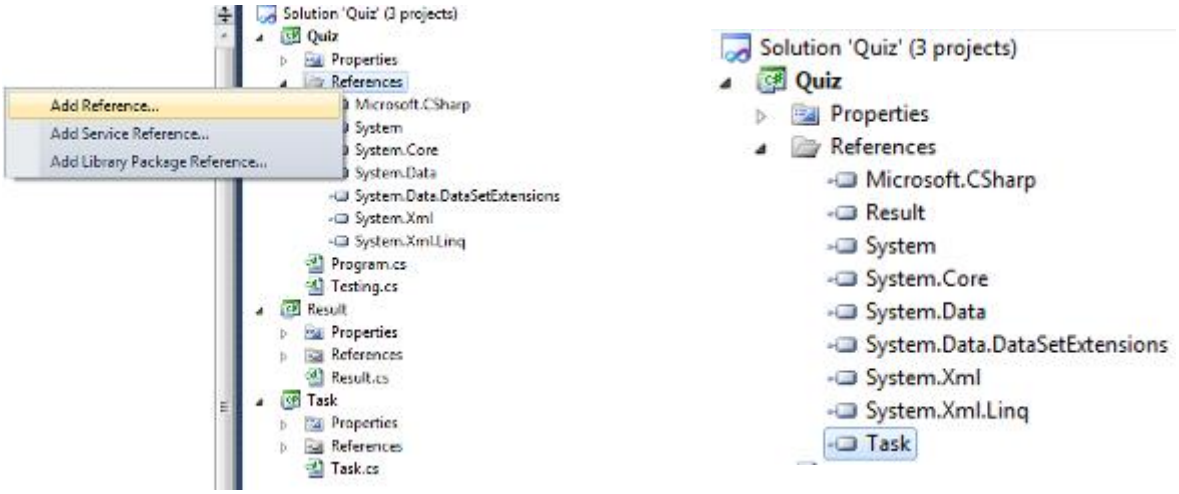
7. Модифицируйте файл Result.cs проекта Result следующим образом:

```
namespace Result
{
    public static class Result
    {
        public static List<string> Messages;

        // Статический конструктор
        static Result()
        {
            Messages = new List<string>()
            {
                "Вы набрали менее 10 баллов! Плохо!",
                "Вы набрали не менее 10 и менее 40 баллов! Нормально!",
                "Вы набрали не менее 40 баллов! Отлично"
            };
        }

        // Метод для возвращения сообщения
        public static string GetMessage(int Ball)
        {
            string mes = "";

            if (Ball < 10)
                mes = Messages[0];
            else if (Ball < 40)
                mes = Messages[1];
            else
                mes = Messages[2];

            return mes;
        }
    }
}
```

Рисунок 15.9 – Класс проекта Result.

Создан один статический файл, для представления результирующих сообщений.

8. Теперь модифицируем класс Tasting проекта Quiz. В своей работе класс будет использовать ранее созданные классы Task и Result, которые находятся в других проектах. Поэтому необходимо установит ссылки на данные проекты. Для этого вызовите команду (рис. 15.10а) контекстного меню папки References проекта Quiz. После этого ссылки на сборки (откомпилированные проекты Task и Result) появятся в списке ссылок Reference (рис. 15.10б).



а *б*

Рисунок 15.10 – Добавление ссылки на проекты: а – команда контекстного меню; б – список ссылок на проекты.

9. Модифицируем файл Testing.cs для реализации логики тестирования (рис. 15.11).

```
class Testing
{
    public string DoTesting()
    {
        int SumBal = 0;
        int ans = 0;

        foreach (Task.SingleTest p in Task.Task.AllQuestions)
        {
            Console.WriteLine("////////////////////////////");
            Console.WriteLine(p.Question);
            Console.WriteLine("////////////////////////////");
            Console.WriteLine("1. "+p.Answers[0]);
            Console.WriteLine("2. " + p.Answers[1]);
            Console.WriteLine("3. " + p.Answers[2]);
            Console.Write("Выберите номер ответа > ");
            ans = Convert.ToInt32(Console.ReadLine());
            SumBal += p.Balls[ans-1];
        }

        return Result.Result.GetMessage(SumBal);
    }
}
```

Рисунок 15.11 – Класс Testing.

10. Модифицируем функцию Main:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Quiz
{
    class Program
    {
        static void Main(string[] args)
        {
            Testing newTest = new Testing();

            Console.WriteLine("\n\n"+newTest.DoTesting());

            Console.ReadKey();
        }
    }
}
```

Рисунок 15.12 – Функция Main.

Запустите полученное приложение. Внимательно проанализируйте код приложения. Попробуйте изменить вопросы и их количество.

Затем, выполните индивидуальное задание.

Индивидуальное задание.

Спроектируйте многомодульное приложение (3 модуля), реализующее поставленную задачу. Перед выполнением приложения необходимо выполнить декомпозицию задачи и выявить основные объекты предметной области.

Вариант	Структура данных
1, 6, 11, 16, 21	Приложение осуществляет сложение обыкновенных дробей, то есть чисел $\frac{1}{3}$, $\frac{7}{12}$ и т.д. Требуется реализовать только сложение. Исходные слагаемые пользователь вводит с клавиатуры.
2, 7, 12, 17, 22	Приложение осуществляет умножение обыкновенных дробей, то есть чисел $\frac{1}{3}$, $\frac{7}{12}$ и т.д. Требуется реализовать только умножение. Исходные множители пользователь вводит с клавиатуры.

Вариант	Структура данных
3, 8, 13, 18, 23	Приложение осуществляет деление обыкновенных дробей, то есть чисел $\frac{1}{3}$, $\frac{7}{12}$ и т.д. Требуется реализовать только деление. Исходные дроби пользователь вводит с клавиатуры.
4, 9, 14, 19, 24	Реализуйте приложение, осуществляющее сокращение обыкновенной дроби. Дробь пользователь вводит с клавиатуры.
5, 10, 15, 20, 25	Реализуйте программу для выполнения операций над комплексными числами: сложение, умножение, вычисление модуля комплексного числа. Исходные комплексные числа пользователь вводит с клавиатуры.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое статический класс?

2. Что такое сборка?

3. Как определить проект по умолчанию в многомодульном решении?

4. Какими способами можно разместить в файлах определение класса?

5. Какой проект начнет выполняться первым если несколько из них в одном решении содержат функцию Main?

6. Что необходимо сделать для того, чтобы появилась возможность использовать классы, определенные в другом проекте?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [1-3].

ЛАБОРАТОРНАЯ РАБОТА 11. ОПЕРАЦИИ КЛАССОВ. ПЕРЕГРУЗКА ОПЕРАЦИЙ.

1. Цель и содержание

Цель лабораторной работы: научиться осуществлять перегрузку операторов относительно пользовательских типов.

Задачи лабораторной работы:

– изучить операции, подлежащие перегрузке;

– получить практические навыки перегрузки операторов.

2. Формируемые компетенции

Лабораторная работа направлена на формирование следующих компетенций:

– способность к проектированию базовых и прикладных информационных технологий (ПК-11);

– способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12).

3. Теоретическая часть

2.1 Операции. Большинство операций пришло в C# из языков C, C++:

Категория	Операции
Арифметические	+, -, *, /, %
Логические	&, |, ^, ~, &&, ||, !
Конкатенация строк	+
Инкремент и декремент	++, --
Сравнение	<, >, >=, >=, ==, !=
Присвоение	+=, -=, /=, *=, %=, =, &=, |=, ^=, <<=, >>=