**Министерство науки и высшего образования Российской Федерации**
федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
(ФГАОУ ВО «СПбПУ»)
**Институт среднего профессионального образования**

# ОТЧЕТ

**по учебной практике (по профилю специальности)**

по профессиональному модулю ПМ.04 «Сопровождение и обслуживание программного обеспечения компьютерных систем»
(код и наименование)

Специальность 09.02.07 «Информационные системы и программирование»
(код и наименование специальности)

Студент(ка) II курса 24290907/3091 группы

Цветкова Анастасия Романовна
(ФИО полностью)

Место прохождения практики: ФГАОУ ВО СПбПУ Петра Великого Институт СПО, учебно-вычислительный центр, пр. Энгельса д.23
(наименование и адрес организации)

Период прохождения практики

с «08» декабря 2025 г. по «27» декабря 2025 г.

Руководитель практики от учебной организации _____ (подпись) Курылева А. А. (расшифровка подписи)

Итоговая оценка по практике _____
М.П.

Санкт-Петербург
2025г.

СОГЛАСОВАНО
Председатель ПЦК
_____/_____
«_____» _____ 202__ г.

# ЗАДАНИЕ
## на учебную практику (по профилю специальности)

по профессиональному модулю ПМ.04 «Сопровождение и обслуживание программного обеспечения компьютерных систем»
(код и наименование)

Специальность 09.02.07 «Информационные системы и программирование»
(код и наименование специальности)

Студент(ка) II курса 24290907/3091 группы

Цветкова Анастасия Романовна
(ФИО полностью)

Место прохождения практики: ФГАОУ ВО СПбПУ Петра Великого Институт СПО, учебно-вычислительный центр, пр. Энгельса д.23
(наименование и адрес организации)

Период прохождения практики

с «08» декабря 2025 г. по «27» декабря 2025 г.

**Виды работ, обязательные для выполнения** (переносится из программы, соответствующего ПМ):

1. Персонализация интегрированной среды разработки Visual Studio Community 2022
2. Отладка в IDE Visual Studio Community 2022
3. Обеспечение качества кода
4. Упаковка приложения

**Индивидуальное задание:** _____

Задание выдал с «8» декабря 2025 г. _____ Курылева А. А.
(подпись) (Ф.И.О.)

Задание получил с «8» декабря 2025 г. _____ _____
(подпись) (Ф.И.О.)

**Министерство науки и высшего образования Российской Федерации**
федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
(ФГАОУ ВО «СПбПУ»)
**Институт среднего профессионального образования**

# ДНЕВНИК
**прохождения учебной практики**
**(по профилю специальности)**

по профессиональному модулю ПМ.04 «Сопровождение и обслуживание программного обеспечения компьютерных систем»
(код и наименование)

Специальность 09.02.07 «Информационные системы и программирование»
(код и наименование специальности)

Студент(ка) II курса 24290907/3091 группы

Цветкова Анастасия Романовна
(ФИО полностью)

Место прохождения практики: ФГАОУ ВО СПбПУ Петра Великого Институт СПО, учебно-вычислительный центр, пр. Энгельса д.23
(наименование и адрес организации)

Период прохождения практики
с «08» декабря 2025 г. по «27» декабря 2025 г.

Руководитель с места прохождения практики _____ (подпись) Курылева А. А. (расшифровка подписи)

Санкт-Петербург
2025 г.

## Содержание дневника

| Дата | Виды выполненных работ и заданий по программе практики | Подпись руководителя практики |
|---|---|---|
| **1** | **2** | **3** |
| 8.12.25 | Настройка меню и панели инструментов | |
| 9.12.25 | Параметры текстового редактора | |
| 10.12.25 | Создание кода и текстового шаблона | |
| 11.12.25 | Навигация по коду с помощью отладчика | |
| 12.12.25 | Использование точек останова | |
| 13.12.25 | Управление исключениями с помощью отладчика | |
| 15.12.25 | Использование файлов дампа<br>Использование средств профилирования | |
| 16.12.25 | Тестирование в IDE Visual Studio Community 2022 | |
| 17.12.25 | Документирование кода с помощью XML-комментариев | |
| 18.12.25 | Изменение кода в соответствии с соглашением о кодировании | |
| 19.12.25 | Анализ качества кода | |
| 20.12.25 | Основы системы контроля версий Git | |
| 22.12.25 | Технологические подходы программирования | |
| 23.12.25 | Методология программирования | |
| 24.12.25 | Работа с реестром ОС Windows | |
| 25.12.25 | Упаковка классического приложения вручную | |
| 26.12.25 | Упаковка приложения с помощью Visual Studio Package Installer | |
| 27.12.25 | Создание виртуальной машины | |

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

# Практическая работа №1 «Настройка внешнего вида и функциональности среды разработки Visual Studio»

**Цель:** Освоить настройку интерфейса и функциональности Visual Studio для комфортной работы.

2 вариант (28) - Вам даны два целых числа. Напишите программу, выводящую сумму целых чисел.

## 1.1. Параметры текстового редактора

В настройках текстового редактора необходимо настроить автоматическое форматирование при вводе } и ;. Для этого необходимо в разделе «Средства → Параметры → Текстовый редактор» выбрать язык программирования, открыть общие настройки и установить флаги на соответствующих пунктах. Эти настройки позволят автоматически форматировать код во время ввода указанных символов (Рисунок 1).

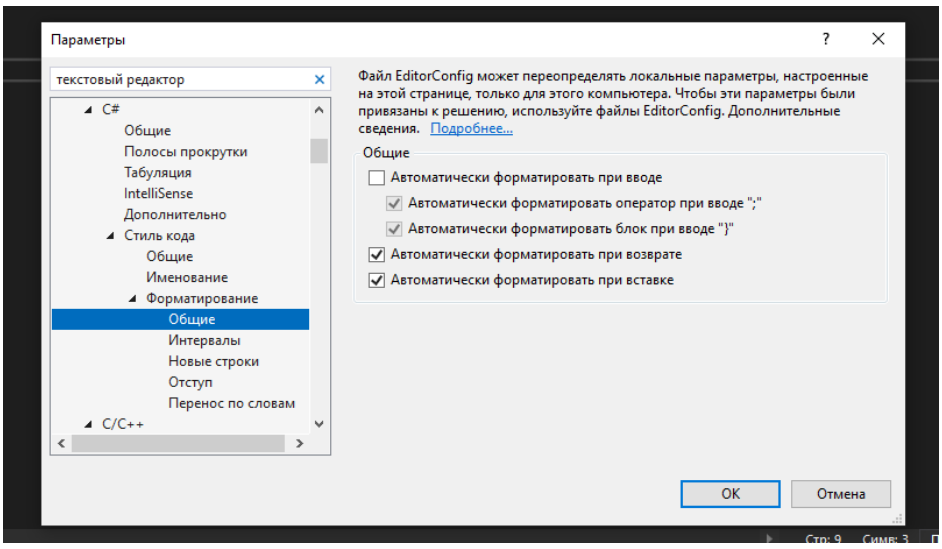


Рисунок 1 - Автоматическое форматирование при вводе } и ;.

Для изменения интервалов в разделе «Средства → Параметры → Текстовый редактор» выбираем раздел «Форматирование» устанавливаем флаг на пункт «Вставлять пробел между скобками со списком параметров» (Рисунок 2).

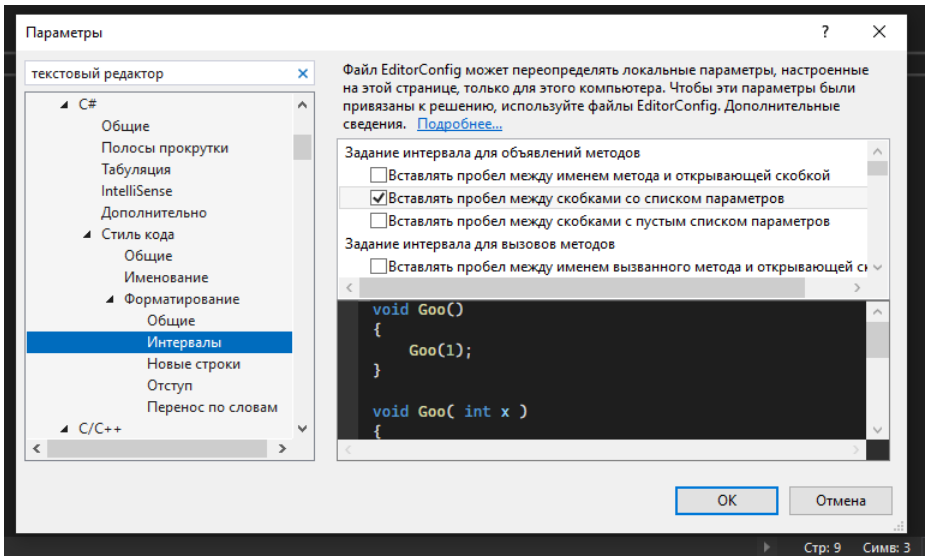


Рисунок 2 - Изменение интервалов

Для изменения нумерации строк в разделе «Средства → Параметры → Текстовый редактор» выбираем раздел «Общие» устанавливаем флаг на пункт «Номера строк» (Рисунок 3).

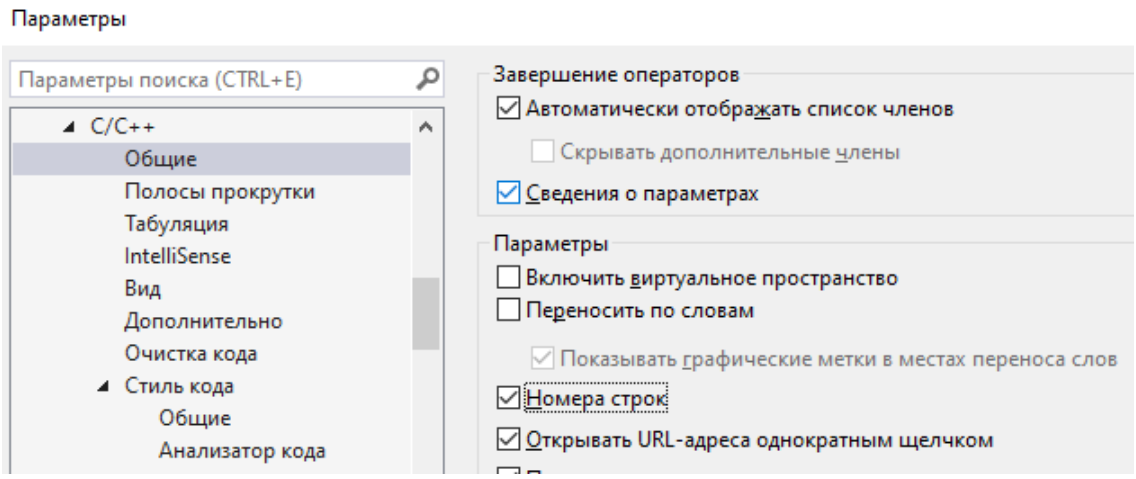


Рисунок 3 - Номера строк

## 1.2. Изменение шрифтов и цветов

В настройках окружения необходимо изменить настройки для шрифта и цвета. Для этого необходимо в разделе «Средства → Параметры → Окружение → Шрифты и цвета» установить необходимые параметры. (Рисунок 4-7).

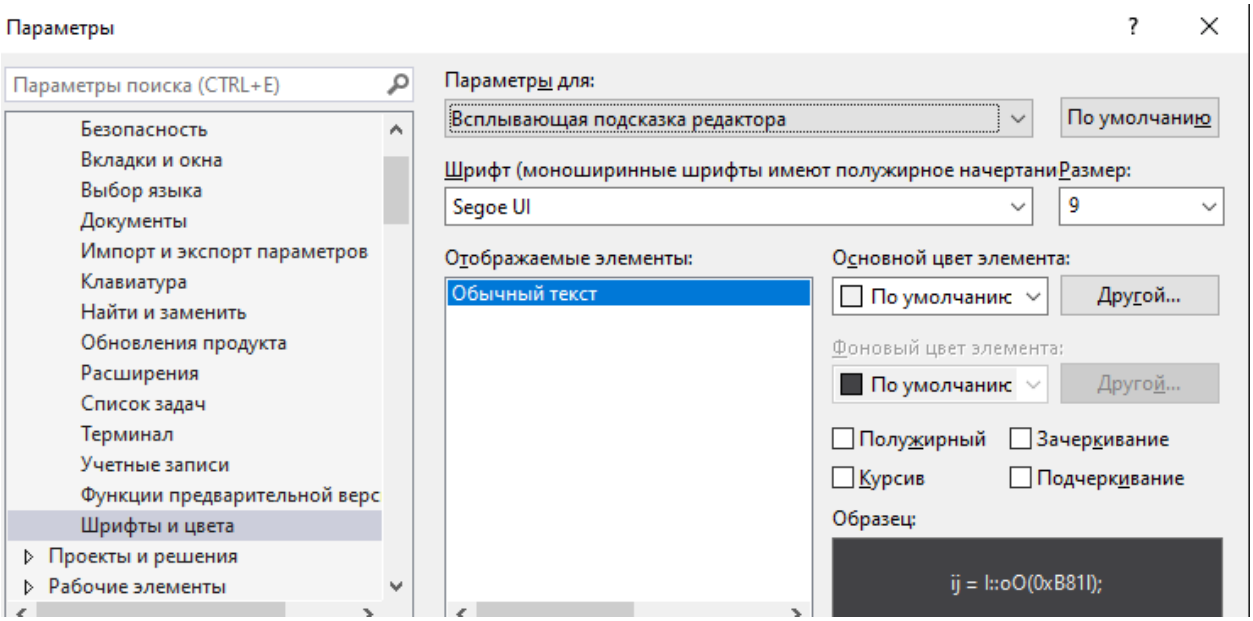


Рисунок 4 - всплывающая подсказка

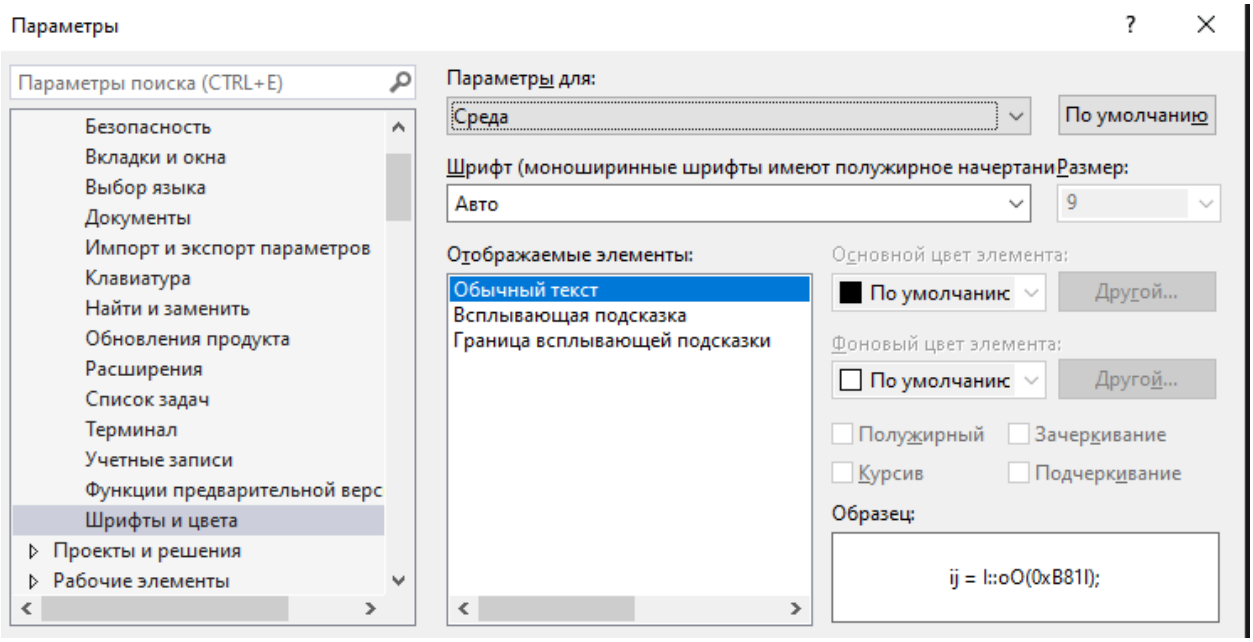


Рисунок 5 – Среда

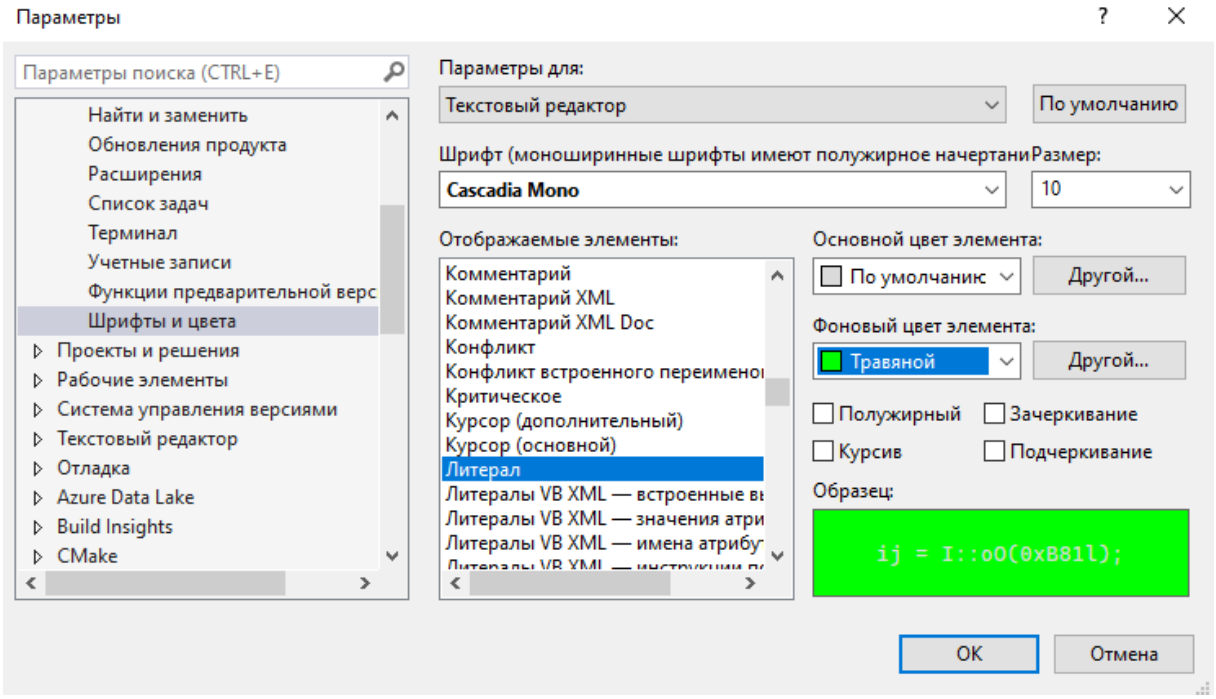


Рисунок 6 - Строковой литерал

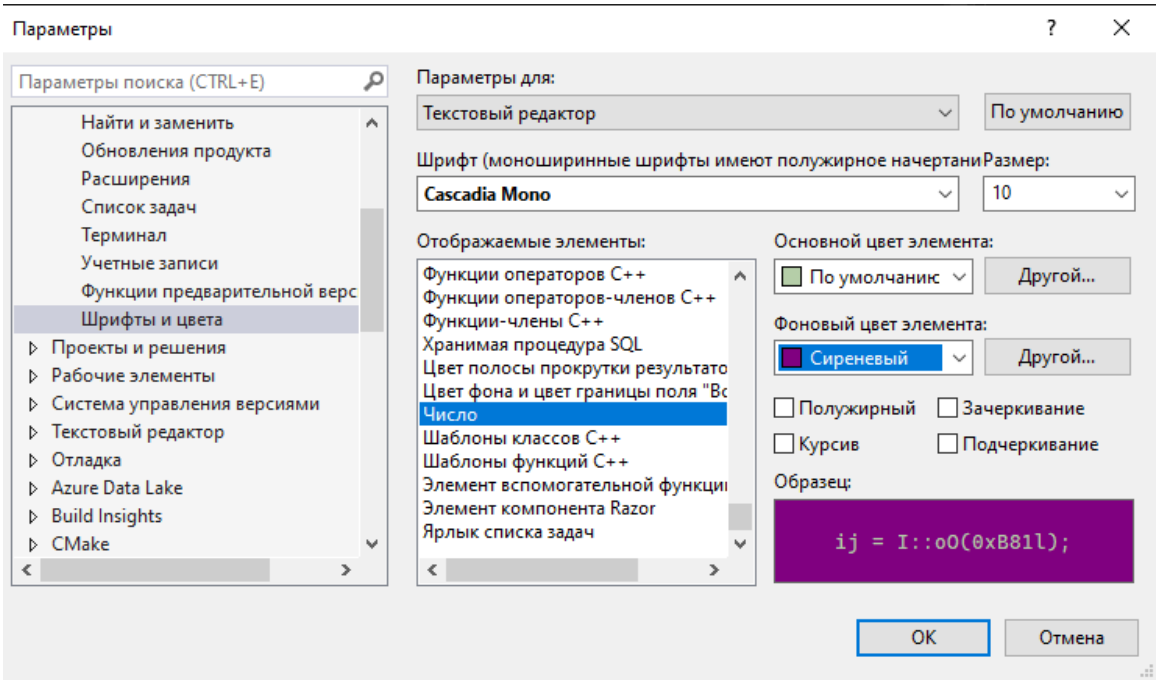


Рисунок 7 – число

### 1.3. Настройка меню и панели инструментов

Необходимо настроить меню и панели инструментов. Для этого необходимо в разделе «Средства → Настройка». Создать свою панель инструментов со своей фамилией. (Рисунок 8).

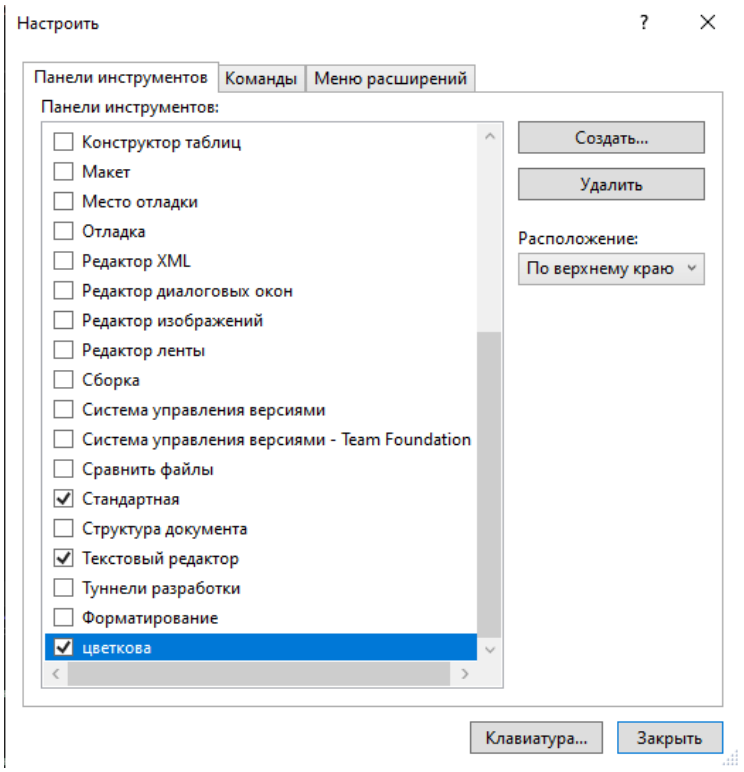


Рисунок 8 – панель инструментов

Необходимо добавить 5 команд. Для этого необходимо открыть «Средства → Параметры → Окружение → Клавиатура». Выбрать 5 команд. (Рисунок 9-13).

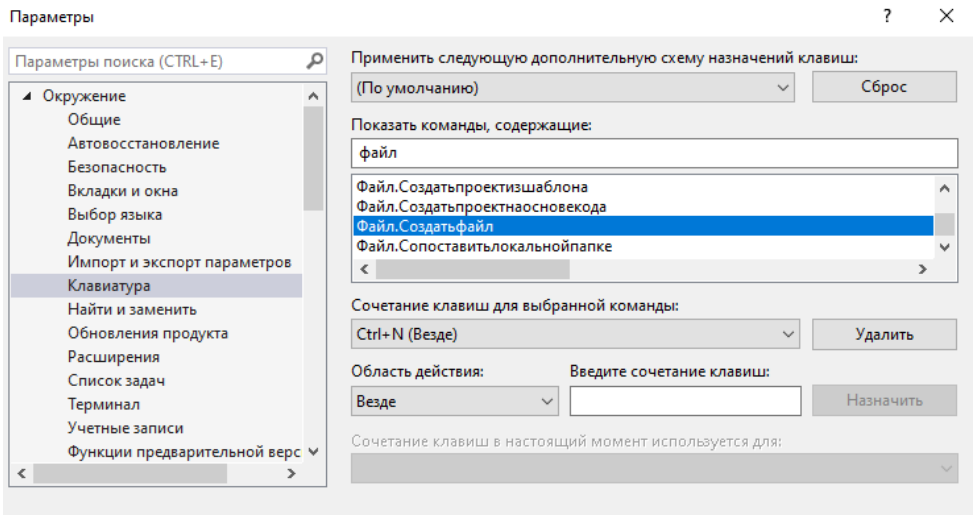


Рисунок 9 - Файл.СздатьФайл

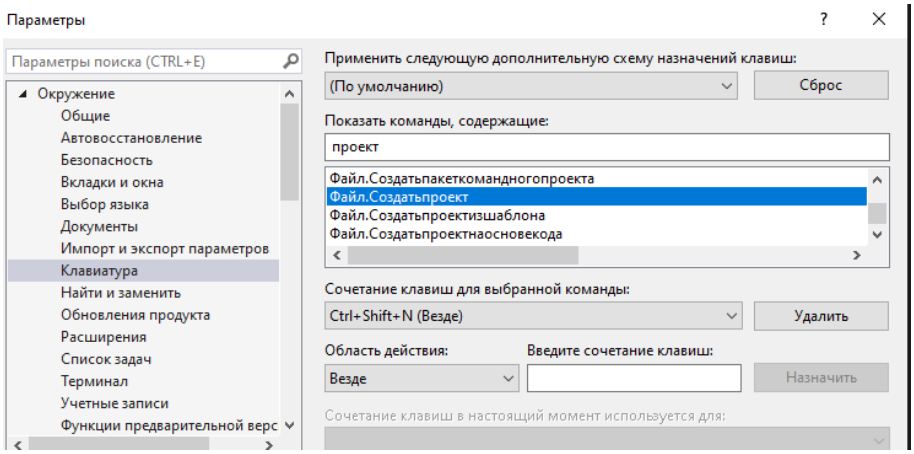


Рисунок 10 - Файл.Создатьпроект

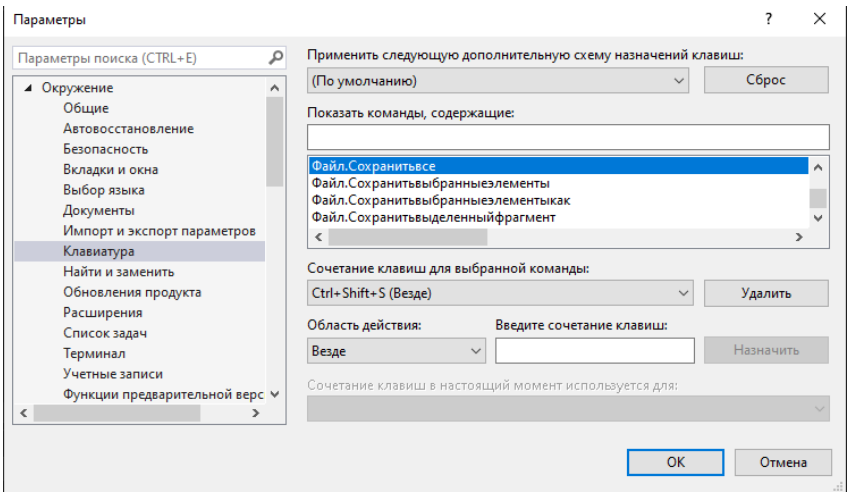


Рисунок 11 - Файл.Сохранитьвсе

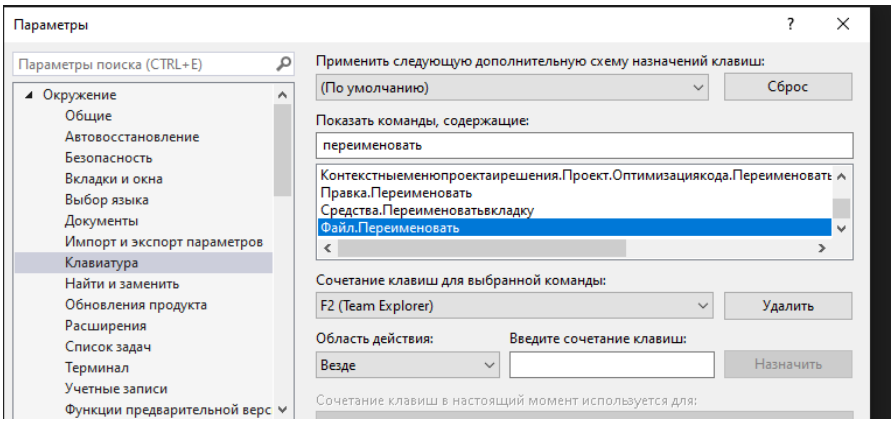


Рисунок 12 - Переименовать

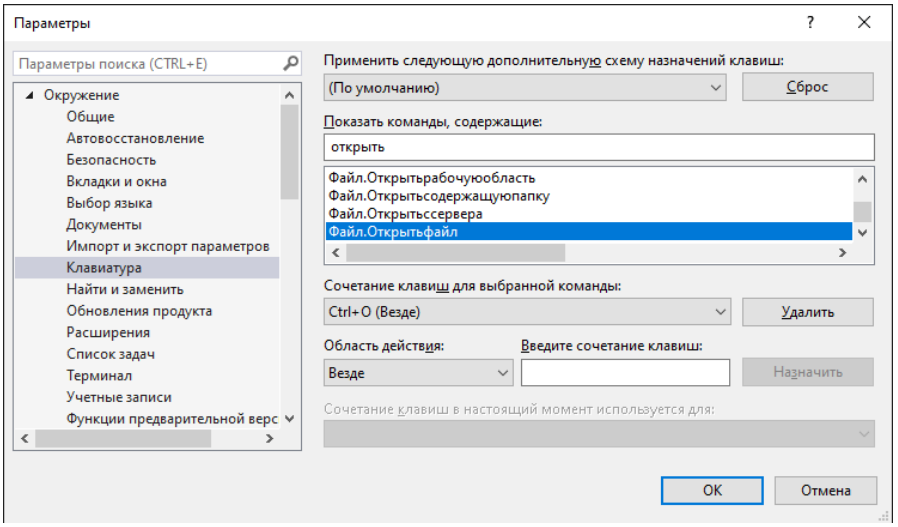


Рисунок 13 - Файл.Открытьфайл

## 1.4. Настройка макетов окон

Необходимо настроить макет окон. Для этого необходимо в разделе «Вид → Другие окна». Настроить свой макет и назвать его своей фамилией. Для этого нужно «Окно → Сохранить макет окна». (Рисунок 14).

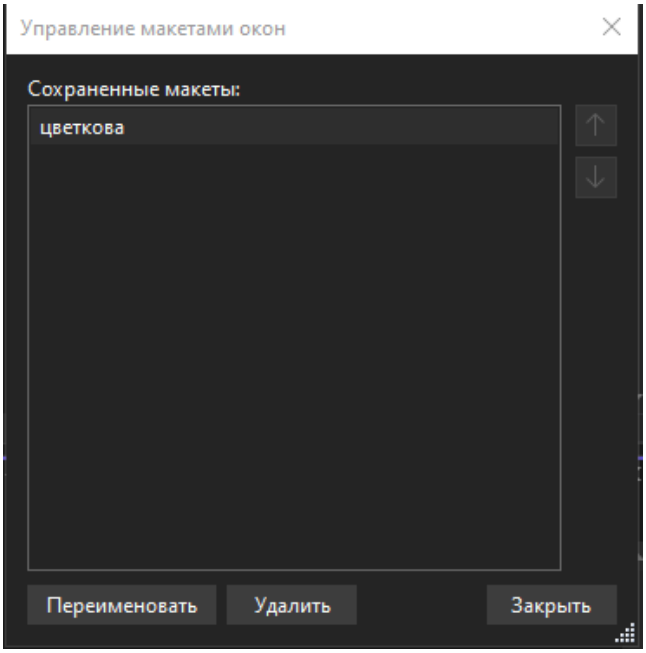


Рисунок 14- Управление макетами окон

### 1.5. Экспорт настроек

Необходимо настроить экспорт настроек. Для этого необходимо «Средства → Импорт и экспорт параметров». Экспортировать текущие настройки, сбросить и импортировать ранее экспортированные настройки. (Рисунок 15-17).

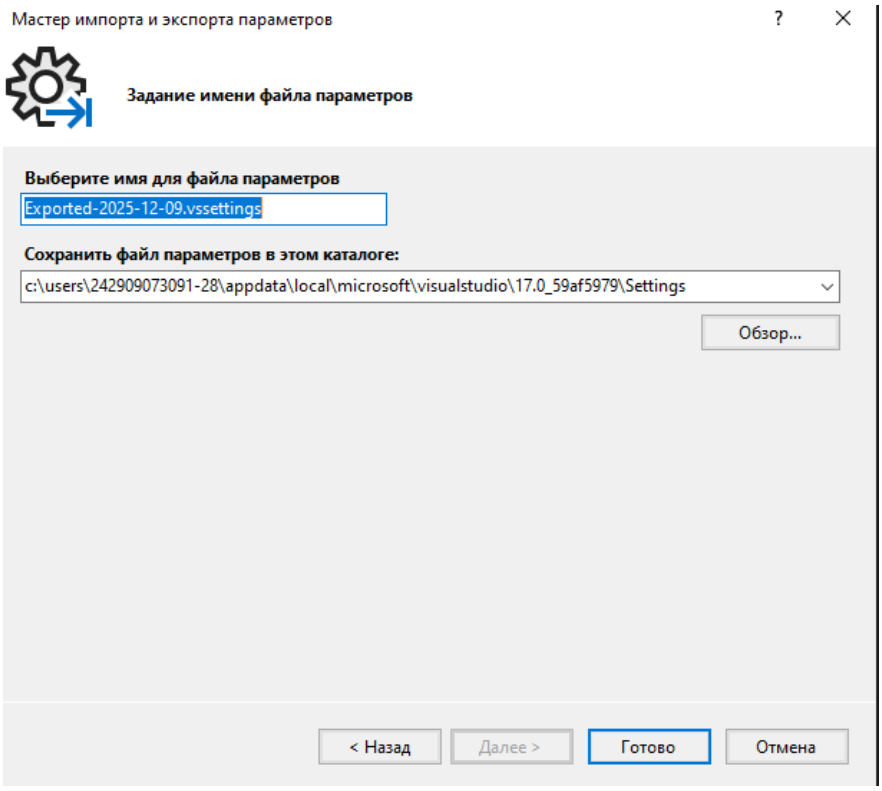


Рисунок 15 - экспорт настроек

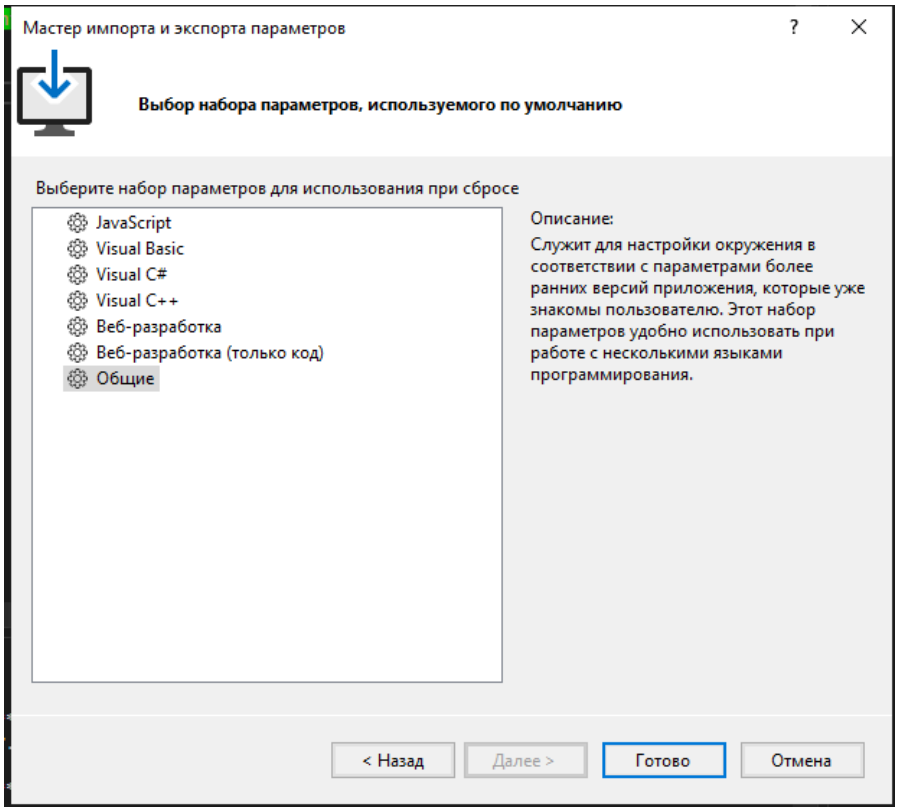


Рисунок 16 - сброс настроек

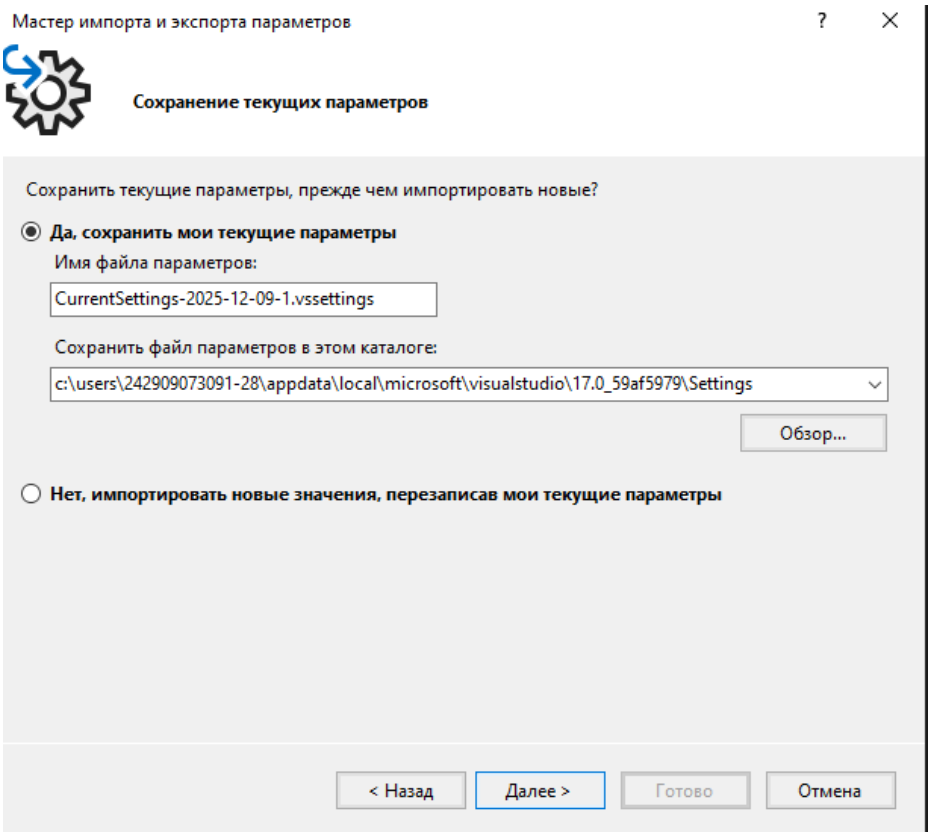


Рисунок 17 - импорт ранее экспортированных параметров

Код программы

```
#include <iostream>
using namespace std;
int main () {
    setlocale (0, "");
    int number1;
    int number2;
    cout << "Введите первое целое число: ";
    cin >> number1;
    cout << "Введите второе целое число: ";
    cin >> number2;
    int sum = number1 + number2;
    cout << "Сумма чисел " << number1 << " и " << number2 << "
равна: " << sum << endl;

    return 0;
```

«Отладка в Visual Studio» (Рисунок 18).

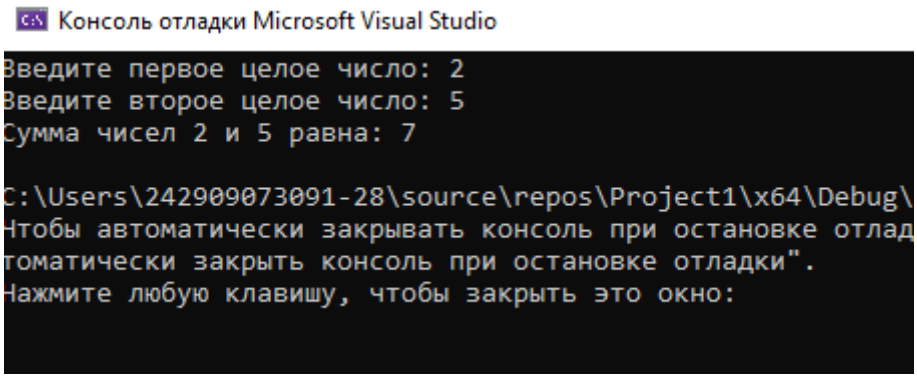


Рисунок 18 - откладка

## Практическая работа №2. «Отладка в Visual Studio»

**Цель:** освоить базовые и продвинутые техники отладки.

### 2.1. Навигация по коду с помощью отладчика

Необходимо создать консольное приложение на C++, содержащее цикл у установить точку останова на строке с циклом. Для этого на строке внутри цикла For за нумерацией строк нажать один раз или нажать F9. (Рисунок 19).

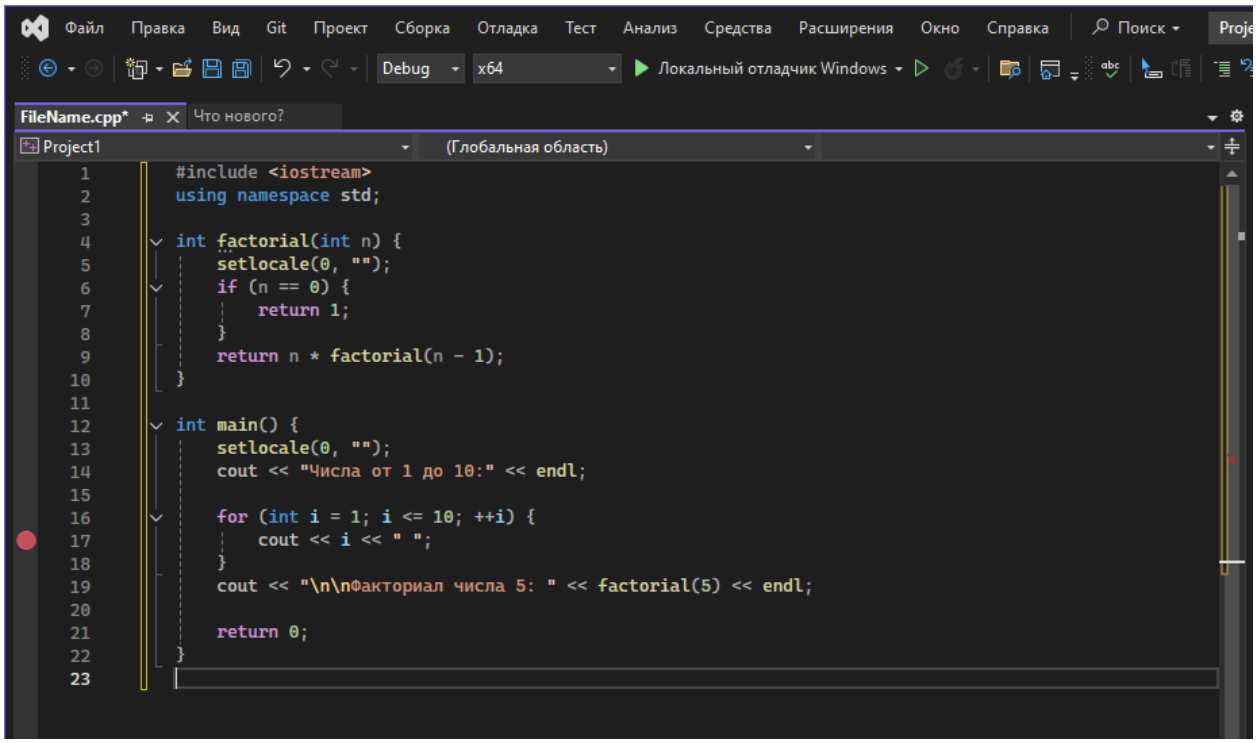


Рисунок 19 - точка останова

Далее запустить откладку F5. Программа запустится и остановится на строке с циклом. В окне локальных переменных можно увидеть, что переменная i еще не создана (она создастся при первом проходе цикла). (Рисунок 20).

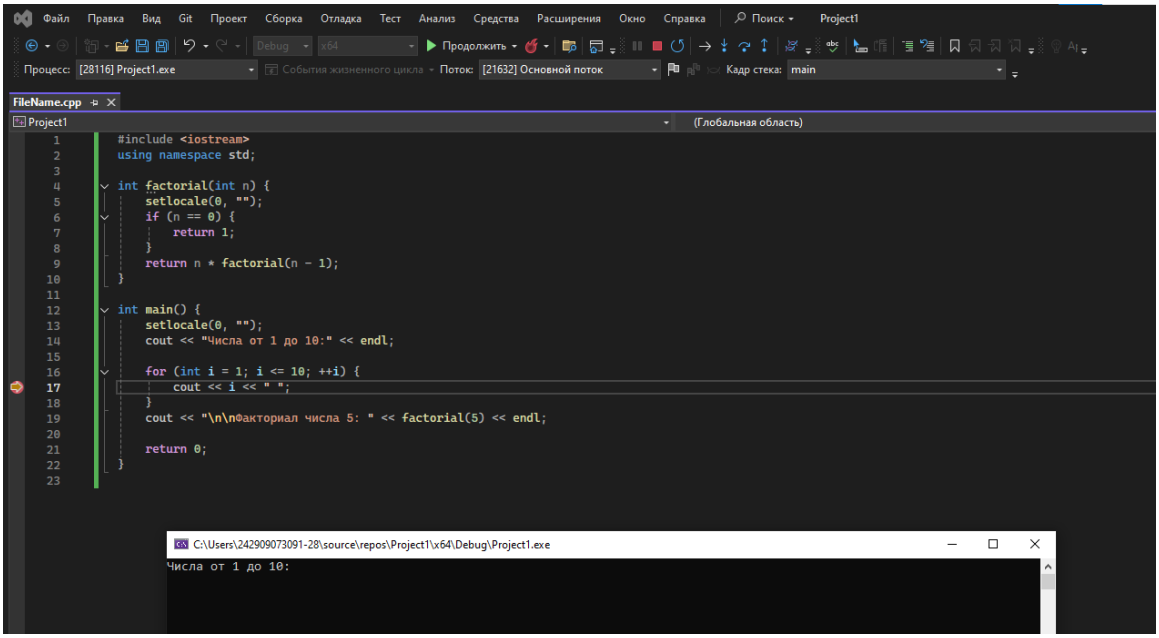


Рисунок 20 - F5

Запустить откладку F10. Выполняет текущую строку без заглядывания внутрь функций. строка выполняется, выводится 1, и отладчик переходит на строку for (int i = 1; ...). (Рисунок 21).

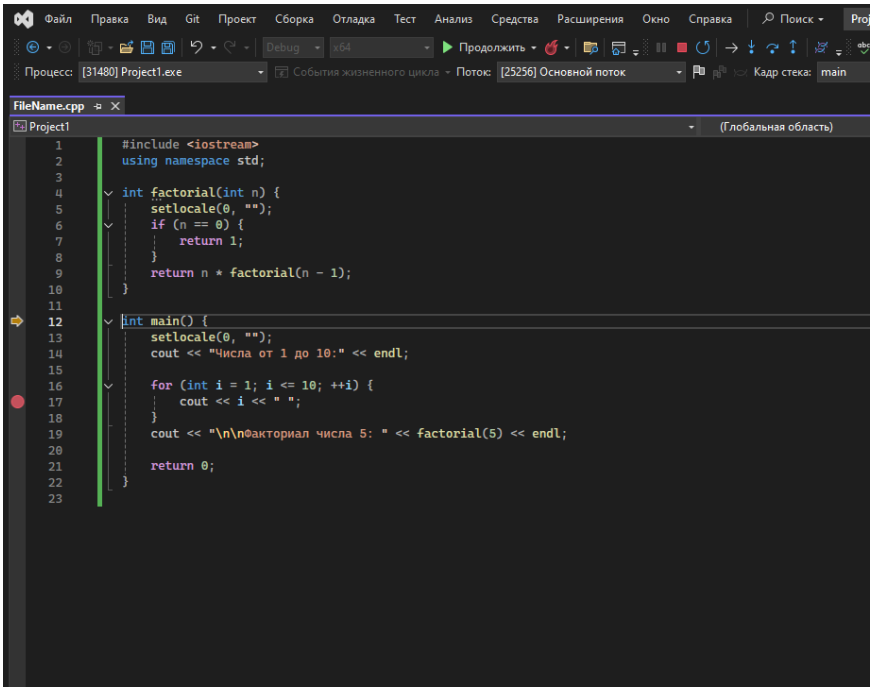


Рисунок 21 - F10

Запустить откладку F11. Заходит внутрь вызываемой функции, если текущая строка содержит вызов функции. Отладчик переходит на первую строку вызываемой функции. (Рисунок 22). Пример в нашем коде:

1. В main() на строке:

cout << "\n\nФакториал числа 5: " << factorial(5) << endl;

2. Нажмите F11 → отладчик войдет в функцию `factorial(int n)` и остановится на строке:

if (n == 0) {

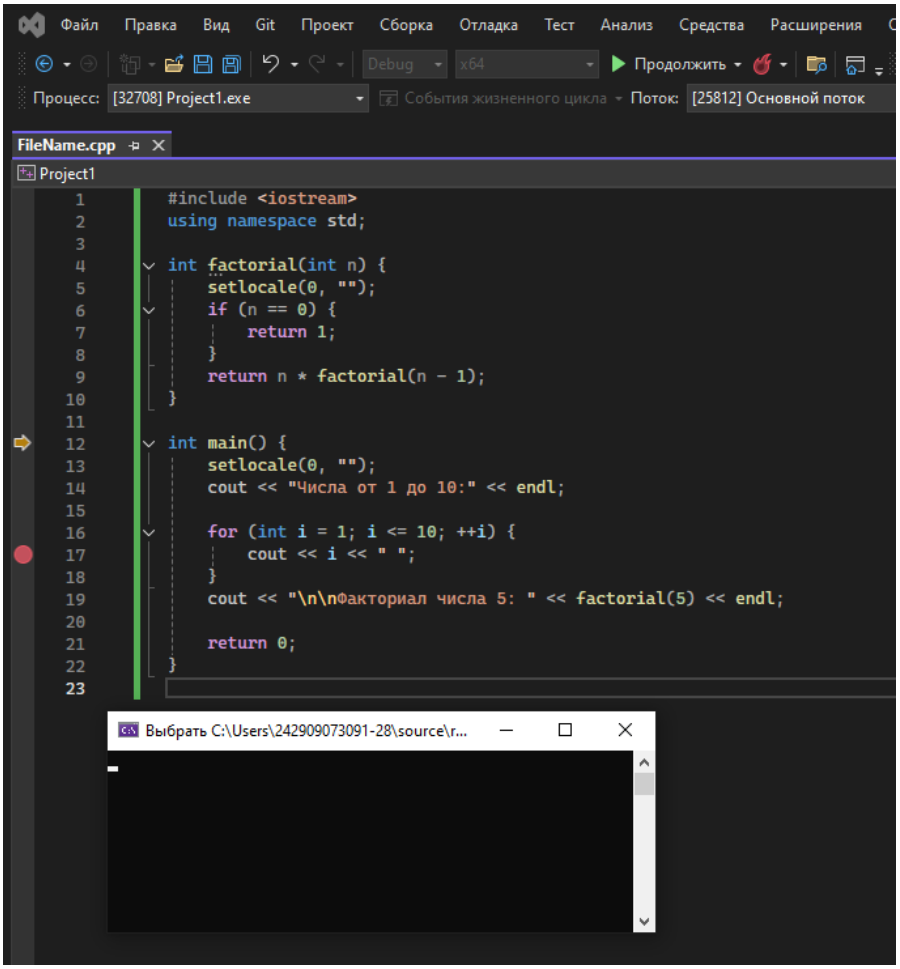


Рисунок 22 - F11

Запустить откладку Shift+F11. Выполняет текущую функцию до возврата и останавливается на строке, которая вызвала эту функцию. Очень полезно, когда вы зашли внутрь функции (через F11) и хотите быстро вернуться обратно. (Рисунок 23).

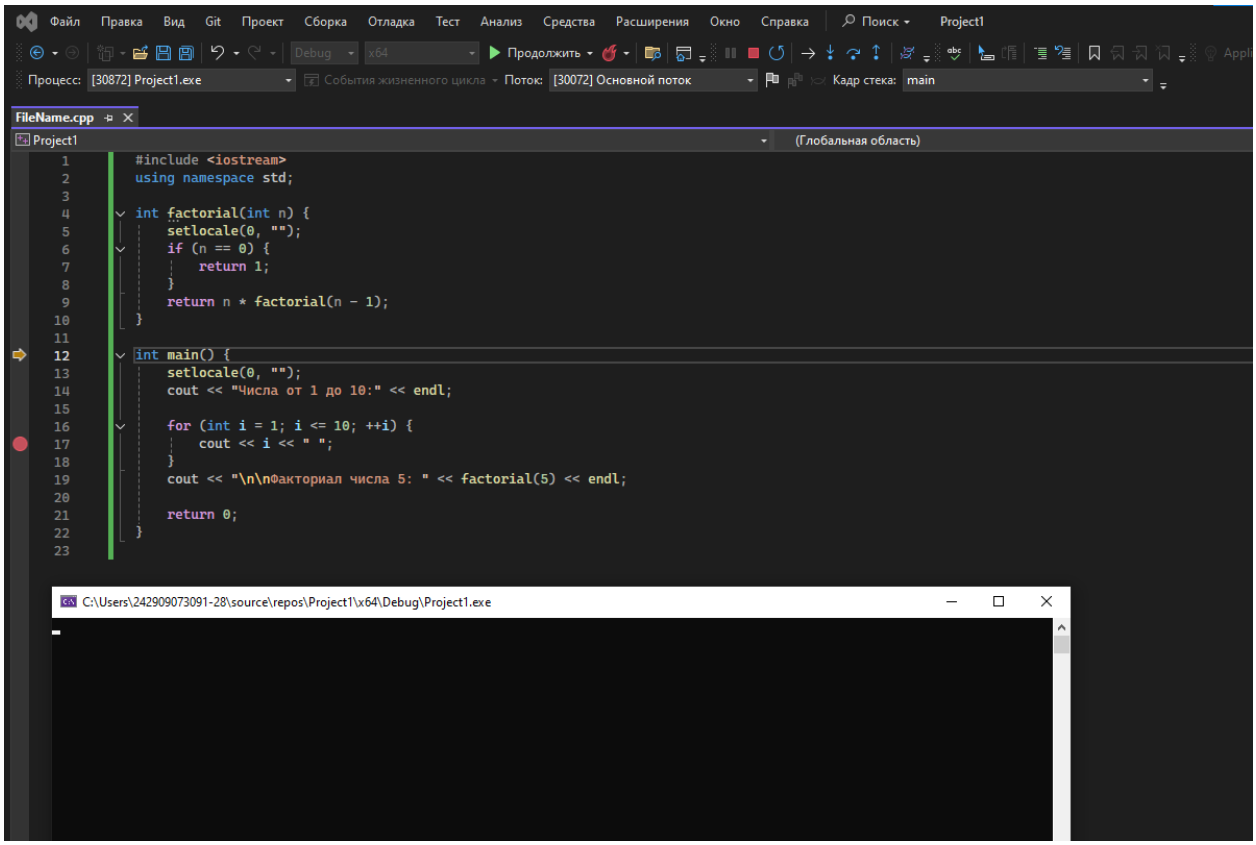


Рисунок 23 - Shift+F11

Шаг с обходом (F10) выполняет текущую строку. Не заходит внутрь вызываемых функций. Когда не нужно разбирать работу других функций. Шаг с заходом (F11) заходит внутрь вызываемой функции. Чтобы детально изучить работу функции. Шаг с выходом (Shift+F11) выходит из текущей функции, выполняет ее до возврата. Когда вы внутри функции и хотите вернуться обратно.

Установить условную точку останова, например, для остановки при значении переменной. Для этого необходимо щёлкнуть правой кнопкой мыши на точке останова в строке цикла. Выбрать «Условие» и в открывшемся окне ввести i == 5. (Рисунок 24).

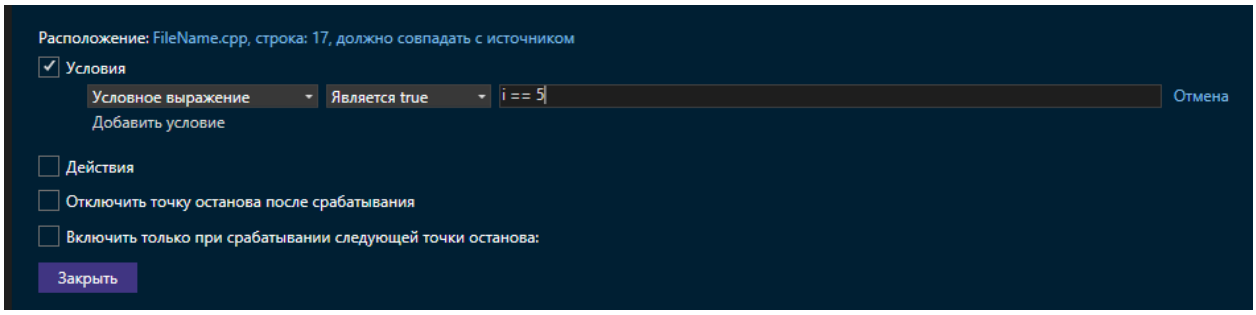


Рисунок 24- условие

Видимые – показывает переменные, используемые в текущей и предыдущей строках кода. Показывает значения всех переменных и выражений, имеющихся в текущей выполняющейся строке кода или в предыдущей строке. В C++ в окне отображаются переменные в трёх предыдущих строках кода. (Рисунок 25).

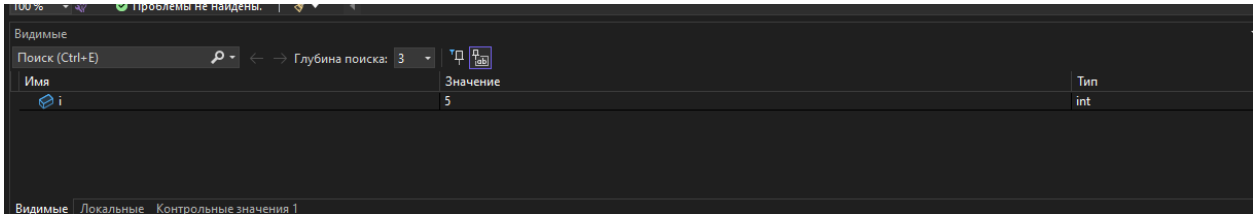


Рисунок 25 - Видимое окно

Окно «Локальные» показывает переменные, которые находятся в текущей области. (Рисунок 26).

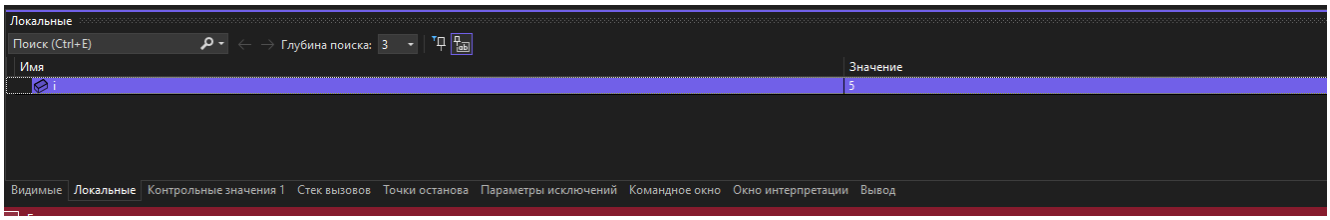


Рисунок 26 - Локальное окно

Окно «Контрольные значения» позволяет настраивать список переменных и выражений, за которыми нужно наблюдать. В отличие от других окон переменных, в окне «Контрольное значение» всегда отображаются просматриваемые переменные (они выделяются серым цветом, когда находятся вне области действия). (Рисунок 27).

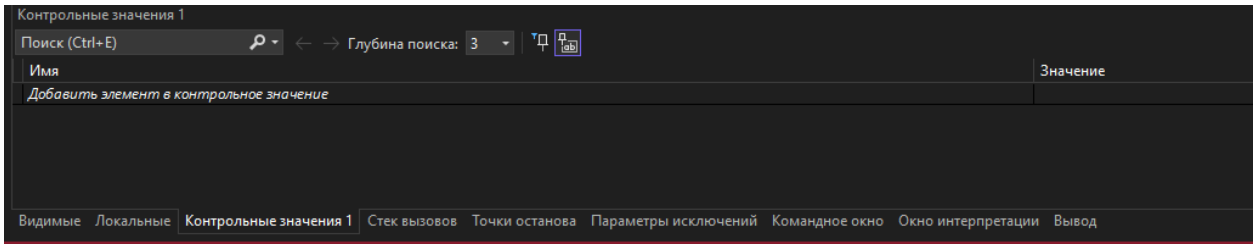


Рисунок 27- Контрольные значения

Окно «Быстрая проверка» отображается одна переменная за раз. Его следует закрыть до того, как можно будет продолжить отладку. предназначено для вычисления текущего значения переменной или выражения, распознанного отладчиком, либо содержимого регистра. (Рисунок 28).

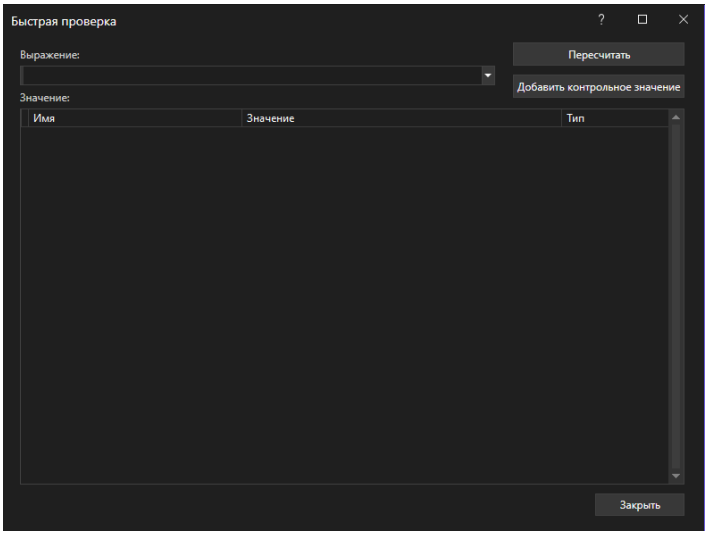


Рисунок 28- Быстрая проверка

```cpp
#include <iostream>
using namespace std;

int factorial(int n) {
    setlocale(0, "");
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}

int main() {
    setlocale(0, "");
    cout << "Числа от 1 до 10:" << endl;

    for (int i = 1; i <= 10; ++i) {
        cout << i << " ";
    }
    cout << "\n\nФакториал числа 5: " << factorial(5) << endl;

    return 0;
}
```

«Отладка в Visual Studio» (Рисунок 29).

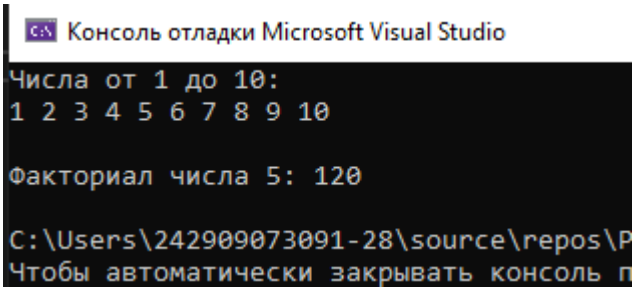


Рисунок 29 – откладка

### 2.2. Управление исключениями

Добавьте код, генерирующий исключение (например, деление на ноль (System.DivideByZeroException). Настройте отладчик для прерывания при определённых исключениях: Отладка → Окна → Параметры исключения (только когда код запущен). (Рисунок 30).

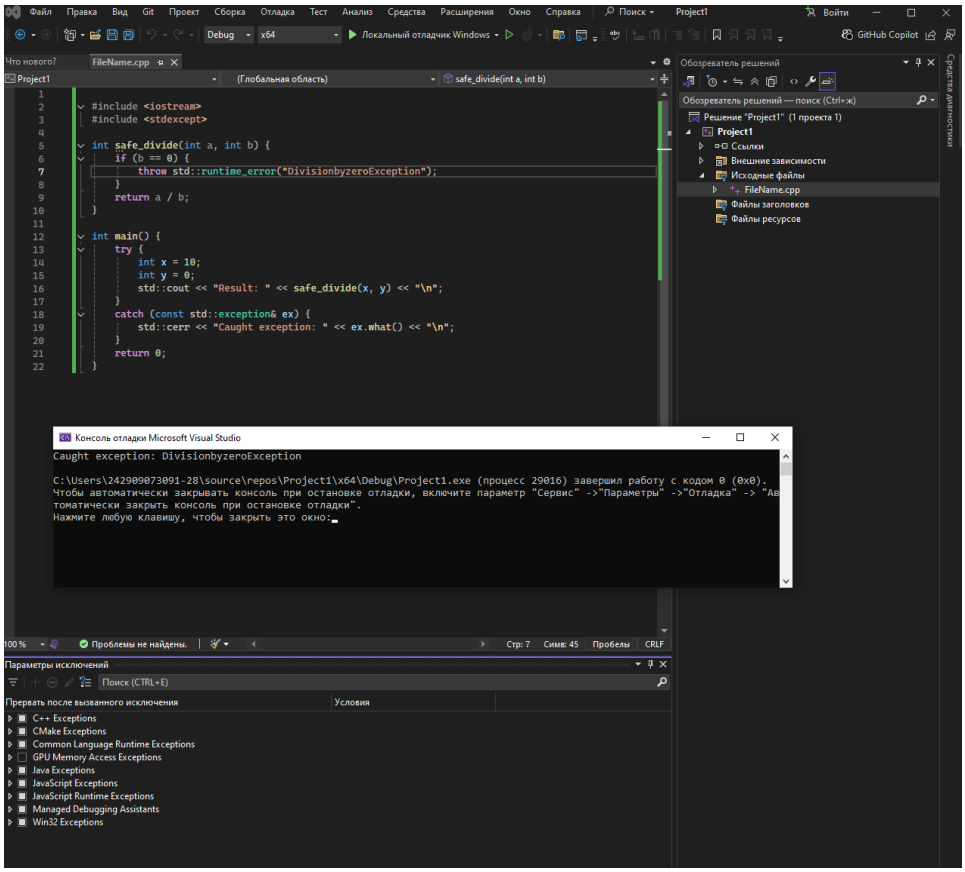


Рисунок 30- Параметры исключения

## 2.3. Профилирование

Найдите ветку «C++ Exceptions». Убедитесь, что галочки для всех исключений установлены (отладчик будет прерываться в момент генерации исключения). Попробуйте снять галочку для std::out_of_range и посмотрите, как изменится поведение. (Рисунок 31).

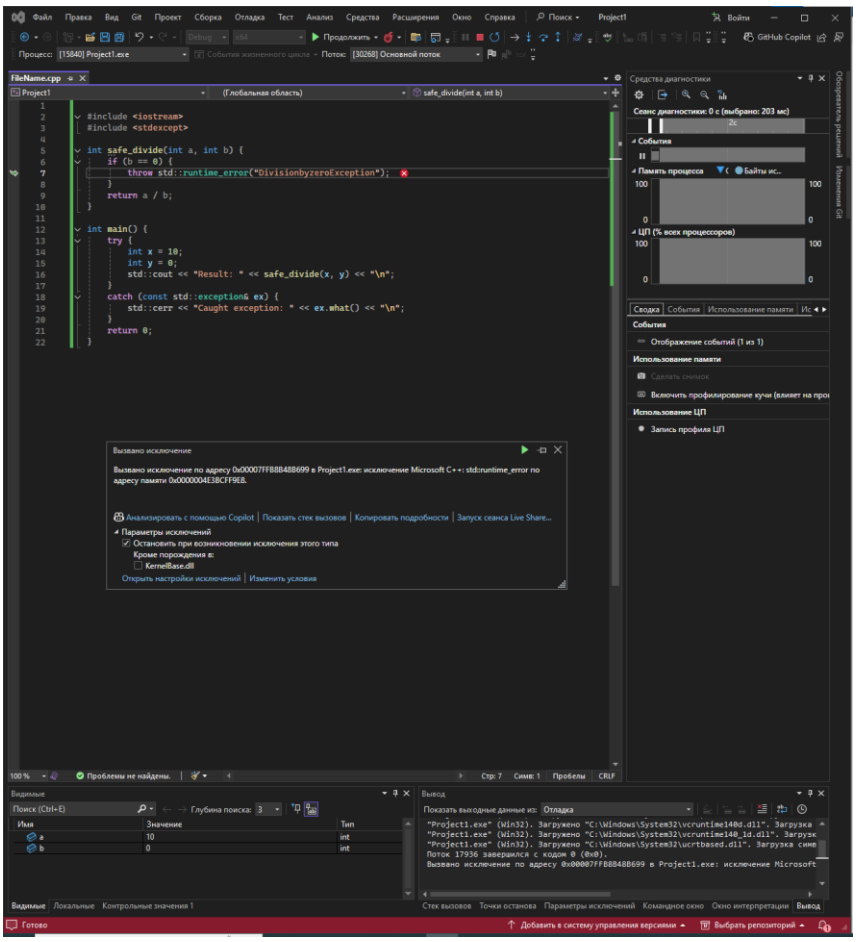


Рисунок 31 – исключение

```cpp
#include <iostream>
#include <stdexcept>

int safe_divide(int a, int b) {
    if (b == 0) {
        throw std::runtime_error("DivisionbyzeroException");
    }
    return a / b;
}

int main() {
    try {
        int x = 10;
        int y = 0;
        std::cout << "Result: " << safe_divide(x, y) << "\n";
    }
    catch (const std::exception& ex) {
        std::cerr << "Caught exception: " << ex.what() << "\n";
    }
    return 0;
}
```

«Отладка в Visual Studio» (Рисунок 32).

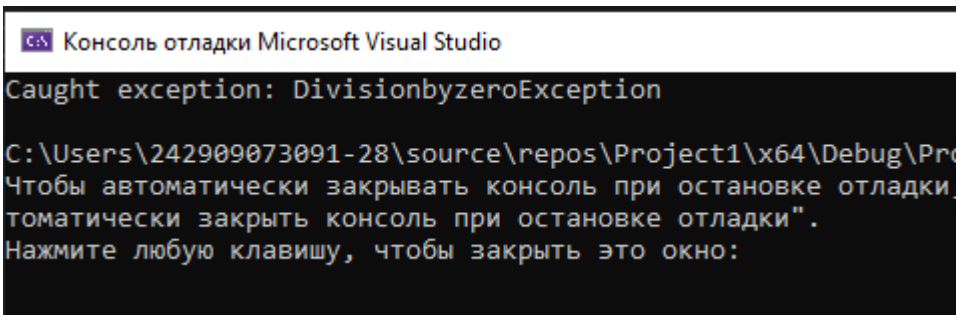


Рисунок 32 - откладка

Добавьте код с интенсивными вычислениями (например, цикл с большим количеством итераций или работа с коллекциями). Используйте Отладка → Профилировщик производительности: (Рисунок 33-35).

- Запустите анализ **Использование ЦП** для определения нагрузочных функций.
- Запустите анализ **Использование памяти** для отслеживания выделения памяти и утечек.
- Сохраните отчёт профилирования.

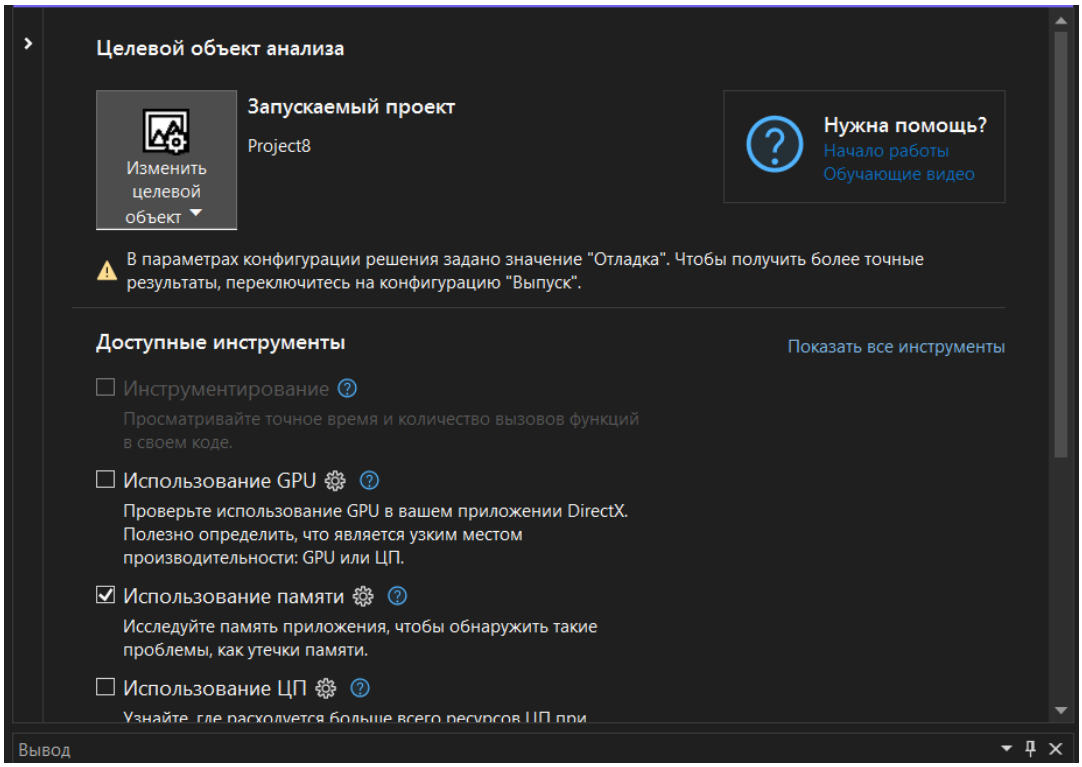


Рисунок 33- целевой объект анализа

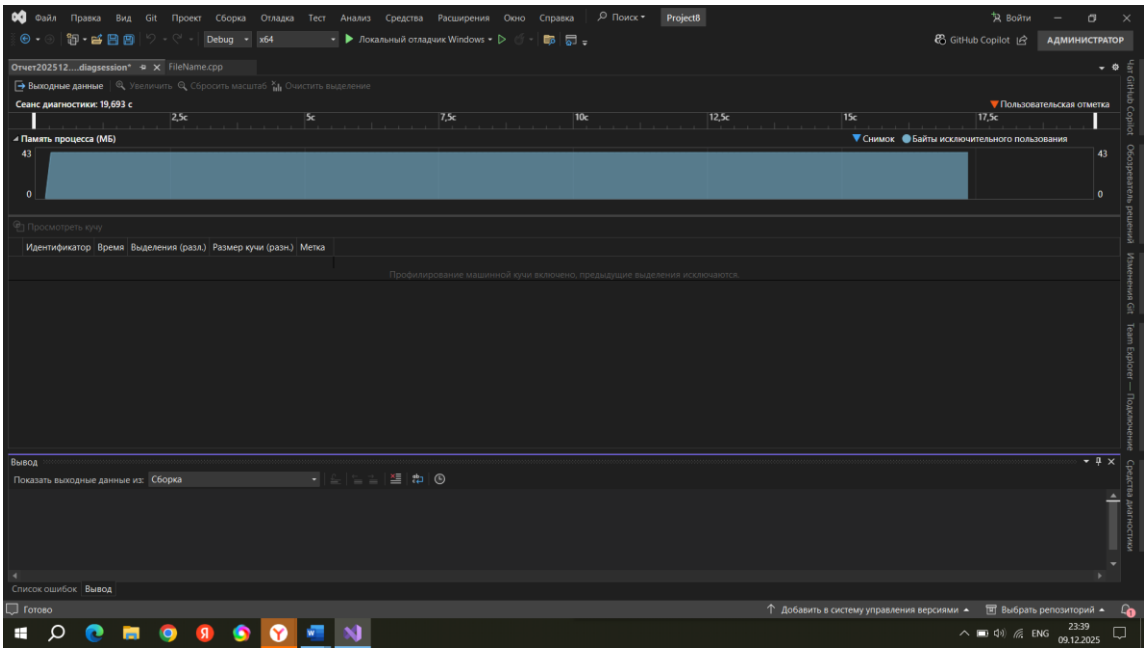


Рисунок 34 - использование памяти

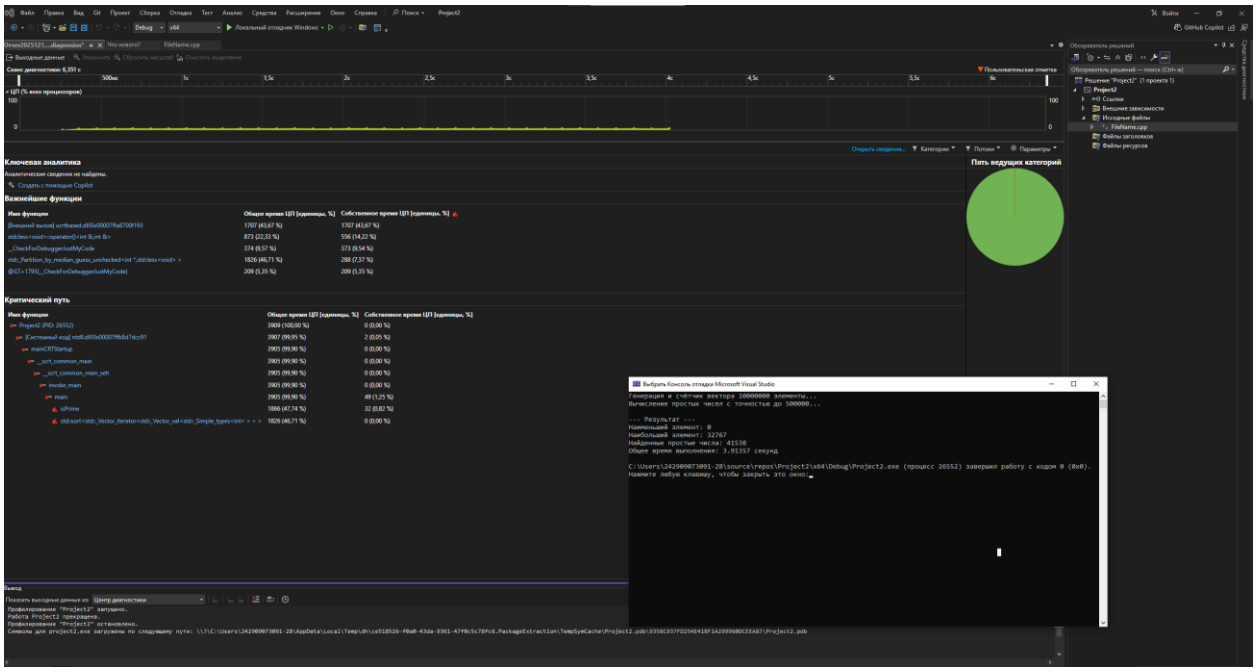

Рисунок 35- использование ЦП

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <chrono>
bool isPrime(int n) {
    setlocale(0, "");
    if (n <= 1) return false;
    for (int i = 2; i <= std::sqrt(n); ++i) {
        if (n % i == 0) return false;
    }
    return true;
}

int main() {
    setlocale(0, "");
    const int VECTOR_SIZE = 10'000'000;
    const int PRIME_LIMIT = 500'000;

    auto start = std::chrono::high_resolution_clock::now();

    std::cout << "Генерация и счётчик вектора " << VECTOR_SIZE << " элементы..."
<< std::endl;
    std::vector<int> data(VECTOR_SIZE);

    for (int i = 0; i < VECTOR_SIZE; ++i) {
        data[i] = rand() % 1000000;
    }

    std::sort(data.begin(), data.end());

    std::cout << "Вычисление простых чисел с точностью до " << PRIME_LIMIT <<
"..." << std::endl;
    int primeCount = 0;
```

```
            for (int i = 0; i < PRIME_LIMIT; ++i) {
                if (isPrime(i)) {
                    primeCount++;
                }
            }

            auto end = std::chrono::high_resolution_clock::now();
            std::chrono::duration<double> elapsed = end - start;

            std::cout << "\n--- Результат ---" << std::endl;
            std::cout << "Наименьший элемент: " << data[0] << std::endl;
            std::cout << "Наибольший элемент: " << data[VECTOR_SIZE - 1] << std::endl;
            std::cout << "Найденные простые числа: " << primeCount << std::endl;
            std::cout << "Общее время выполнения: " << elapsed.count() << " секунд" <<
std::endl;

            return 0;
```

«Отладка в Visual Studio» (Рисунок 36).

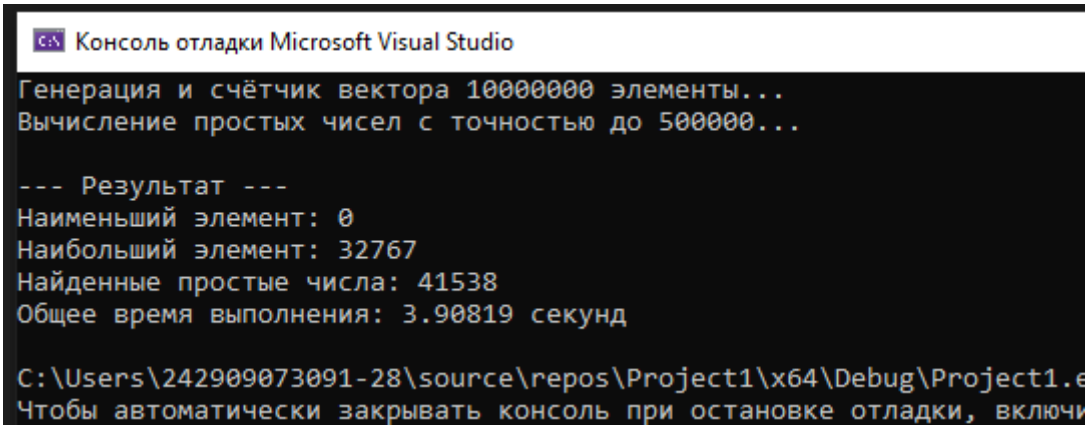


Рисунок 36 - откладка

## Практическая работа №3: «Обеспечение качества кода на C++»

**Цель**: научиться писать чистый, понятный и хорошо документированный код.

Необходимо создать новый консольный проект C++ и написать свою функцию по вариантам. Добавить к функции XML-комментарии.

Вариант 3. Реализуйте функцию вычисления периметра треугольника. Функция принимает три стороны треугольника и возвращает их сумму. Показать пример работы в main.

### 3.1. Самодокументирующийся код

```
#include <iostream>
#include <iomanip>

/// <summary>
/// Вычисляет периметр треугольника по трем его сторонам.
/// </summary>
/// <param name="a">Длина первой стороны.</param>
/// <param name="b">Длина второй стороны.</param>
/// <param name="c">Длина третьей стороны.</param>
/// <returns>
/// Сумма всех сторон (периметр).
/// Возвращает 0, если хотя бы одна сторона отрицательная.
/// </returns>
double calculateTrianglePerimeter(double a, double b, double c) {
    if (a < 0 || b < 0 || c < 0) {
        return 0.0;
    }
    return a + b + c;
}

int main() {
    // Установка локали для корректного отображения кириллицы
    setlocale(LC_ALL, "Russian");

    double side1, side2, side3;

    std::cout << " Расчет периметра треугольника " << std::endl;
    std::cout << "Введите длины трех сторон: ";
```

```
    if (std::cin >> side1 >> side2 >> side3) {
        double perimeter = calculateTrianglePerimeter(side1, side2, side3);

        if (perimeter > 0) {
            std::cout << std::fixed << std::setprecision(2);
            std::cout << "Периметр треугольника равен: " << perimeter << std::endl;
        }
        else {
            std::cout << "Ошибка: длины сторон не могут быть отрицательными." <<
std::endl;
        }
    }
    else {
        std::cout << "Ошибка: введено некорректное значение." << std::endl;
    }

    return 0;
}
```

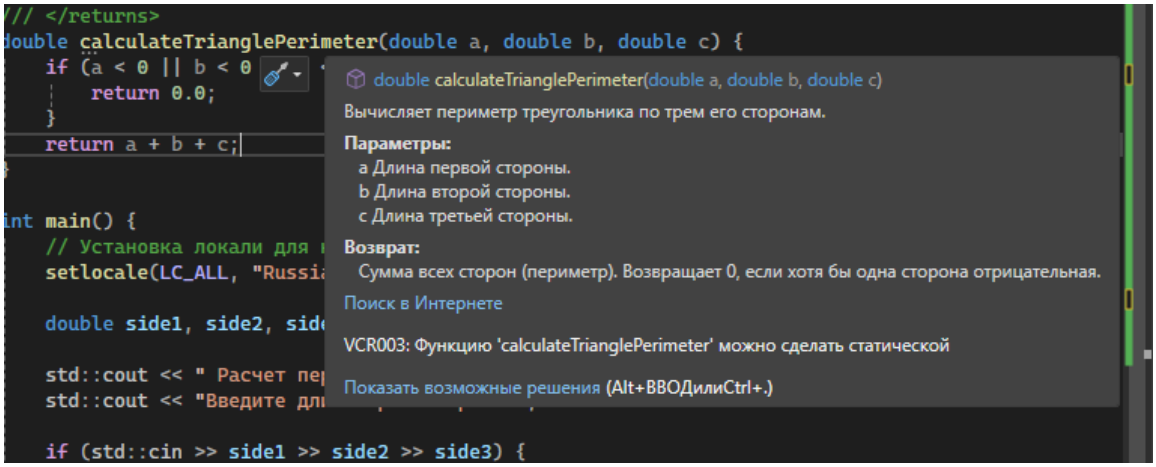


Рисунок 37- подсказка

### 3.2. Соглашение о кодировании

**Цель**: Оформить свой код согласно правилам C++

Исправленный код должен:

Иметь правильные отступы и пробелы

Использовать правильные имена переменных и функций

Иметь пробелы вокруг операторов

Правильно расставленные фигурные скобки

Убрать `using namespace std;` (опционально, но рекомендуется).

```
#include <iostream>
#include <iomanip>

/// <summary>
/// Вычисляет периметр треугольника по трем его сторонам.
/// </summary>
/// <param name="a">Длина первой стороны.</param>
/// <param name="b">Длина второй стороны.</param>
/// <param name="c">Длина третьей стороны.</param>
/// <returns>
/// Сумма всех сторон (периметр).
/// Возвращает 0, если хотя бы одна сторона отрицательная.
/// </returns>
double calculateTrianglePerimeter(double a, double b, double c)
{
    if (a < 0 || b < 0 || c < 0)
    {
        return 0.0;
    }
    return a + b + c;
}

int main()
{
    // Установка локали для корректного отображения кириллицы
    setlocale(LC_ALL, "Russian");

    double side1, side2, side3;

    std::cout << " Расчет периметра треугольника " << std::endl;
    std::cout << "Введите длины трех сторон: ";

    if (std::cin >> side1 >> side2 >> side3)
    {
        double perimeter = calculateTrianglePerimeter(side1, side2, side3);

        if (perimeter > 0)
        {
            std::cout << std::fixed << std::setprecision(2);
            std::cout << "Периметр треугольника равен: " << perimeter <<
std::endl;
        }
        else
        {
            std::cout << "Ошибка: длины сторон не могут быть отрицательными."
<< std::endl;
        }
    }
    else
    {
        std::cout << "Ошибка: введено некорректное значение." << std::endl;
    }

    return 0;

}
```

### 3.3. Метрика Джилба (упрощенный вариант)

**Цель**: Для вашей функции из Задания 1, рассчитать сложность по метрике Джилба:

Посчитайте η1 - количество всех исполняемых строк кода (исключая объявления и комментарии)

Посчитайте η2 - количество логических переходов (if, условия в циклах)

Вычислите Gillb = η1 + η2

```
#include <iostream>
#include <iomanip>


/// <summary>
/// Вычисляет периметр треугольника по трем его сторонам.
/// </summary>
/// <param name="a">Длина первой стороны.</param>
/// <param name="b">Длина второй стороны.</param>
/// <param name="c">Длина третьей стороны.</param>
/// <returns>
/// Сумма всех сторон (периметр).
/// Возвращает 0, если хотя бы одна сторона отрицательная.
/// </returns>
double calculateTrianglePerimeter(double a, double b, double c)
{
    if (a < 0 || b < 0 || c < 0) //+1 к n2
    {
        return 0.0;//+1 к n1
    }
    return a + b + c; //+1 к n1

    int main(); {
    // Установка локали для корректного отображения кириллицы
    setlocale(LC_ALL, "Russian");

    double side1, side2, side3;

    std::cout << " Расчет периметра треугольника " << std::endl;
    std::cout << "Введите длины трех сторон: ";

    if (std::cin >> side1 >> side2 >> side3)//+1 к n2
    {
        double perimeter = calculateTrianglePerimeter(side1, side2, side3);

        if (perimeter > 0) //+1 к n2
        {
            std::cout << std::fixed << std::setprecision(2);
            std::cout << "Периметр треугольника равен: " << perimeter <<
std::endl;
        }
        else//+1 к n2
```

```
                {
                    std::cout << "Ошибка: длины сторон не могут быть отрицательными."
<< std::endl;
                }
            }
            else//+1 к n2
            {
                std::cout << "Ошибка: введено некорректное значение." << std::endl;
            }

            return 0;//+1 к n1
        }
```

n1=3

n2= 5

Gillb = 3 + 5 = 8

**Вывод**: Функция имеет низкую сложность (Gillb=8), код простой.

«Отладка в Visual Studio» (Рисунок 38).

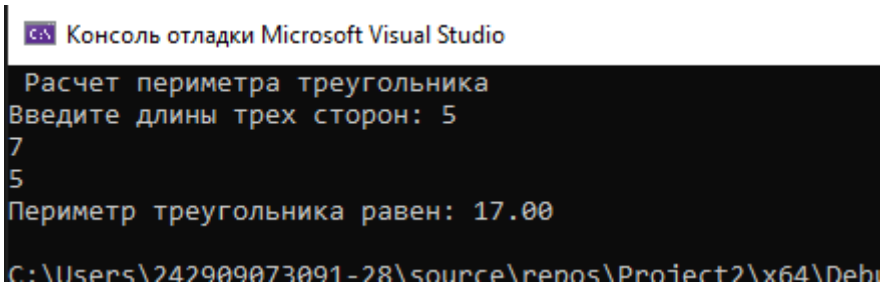


Рисунок 38 - откладка

## Практическая работа №4 «Работа с реестром ОС Windows»

**Цель**: Понять что такое реестр, для чего нужен. Научиться работать с файлами.

Необходимо создать консольное приложение и реализовать функционал для сохранения и восстановления из файла.

**Вариант 3**

1. Сохранить и вывести любимую еду.

2. Сохранить цвет фона консоли и восстанавливать его при запуске.

```
#include <iostream>
#include <string>
#include <windows.h>
#include <vector>

const char* REG_SUBKEY = "Software\\MyFavoriteApp";


void SaveSettingsToRegistry(const std::string& food, DWORD colorCode) {
    HKEY hKey;

    if (RegCreateKeyExA(HKEY_CURRENT_USER, REG_SUBKEY, 0, NULL,
        REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hKey, NULL) ==
ERROR_SUCCESS) {

        RegSetValueExA(hKey, "FavoriteFood", 0, REG_SZ,
            (const BYTE*)food.c_str(), (DWORD)(food.length() + 1));

        RegSetValueExA(hKey, "BackgroundColor", 0, REG_DWORD,
            (const BYTE*)&colorCode, sizeof(DWORD));

        RegCloseKey(hKey);
    }
}


bool LoadSettingsFromRegistry(std::string& food, DWORD& colorCode) {
    HKEY hKey;
    if (RegOpenKeyExA(HKEY_CURRENT_USER, REG_SUBKEY, 0, KEY_READ, &hKey) ==
ERROR_SUCCESS) {
        char buffer[255];
        DWORD bufferSize = sizeof(buffer);
```

```
            if (RegQueryValueExA(hKey, "FavoriteFood", NULL, NULL, (BYTE*)buffer,
&bufferSize) == ERROR_SUCCESS) {
                food = buffer;
            }

            DWORD color;
            DWORD colorSize = sizeof(DWORD);
            if (RegQueryValueExA(hKey, "BackgroundColor", NULL, NULL,
(BYTE*)&color, &colorSize) == ERROR_SUCCESS) {
                colorCode = color;
            }

            RegCloseKey(hKey);
            return true;
        }
        return false;
    }

    void ApplyConsoleColor(DWORD colorCode) {

        char command[10];
        sprintf_s(command, "color %XF", colorCode);
        system(command);
    }

    int main() {
        setlocale(LC_ALL, "Russian");

        std::string food;
        DWORD colorCode = 0;

        if (LoadSettingsFromRegistry(food, colorCode)) {
            ApplyConsoleColor(colorCode);
            std::cout << "--- Настройки восстановлены из РЕЕСТРА ---" <<
std::endl;
            std::cout << "Ваша сохраненная любимая еда: " << food << std::endl;
        }
        else {
            std::cout << "Настройки в реестре не найдены (первый запуск)." <<
std::endl;
        }
        std::cout << "\nВведите вашу любимую еду: ";
        std::getline(std::cin, food);

        std::cout << "Выберите цвет фона (0-Черный, 1-Синий, 2-Зеленый, 4-
Красный, 5-Фиолетовый): ";
        std::cin >> colorCode;

        ApplyConsoleColor(colorCode);
        SaveSettingsToRegistry(food, colorCode);

        std::cout << "\nДанные сохранены в реестр! Перезапустите программу для
проверки." << std::endl;

        system("pause");
        return 0;
    }
```

«Отладка в Visual Studio» (Рисунок 38).

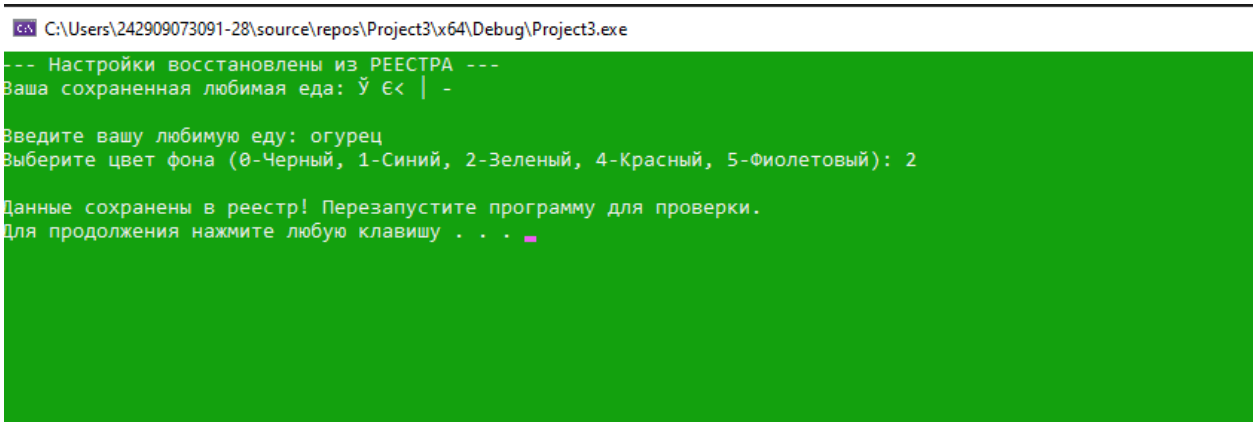


Рисунок 38 - откладка

## Практическая работа №5 «Задание для тестировщика»

**Цель**: Провести комплексное тестирование сайта (сайты по вариантам ниже) с целью оценки его качества, удобства использования и выявления потенциальных дефектов.

Необходимо рассмотреть сайт https://p76om.ru/.

### 1. Функциональное тестирование (Functional Testing)

Проверяем, все ли функции работают так, как задумано.

- **Навигация:**

Все кнопки, ссылки в меню и конки кликабельны. (Рисунок 39).

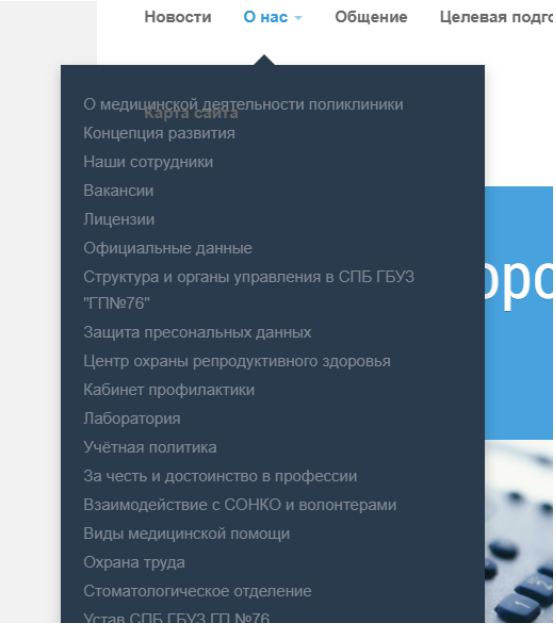


Рисунок 39

Навигационное меню работает на телефоне и на десктопе. В нижней части сайта ссылки кликабельны. Поиск по сайту работает. (Рисунок 40).

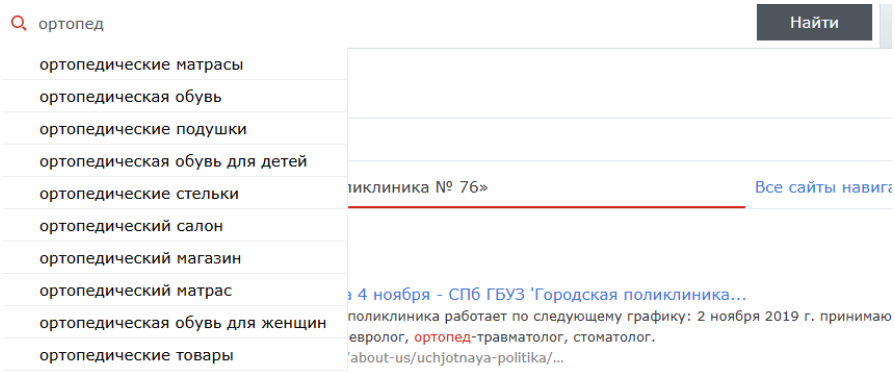


Рисунок 40

- **Формы:**

Форма проверяет введённые данные и грамотно сообщает об ошибке. (Рисунок 41).

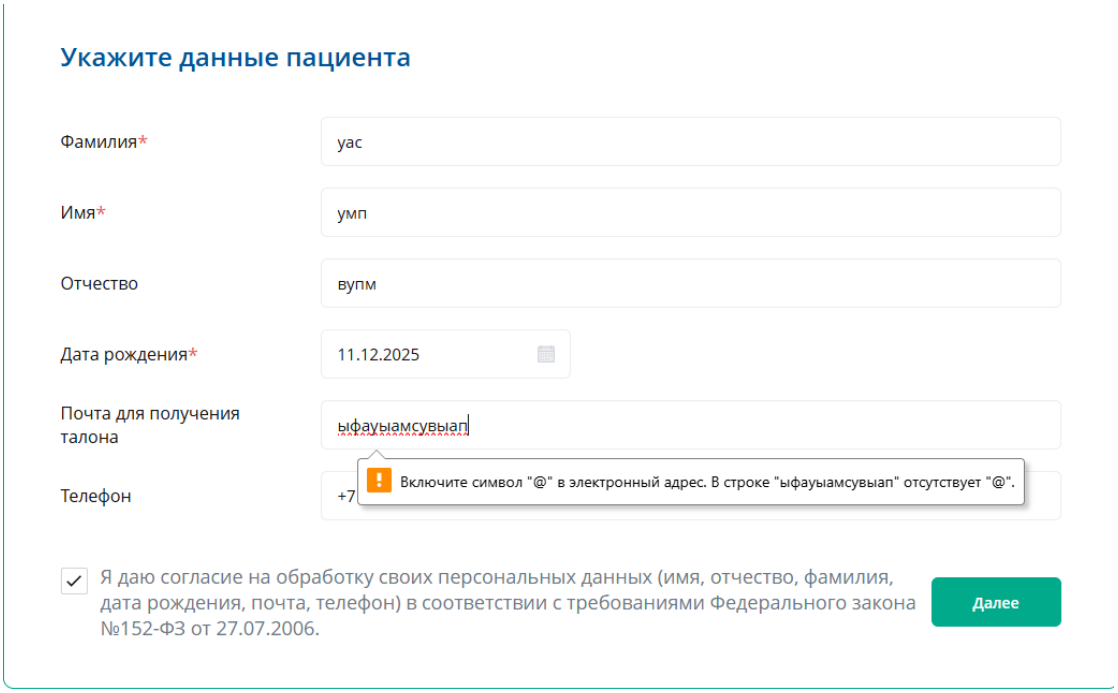


Рисунок 41

Форму нельзя отправить если поля не заполнены. (Рисунок 42).

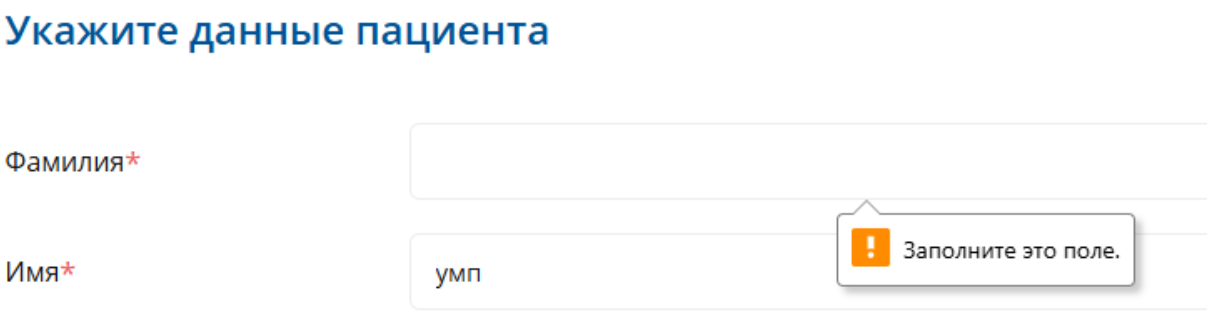


Рисунок 42

После успешной отправки формы на почту приходит талон и форма отчищается.

- **Интерактивные элементы:**

Карусель работают корректно. (Рисунок 43).


Рисунок 43

Аккордеоны работают исправно. (Рисунок 44).

| | |
|---|---|
| Адмиралтейский | Курортный |
| Василеостровский | Московский |
| Выборгский | Невский |
| Калининский | Петроградский |
| Кировский | Петродворцовый |
| Колпинский | Приморский |
| Красногвардейский | Пушкинский |
| Красносельский | Фрунзенский |
| Кронштадтский | Центральный |

Рисунок 44

## 2. Юзабилити (Usability) и Пользовательский опыт (UX)

Проверяем, насколько сайт удобен и понятен для пользователя. (Рисунок 45).

- **Первое впечатление:** Понятна ли цель сайта с первых секунд? Легко ли найти нужную информацию?
- **Контент:**

Текст читабелен.

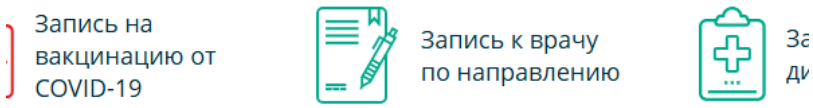


Рисунок 45

Грамматических и орфографических ошибок нет. Все изображения не искажены.

- **Навигация и структура:**

На сайте легко понять, где ты находишься, до самого необходимого можно добраться за 2-3 клика.

- **Обратная связь:**

Сайт даёт обратную связь (при наведении на кнопку она меняет цвет).

### 3. Тестирование совместимости (Compatibility Testing)

- **Кросс-браузерность:** Сайт работает в разных браузерах, он работает и выглядит одинаково.

- **Кросс-платформенность:**

Сайт хорошо работает на IOS и планшете. Сайт адаптирован под мобильное устройство.

### 4. Производительность (Performance)

Проверяем скорость работы сайта. (Рисунок 46).

**Скорость загрузки:** Сайт достаточно быстро загружается.

**Скорость реакции:** Сайт достаточно быстро реагирует на действия.

**Загрузка контента:** Сайт не виснет при загрузке тяжёлых изображений.


Рисунок 46

## 5. Тестирование безопасности (Security Testing) - (Базовая проверка)

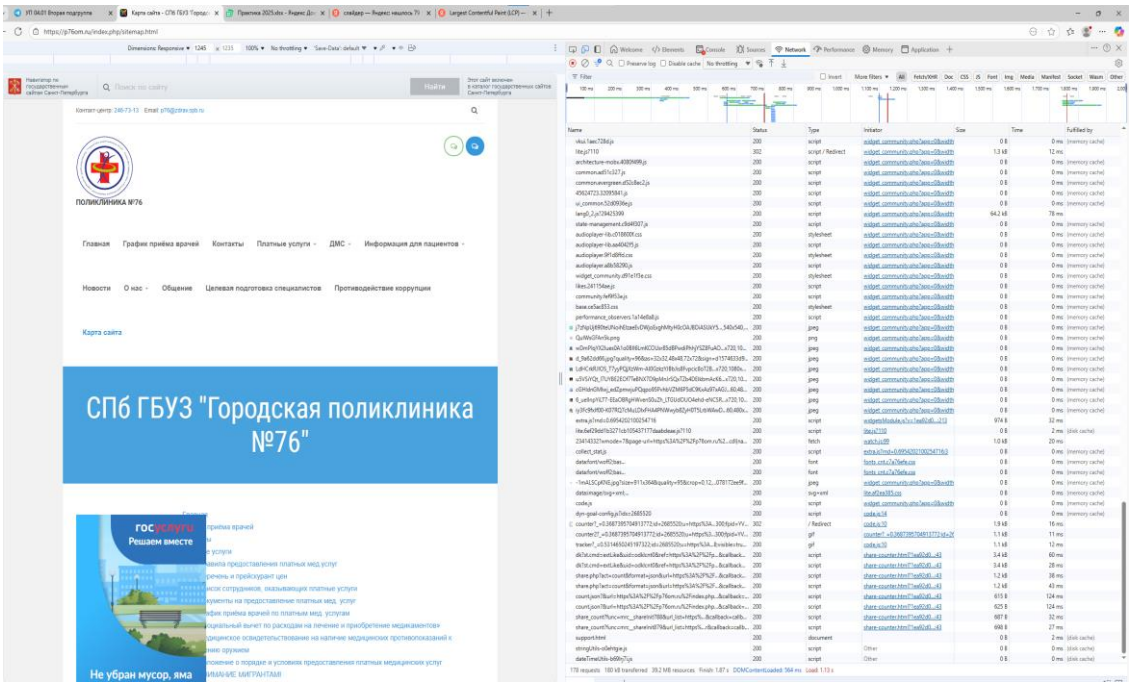


Рисунок 47

**Формы:** Данные не передаются. (Рисунок 48).


Рисунок 48

Ошибки консоли:

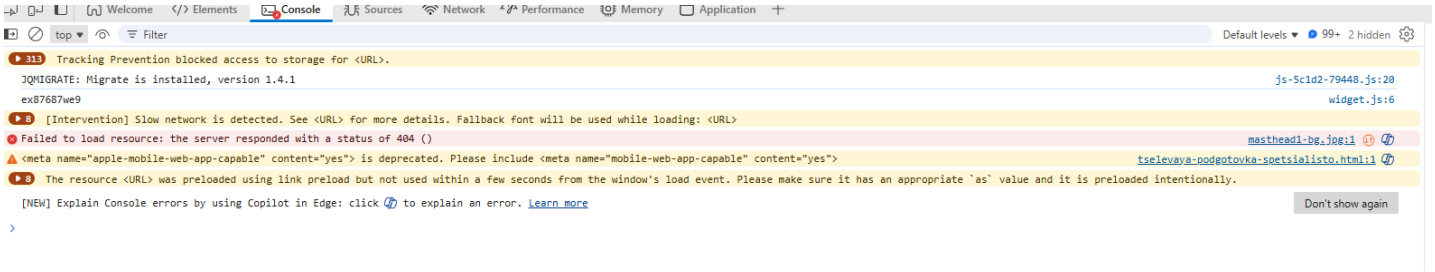


Рисунок 49

## Практическая работа №6 «Создание инсталляторов»

**Цель работы**: создать консольное приложение с классом и сформировать итоговый установочный пакет.

### Задание 1: Создание консольного приложения (ConsoleApp) с классом.

```
#include <iostream>
#include <vector>

class ElementFinder {
private:
    std::vector<int> array;
    int size;

public:
    // Конструктор по умолчанию
    ElementFinder() : size(0) {}

    // Конструктор с параметрами
    ElementFinder(int* arr, int sz) {
        setArray(arr, sz);
    }

    // Установить массив
    void setArray(int* arr, int sz) {
        size = sz;
        array.clear();
        for (int i = 0; i < sz; i++) {
            array.push_back(arr[i]);
        }
    }

    // Найти элемент (вернуть индекс первого вхождения)
    // Возвращает -1, если элемент не найден
    int findElement(int target) const {
        for (int i = 0; i < size; i++) {
            if (array[i] == target) {
                return i;
            }
        }
        return -1; // Элемент не найден
    }

    // Посчитать количество вхождений элемента
    int countOccurrences(int target) const {
        int count = 0;
        for (int i = 0; i < size; i++) {
            if (array[i] == target) {
```

```
            count++;
        }
    }
    return count;
}

// Проверить наличие элемента
bool contains(int target) const {
    return findElement(target) != -1;
}

// Дополнительные методы для удобства

// Найти все индексы вхождений элемента
std::vector<int> findAllIndices(int target) const {
    std::vector<int> indices;
    for (int i = 0; i < size; i++) {
        if (array[i] == target) {
            indices.push_back(i);
        }
    }
    return indices;
}

// Найти индекс последнего вхождения элемента
int findLastElement(int target) const {
    for (int i = size - 1; i >= 0; i--) {
        if (array[i] == target) {
            return i;
        }
    }
    return -1;
}

// Получить размер массива
int getSize() const {
    return size;
}

// Вывести массив
void printArray() const {
    setlocale(0, "");
    std::cout << "Массив: [";
    for (int i = 0; i < size; i++) {
        std::cout << array[i];
        if (i < size - 1) std::cout << ", ";
    }
    std::cout << "]" << std::endl;
}

// Получить элемент по индексу
```

```
    int getElement(int index) const {
        setlocale(0, "");
        if (index >= 0 && index < size) {
            return array[index];
        }
        throw std::out_of_range("Индекс вне диапазона");
    }
};

// Демонстрация работы класса
int main() {
    setlocale(0, "");
    // Создание массива
    int arr[] = { 5, 3, 7, 3, 9, 3, 1, 7, 3 };
    int size = sizeof(arr) / sizeof(arr[0]);

    // Создание объекта класса
    ElementFinder finder;
    finder.setArray(arr, size);

    std::cout << "=== Демонстрация класса ElementFinder ===" <<
std::endl;
    finder.printArray();
    std::cout << std::endl;

    // Поиск элемента
    int target = 7;
    int index = finder.findElement(target);
    std::cout << "findElement(" << target << "): ";
    if (index != -1) {
        std::cout << "найден на индексе " << index << std::endl;
    }
    else {
        std::cout << "не найден" << std::endl;
    }

    // Подсчет вхождений
    target = 3;
    int count = finder.countOccurrences(target);
    std::cout << "countOccurrences(" << target << "): " << count << "
раз(а)" << std::endl;

    // Проверка наличия элемента
    target = 9;
    bool exists = finder.contains(target);
    std::cout << "contains(" << target << "): " << (exists ? "да" :
"нет") << std::endl;

    target = 15;
    exists = finder.contains(target);
```

```
    std::cout << "contains(" << target << "): " << (exists ? "да" :
"нет") << std::endl;

    std::cout << std::endl;

    // Дополнительные примеры
    std::cout << "=== Дополнительные методы ===" << std::endl;

    // Найти все индексы
    target = 3;
    std::vector<int> indices = finder.findAllIndices(target);
    std::cout << "findAllIndices(" << target << "): [";
    for (size_t i = 0; i < indices.size(); i++) {
        std::cout << indices[i];
        if (i < indices.size() - 1) std::cout << ", ";
    }
    std::cout << "]" <<
        std::endl;

    // Найти последнее вхождение
    target = 7;
    int lastIndex = finder.findLastElement(target);
    std::cout << "findLastElement(" << target << "): " << lastIndex <<
std::endl;

    std::cout << std::endl;

    // Создание нового массива с помощью конструктора
}
```

**Задание 2: Создание инсталлятора**

Для этого задания нем необходимо после написания кода собрать решение в Release. Далее необходимо в папке найти «x64 → Release → Project1.exe», далее необходимо создать папку со своим названием и добавить туда инструкцию README. Выделить эти два файла, нажать правой кнопокй мыши, «7-Zip → Добавить к архиву…». Выбрать нудный форматт архива и нажать «Ок». (Рисунок 50-57).

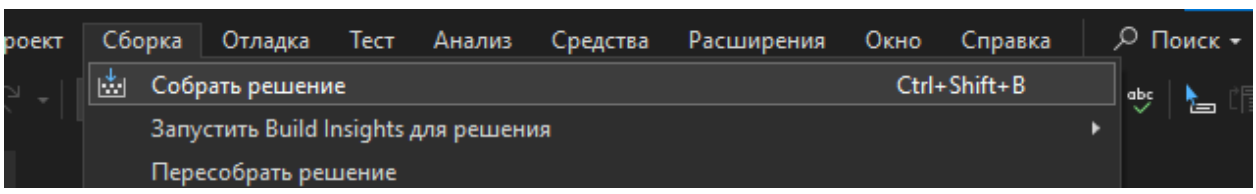


Рисунок 50

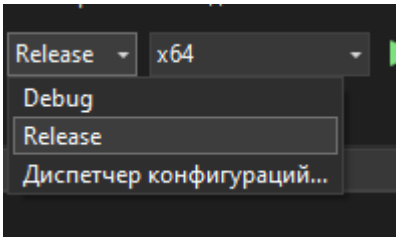


Рисунок 51

| Имя | Дата изменения | Тип | Размер |
|---|---|---|---|
| x64 | 17.12.2025 9:45 | Папка с файлами | |
| FileName.cpp | 17.12.2025 9:45 | Исходный файл ... | 5 КБ |
| Project1.vcxproj | 17.12.2025 9:45 | VC++ Project | 7 КБ |
| Project1.vcxproj.filters | 17.12.2025 9:45 | VC++ Project Filte... | 2 КБ |
| Project1.vcxproj.user | 17.12.2025 9:44 | Per-User Project O... | 1 КБ |

Рисунок 52

| | | |
|---|---|---|
| Debug | 17.12.2025 9:45 | Папка с файлами |
| Release | 17.12.2025 9:45 | Папка с файлами |

Рисунок 53

| | | | |
|---|---|---|---|
| Project1.exe | 17.12.2025 9:45 | Приложение | 18 КБ |
| Project1.pdb | 17.12.2025 9:45 | Program Debug D... | 828 КБ |

Рисунок 54

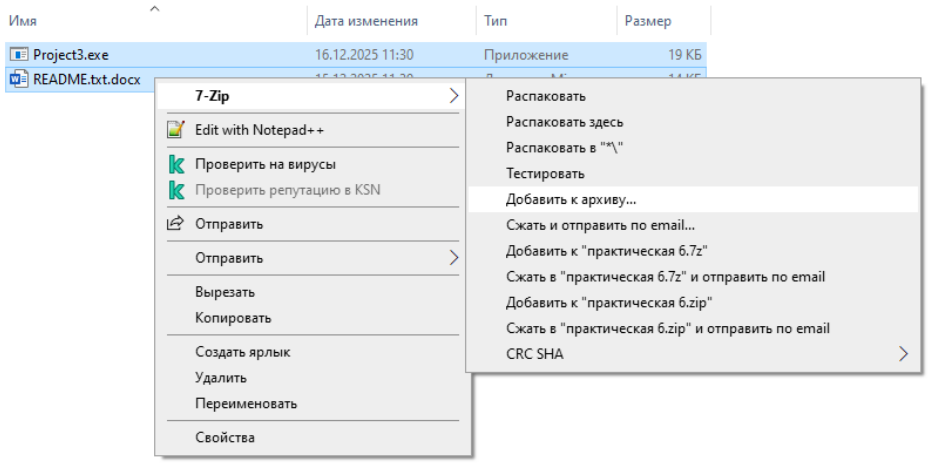


Рисунок 55

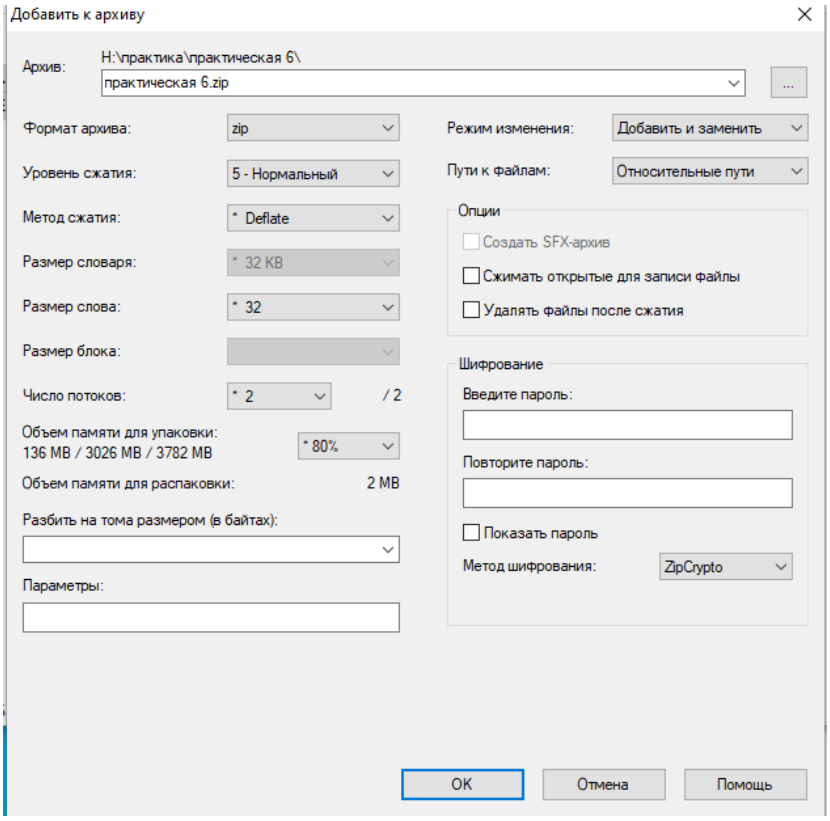


Рисунок 56

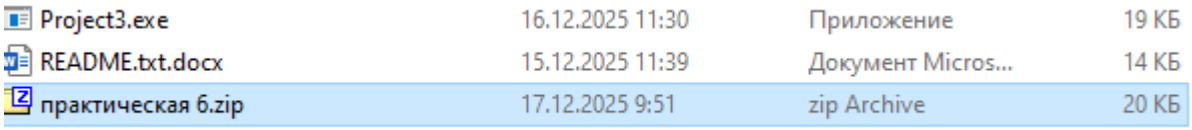


Рисунок 57

```
#include <stdexcept>
#include <iostream>
#include <cmath>


struct PerimeterOperations {

    // Периметр квадрата: P = 4a
    static double squarePerimeter(double a) {
        if (a <= 0) {
            throw std::invalid_argument("Сторона квадрата должна быть
положительной");
        }
        return 4.0 * a;
    }

    // Периметр прямоугольника: P = 2(a + b)
    static double rectanglePerimeter(double a, double b) {
        if (a <= 0 || b <= 0) {
            throw std::invalid_argument("Стороны прямоугольника должны быть
положительными");
        }
        return 2.0 * (a + b);
    }

    // Периметр треугольника: P = a + b + c
    static double trianglePerimeter(double a, double b, double c) {
        if (a <= 0 || b <= 0 || c <= 0) {
            throw std::invalid_argument("Стороны треугольника должны быть
положительными");
        }
        // Неравенство треугольника
        if (a + b <= c || a + c <= b || b + c <= a) {
            throw std::invalid_argument("Треугольник с такими сторонами не
существует");
        }
        return a + b + c;
    }

    // Длина окружности: C = 2πr
    static double circleCircumference(double r) {
        if (r <= 0) {
            throw std::invalid_argument("Радиус должен быть положительным");
        }
        const double PI = 3.14159265358979323846;
        return 2.0 * PI * r;
    }
};
```

```
int main() {
    setlocale(0, "");
    std::cout << "Квадрат (a=2): " << PerimeterOperations::squarePerimeter(2) <<
std::endl;
    std::cout << "Прямоугольник (2,3): " <<
PerimeterOperations::rectanglePerimeter(2, 3) << std::endl;
    std::cout << "Треугольник (3,4,5): " <<
PerimeterOperations::trianglePerimeter(3, 4, 5) << std::endl;
    std::cout << "Окружность (r=1): " << PerimeterOperations::circleCircumference(1)
<< std::endl;
    return 0;
```

## Практическая работа №7 «Тестирование»

**Цель**: научиться работать со структурами, понять разницу между классом и структурой, научиться тестировать программу с помощью unit-тестов.

Структура для вычисления периметров фигур

Написать структуру для вычисления периметров фигур, реализовать методы:

- static double squarePerimeter() - периметр квадрата
- static double rectanglePerimeter() - периметр прямоугольника
- static double trianglePerimeter() - периметр треугольника
- static double circleCircumference() - длина окружности

**Задание 1**

```
#include <iostream>
#include <stdexcept>
#include <limits>
#include <cmath>

struct PerimeterOperations {
    static double squarePerimeter(double a) {
        if (a <= 0) throw std::invalid_argument("Сторона квадрата
должна быть > 0");
        return 4.0 * a;
    }

    static double rectanglePerimeter(double a, double b) {
        if (a <= 0 || b <= 0) throw std::invalid_argument("Стороны
прямоугольника должны быть > 0");
        return 2.0 * (a + b);
    }

    static double trianglePerimeter(double a, double b, double c) {
        if (a <= 0 || c <= 0) throw std::invalid_argument("Стороны
треугольника должны быть > 0");
        if (a + b <= c || b + c <= a)
            throw std::invalid_argument("Треугольник с такими
сторонами не существует");
        return a + b + c;
    }

    static double circleCircumference(double r) {
        if (r <= 0) throw std::invalid_argument("Радиус должен быть >
0");
```

```
        const double PI = 3.14159265358979323846;
        return 2.0 * PI * r;
    }
};

// безопасное чтение числа double с проверкой ввода
double readDouble(const char* prompt) {
    while (true) {
        std::cout << prompt;
        double x;
        if (std::cin >> x) return x;

        std::cout << "Ошибка ввода. Введите число.\n";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
    }
}

// безопасное чтение пункта меню (int)
int readInt(const char* prompt) {
    while (true) {
        std::cout << prompt;
        int x;
        if (std::cin >> x) return x;

        std::cout << "Ошибка ввода. Введите целое число.\n";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
    }
}

int main() {
    setlocale(0, "");

    while (true) {
        std::cout << "\n=== Вычисление периметров ===\n";
        std::cout << "1) Периметр квадрата\n";
        std::cout << "2) Периметр прямоугольника\n";
        std::cout << "3) Периметр треугольника\n";
        std::cout << "4) Длина окружности\n";
        std::cout << "0) Выход\n";

        int choice = readInt("Выберите пункт: ");

        try {
            if (choice == 0) {
                std::cout << "Выход.\n";
                break;
            }
```

```
            else if (choice == 1) {
                double a = readDouble("Введите сторону квадрата a: ");
                double p = PerimeterOperations::squarePerimeter(a);
                std::cout << "Периметр квадрата: " << p << "\n";
            }
            else if (choice == 2) {
                double a = readDouble("Введите сторону прямоугольника
a: ");
                double b = readDouble("Введите сторону прямоугольника
b: ");
                double p = PerimeterOperations::rectanglePerimeter(a,
b);
                std::cout << "Периметр прямоугольника: " << p << "\n";
            }
            else if (choice == 3) {
                double a = readDouble("Введите сторону треугольника a:
");
                double b = readDouble("Введите сторону треугольника b:
");
                double c = readDouble("Введите сторону треугольника c:
");
                double p = PerimeterOperations::trianglePerimeter(a,
b, c);
                std::cout << "Периметр треугольника: " << p << "\n";
            }
            else if (choice == 4) {
                double r = readDouble("Введите радиус окружности r:
");
                double c =
PerimeterOperations::circleCircumference(r);
                std::cout << "Длина окружности: " << c << "\n";
            }
            else {
                std::cout << "Нет такого пункта меню. Повторите
ввод.\n";
            }
        }
        catch (const std::invalid_argument& e) {
            std::cout << "Ошибка: " << e.what() << "\n";
        }
    }

    return 0;
                                    }
```

«Отладка в Visual Studio» (Рисунок 58).

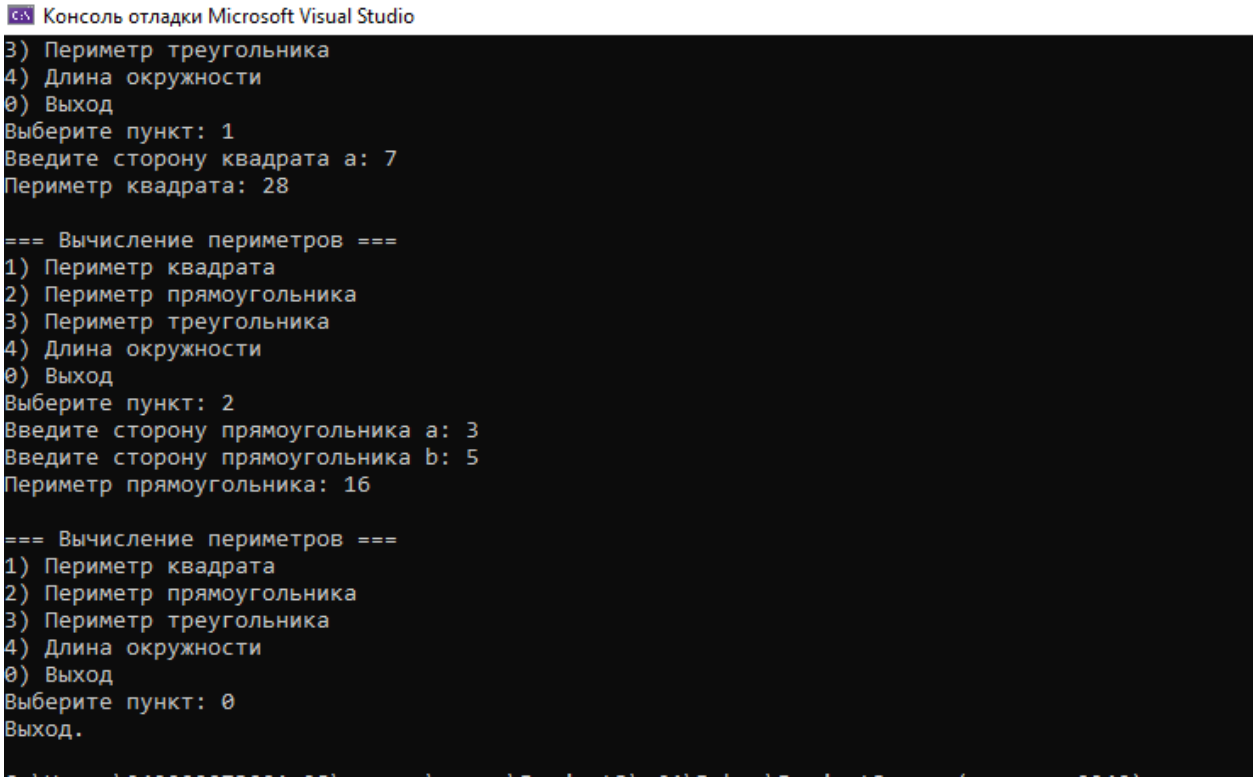


Рисунок 58

**Задание 2**

Написать 2 позитивных и 2 негативных теста.

Для создания проекта модельного тестирование кликаем правой кнопкой мыши по решению и Добавить -> Создать проект -> Проект машинного модульного теста. (Рисунок 59-60).

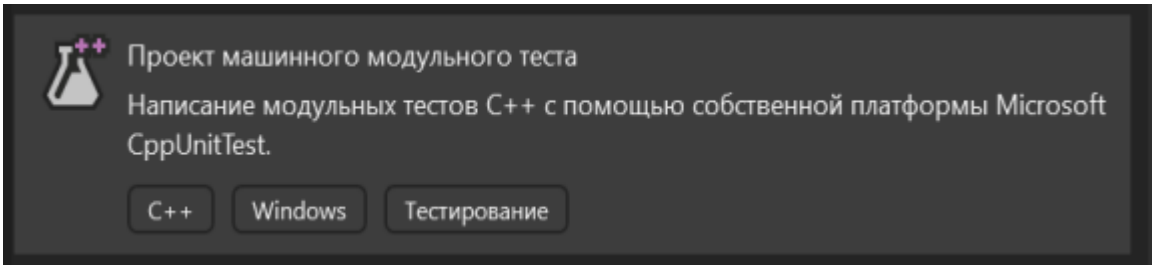


Рисунок 59

Далее пишем 4 теста:

```
#include "pch.h"
#include "CppUnitTest.h"
#include "..\task7\практическая 7.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1
{
    TEST_CLASS(PerimeterOperationsTests)
    {
    public:
        // Позитивный тест 1: квадрат
        TEST_METHOD(TestSquarePerimeter_ValidInput_ReturnsCorrectValue)
        {
            double a = 5.0;
            double result = PerimeterOperations::squarePerimeter(a);
            Assert::AreEqual(20.0, result, 0.0001);
        }

        // Позитивный тест 2: окружность
        TEST_METHOD(TestCircleCircumference_ValidInput_ReturnsCorrectValue)
        {
            double r = 2.0;
            const double PI = 3.14159265358979323846;
            double expected = 2.0 * PI * r;
            double result = PerimeterOperations::circleCircumference(r);
            Assert::AreEqual(expected, result, 0.0001);
        }

        // Негативный тест 1: квадрат с отрицательной стороной
        TEST_METHOD(TestSquarePerimeter_NegativeSide_ThrowsException)
        {
            double a = -1.0;
            Assert::ExpectException<std::invalid_argument>(
                &;
        }
            );
    }

    // Негативный тест 2: несуществующий треугольник
    TEST_METHOD(TestTrianglePerimeter_InvalidSides_ThrowsException)
    {
        double a = 1.0;
        double b = 2.0;
        double c = 3.0;
        Assert::ExpectException<std::invalid_argument>(
            &;
    }
        );
}
    };
}
```

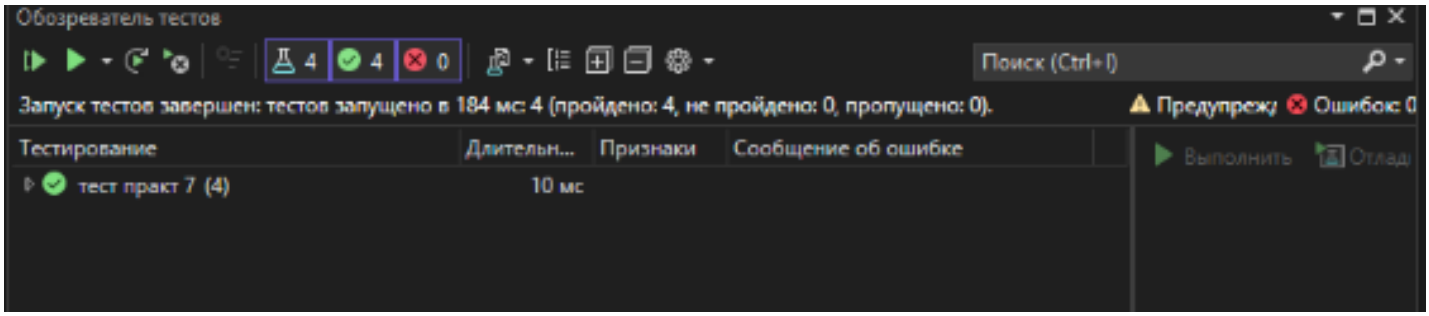


Рисунок 60

**Задание 3**. Написать 1 тест для функции напарника (обменятся проектами, разобраться в коде напарника, протестировать) (Рисунок 61-62).

```
#include <iostream>
#include <cmath>
#include <limits> // Для очистки буфера ввода

struct PowerRootOperations {
    static double square(double a) {
        return a * a;
    }

    static double cube(double a) {
        return a * a * a;
    }

    static double power(double a, double n) {
        return std::pow(a, n);
    }

    static double nthRoot(double a, int n) {
        if (n <= 0) {
            throw std::invalid_argument("Показатель корня должен быть
положительным числом.");
        }

        if (a < 0 && n % 2 == 0) {
            throw std::invalid_argument("Извлечение четного корня из
отрицательного числа невозможно.");
        }
        return std::pow(a, 1.0 / n);
    }
};

void clearInputBuffer() {
    std::cin.clear();
```

```
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
}

int main() {
    setlocale(0, "");

    int choice;
    double number, exponent;
    int rootDegree;

    do {
        std::cout << "Меню операций: " << std::endl;
        std::cout << "1. Возвести число в квадрат" << std::endl;
        std::cout << "2. Возвести число в куб" << std::endl;
        std::cout << "3. Возвести число в произвольную степень" <<
std::endl;
        std::cout << "4. Извлечь корень произвольной степени из числа"
<< std::endl;
        std::cout << "5. Выход" << std::endl;
        std::cout << "Выберите пункт меню: ";

        if (!(std::cin >> choice)) {
            std::cout << "Ошибка ввода! Введите число." << std::endl;
            clearInputBuffer();
            continue;
        }

        switch (choice) {
        case 1: // Квадрат
            std::cout << "Введите число: ";
            if (!(std::cin >> number)) {
                std::cout << "Ошибка ввода числа!" << std::endl;
                clearInputBuffer();
                break;
            }
            std::cout << "Квадрат введенного числа" << number << " = "
<< PowerRootOperations::square(number) << std::endl;
            break;

        case 2: // Куб
            std::cout << "Введите число: ";
            if (!(std::cin >> number)) {
                std::cout << "Ошибка ввода числа!" << std::endl;
                clearInputBuffer();
                break;
            }
            std::cout << "Куб введенного числа" << number << " = " <<
PowerRootOperations::cube(number) << std::endl;
            break;
```

```
        case 3: // Произвольная степень
            std::cout << "Введите число: ";
            if (!(std::cin >> number)) {
                std::cout << "Ошибка ввода числа!" << std::endl;
                clearInputBuffer();
                break;
            }
            std::cout << "Введите степень: ";
            if (!(std::cin >> exponent)) {
                std::cout << "Ошибка ввода степени!" << std::endl;
                clearInputBuffer();
                break;
            }
            std::cout << number << " в степени " << exponent << " = "
<< PowerRootOperations::power(number, exponent) << std::endl;
            break;

        case 4: // Корень произвольной степени
            std::cout << "Введите число: ";
            if (!(std::cin >> number)) {
                std::cout << "Ошибка ввода числа!" << std::endl;
                clearInputBuffer();
                break;
            }
            std::cout << "Введите степень корня (целое положительное
число): ";
            if (!(std::cin >> rootDegree)) {
                std::cout << "Ошибка ввода степени корня!" <<
std::endl;
                clearInputBuffer();
                break;
            }

            try {
                std::cout << "Корень степени " << rootDegree << " из "
<< number << " = " << PowerRootOperations::nthRoot(number, rootDegree)
<< std::endl;
            }
            catch (const std::invalid_argument& e) {
                std::cout << "Ошибка: " << e.what() << std::endl;
            }
            break;

        case 5: // Выход
            std::cout << "Выход из программы." << std::endl;
            break;

        default:
            std::cout << "Неверный пункт меню! Попробуйте снова." <<
std::endl;
            break;
```

```
        }
    } while (choice != 0);
 return 0;
}
```

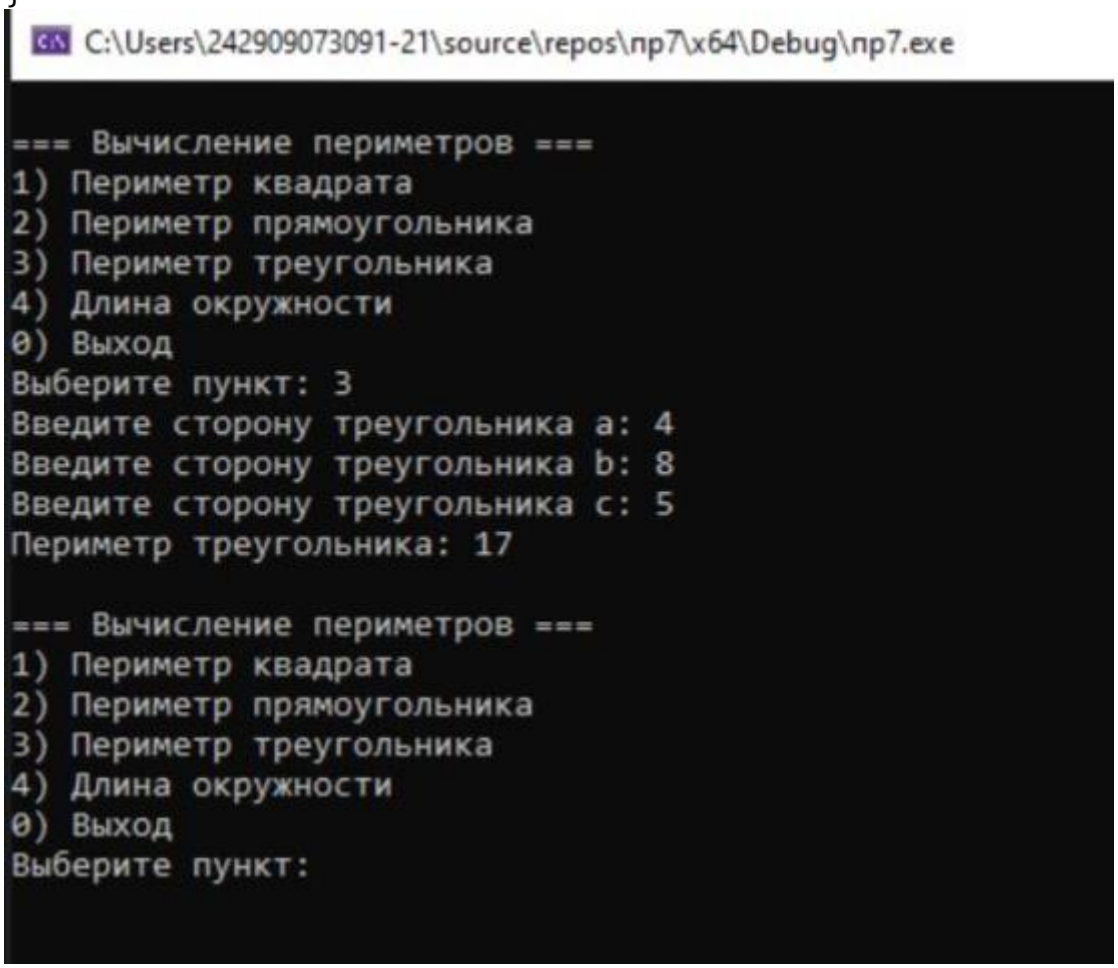


Рисунок 61

```
#include "pch.h"
#include "CppUnitTest.h"
#include "..\task7\task7.cpp"
using namespace Microsoft::VisualStudio::CppUnitTestFramework
namespace UnitTest1
{
    TEST_CLASS(PowerRootOperationsTests)
    {
    public:
        // Позитивный тест: square
        TEST_METHOD(TestSquare_Positive)
        {
            double value = 3.0;
            double expected = 9.0;
            double result = PowerRootOperations::square(value);
            Assert::AreEqual(expected, result, 0.0001);
        }
    }
```

```
};
```

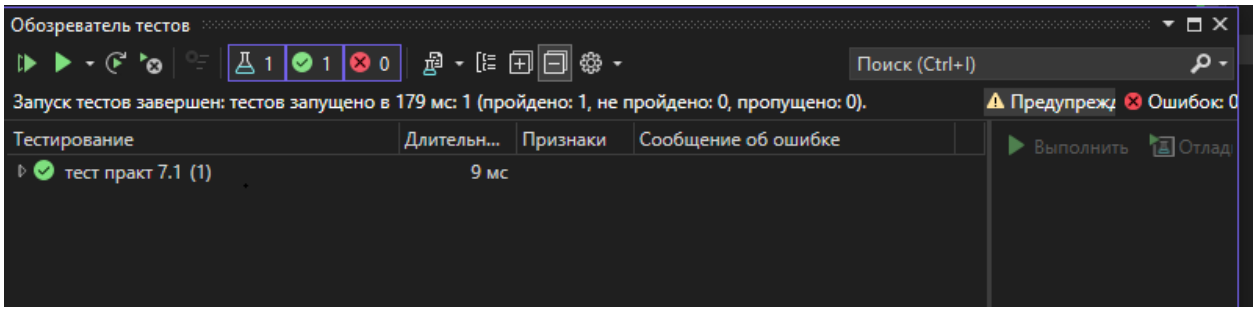


Рисунок 62

**Практическая работа №8: «Разработка ТЗ и макетов для сайта»**

**Цель работы**: научиться создавать четкое техническое задание и разрабатывать к нему визуальные макеты (как черно-белые, так и цветные) для статического сайта.

**Проект: Сайт юридической консультации**

**Часть 1: Разработка Технического Задания (ТЗ)**

**1. Цели и описание проекта**

Ключевая цель: создать профессиональный и надежный онлайн-ресурс для юридической консультации, который будет привлекать новых клиентов, демонстрировать спектр предоставляемых услуг, а также предоставлять удобный инструмент для обратной связи.

Целевая аудитория: физические лица (клиенты, ищущие юридическую помощь) и юридические лица (компании, организации).

Общее описание: проект представляет собой статичный веб-сайт, предназначенный для юридической консультации. Сайт будет содержать информацию о предоставляемых услугах, отзывы клиентов, а также форму для отправки заявок. Цель — создать профессиональный и современный имидж компании, упростить доступ к информации и повысить эффективность коммуникации с клиентами.

## 2. Функциональные требования

Главная страница: статическая страница с приветственным текстом, баннером (слайдером) с тремя ключевыми услугами и кнопкой «Записаться на консультацию».

Страница «Услуги»: страница с описанием всех услуг. Услуги должны быть разделены на две категории: «Для физических лиц» и «Для юридических лиц».

Страница «О нас»: статическая страница с информацией о компании, ее миссии, ценностях и команде. Можно добавить 1-2 личные фотографии (аватар компании/фотографа).

Страница «Отзывы»: страница с отзывами клиентов. Отзывы должны отображаться в виде слайдера (карусель) с возможностью переключения. Количество отзывов — не менее 3-х.

Страница «Контакты»: страница с контактной информацией (адрес, email, телефон) и формой обратной связи. Форма обратной связи должна содержать поля: «Имя» (текстовое), «Email» (с валидацией формата), «Тема» (текстовое), «Сообщение» (многострочное текстовое). Отправка данных из формы должна обрабатываться через внешний сервис без использования базы данных на сайте.

Шапка и подвал сайта: шапка содержит логотип (имя компании/фотографа) и навигационное меню (Главная, Услуги, О нас, Отзывы, Контакты). Подвал содержит копирайт и иконки социальных сетей с ссылками на профили компании.

**3. Техстек и интеграции**

Фронтенд: HTML5, CSS3, чистый JavaScript (можно указать фреймворк, например, Bootstrap, если планируете его использовать).

Хостинг: Статический хостинг (например, GitHub Pages, Netlify).

Интеграции:

Форма обратной связи: Интеграция с внешним сервисом (Formspree/EmailJS) для отправки писем на email фотографа.

Карты: Встроенная карта Google Maps на странице контактов (статический iframe).

Социальные сети: Иконки со ссылками на профили фотографа.

**4. Нефункциональные требования**

a) Производительность:

Время полной загрузки главной страницы не должно превышать 3 секунд при скорости интернета 10 Мбит/с.

Взаимодействия с интерфейсом (прокрутка, переключение слайдеров, открытие модальных окон) должны быть плавными и быстрыми (< 100 мс).

b) Совместимость:

Адаптивность: Сайт должен корректно отображаться на устройствах с шириной экрана от 320px (мобильные) до 1920px (десктоп).

Кроссбраузерность: Сайт должен одинаково работать в последних версиях Chrome, Firefox, Safari и Edge.

c) Удобство использования (Usability):

Стиль: Общее настроение — «Строгий, но современный». Стиль — минимализм, акцент на тексте и качественных фотографиях.

Шрифты: Рубленый шрифт (например, Montserrat) для заголовков, классический sans-serif (например, Lato) для основного текста.

Цветовая палитра: Пастельные, нейтральные цвета (например, серый, белый, синий).

Интерактивные элементы: Все кнопки и ссылки должны иметь визуальный отклик (например, изменение цвета или тени) при наведении мыши.

d) Масштабируемость и сопровождение:

Код должен быть структурирован, с использованием методологии (например, БЭМ) для удобства внесения изменений.

Изображения должны быть подготовлены в нескольких размерах (для ретины и мобильных устройств).

**5. Критерии тестирования**

Проверить корректность отображения и работы всех элементов на разрешениях: 320px, 768px, 1024px, 1920px.

Проверить работу слайдера отзывов (carousel).

Проверить кликабельность всех ссылок (меню, соцсети).

Проверить отображение в разных браузерах (Chrome, Firefox, Safari, Edge).

**6. Сроки поэтапной сдачи**

Этап 1 (Неделя 1): Согласование черно-белых макетов всех страниц.

Этап 2 (Неделя 2): Согласование цветных макетов и финального ТЗ.

Этап 3 (Неделя 3): Верстка главной страницы и других требующихся.

Этап 4 (Неделя 4): Подключение всех интеграций, финальное тестирование и сдача проекта.

# ПРИЛОЖЕНИЕ А


Рисунок А1- Главная страница

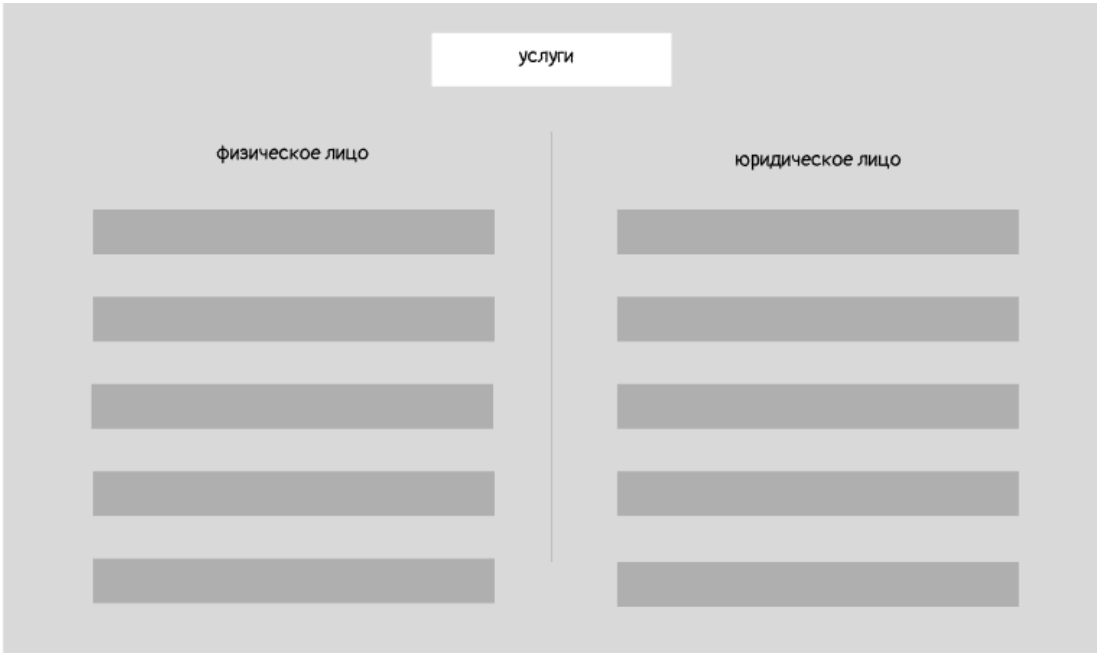


Рисунок А2- “услуги”

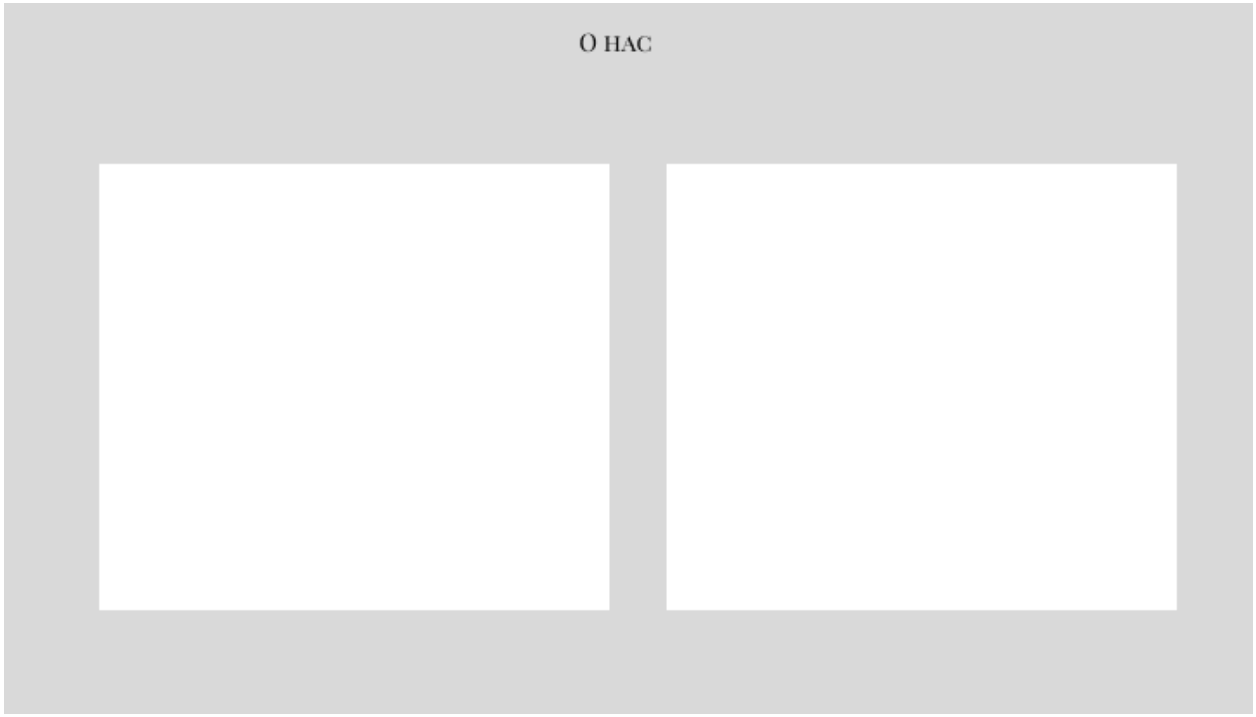


Рисунок А3= “О нас”

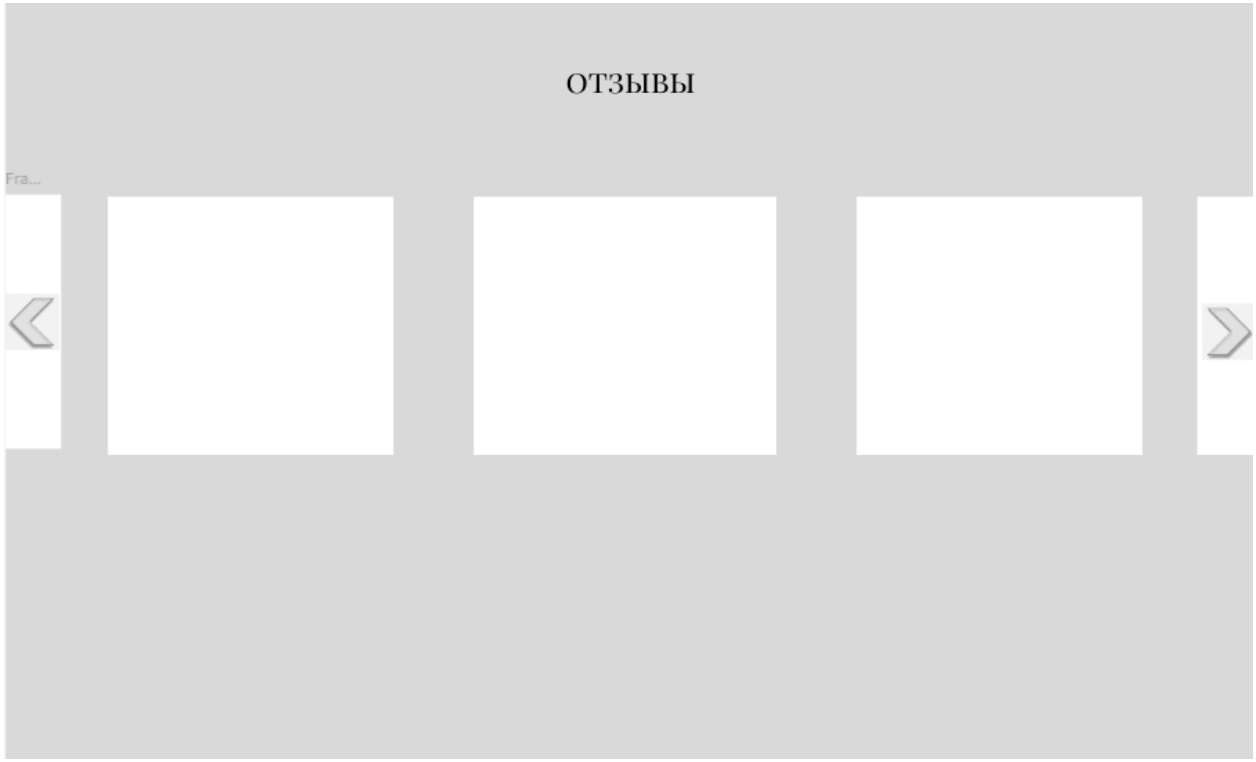


Рисунок А4- отзывы

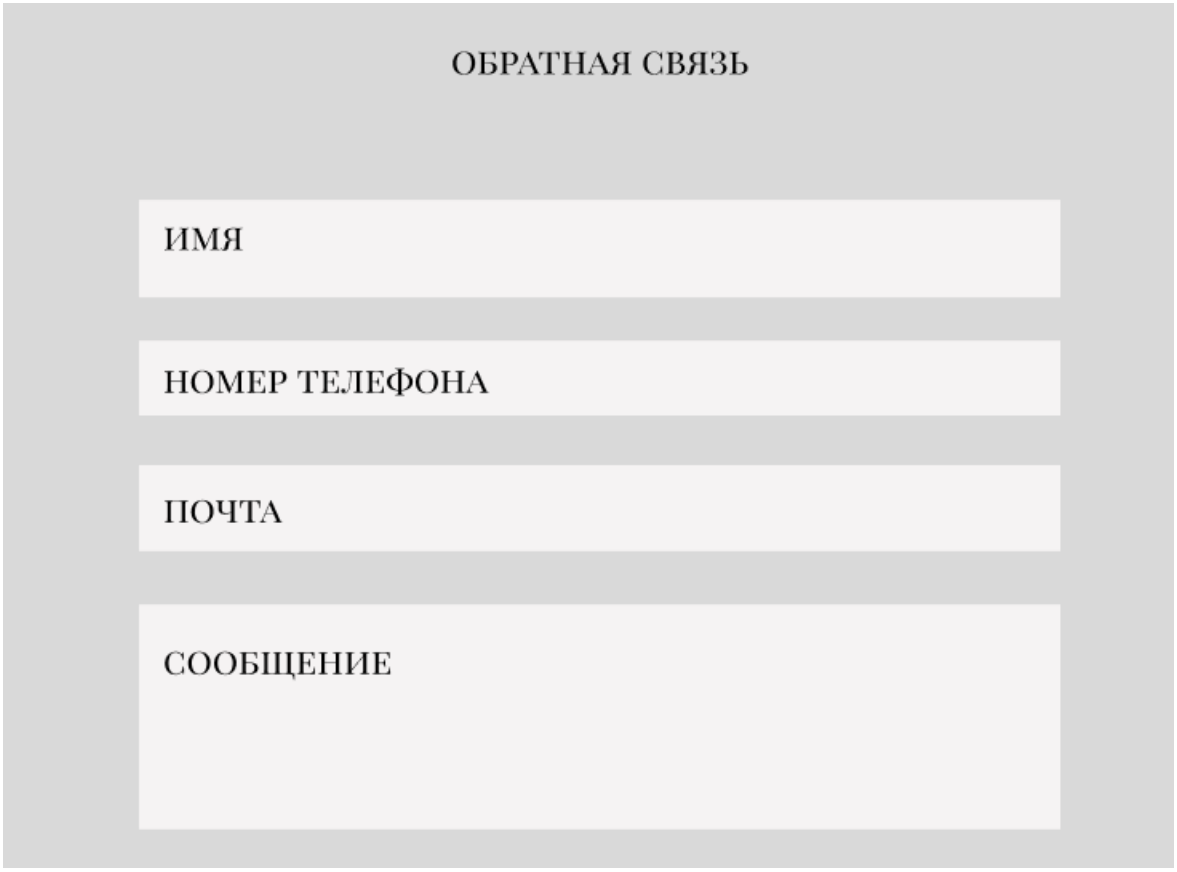


Рисунок А5- обратная связь

# ПРИЛОЖЕНИЕ Б


Рисунок Б1-главная страница


Рисунок Б2-“услуги”

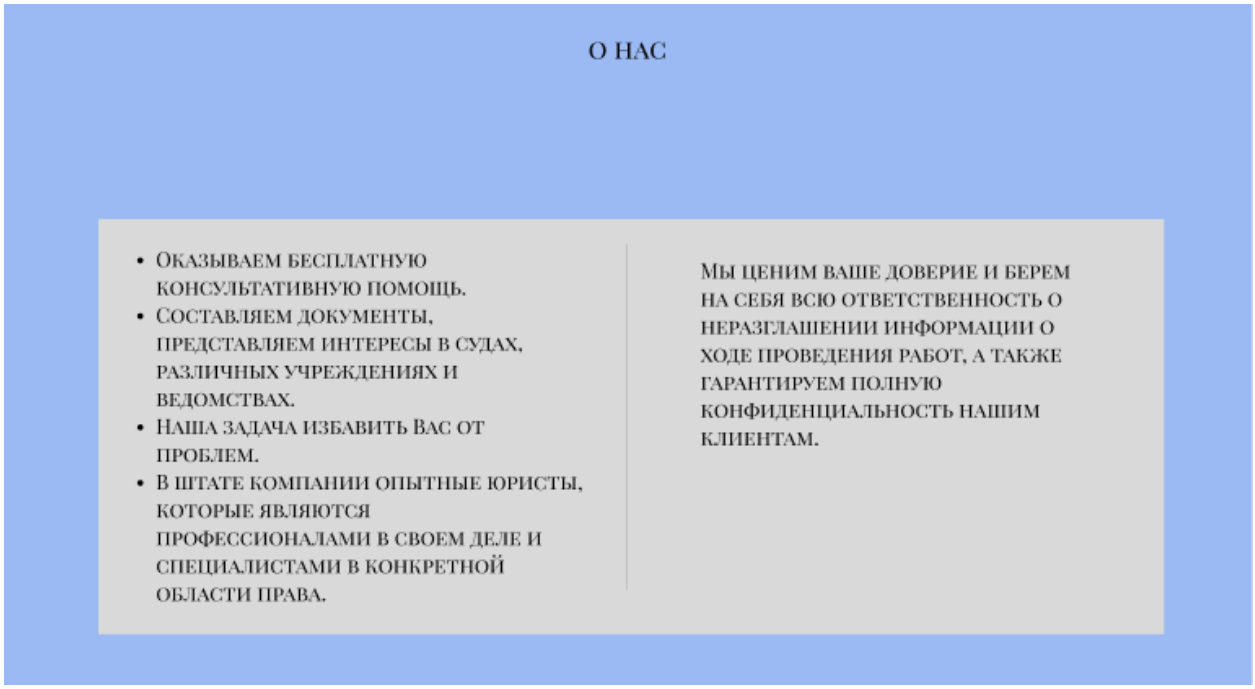


Рисунок Б3- “О нас”

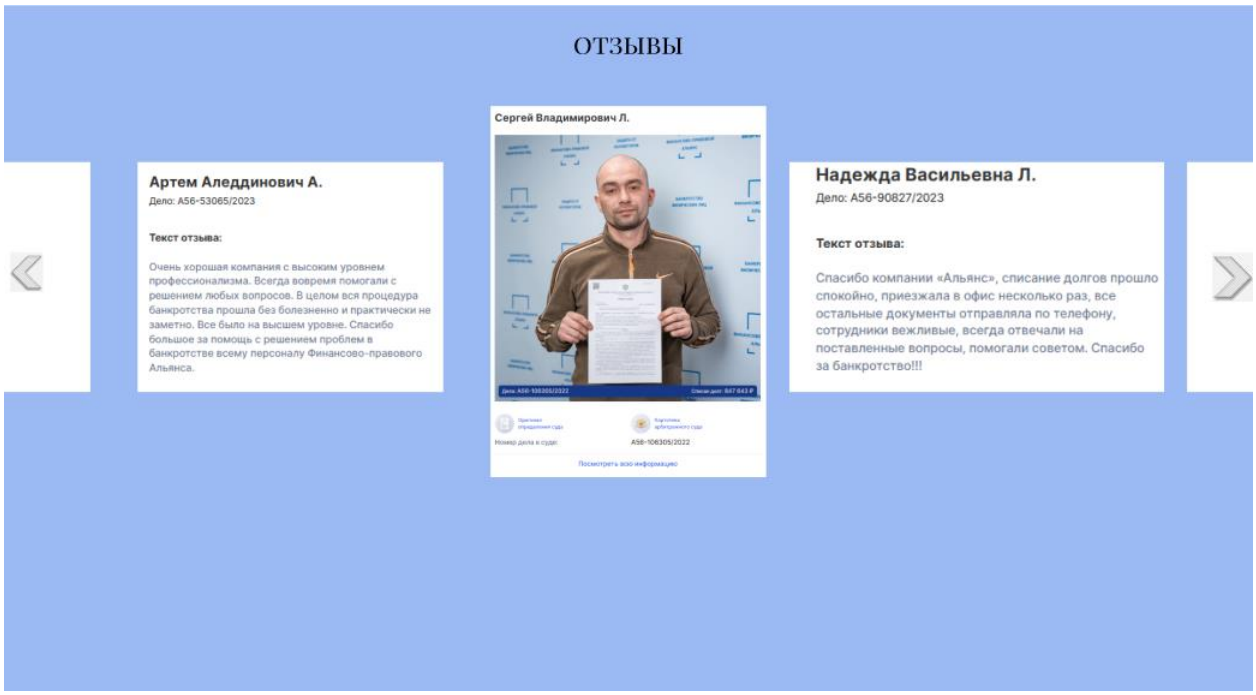


Рисунок Б4-отзывы


Рисунок Б5- обратная связь

**Практическая работа №9 «Работа с системой контроля версий Git»**

**Задание 1.**

Пройти 4 уровня из раздела «Введение», 4 уровня из раздела «Едем дальше» и 4 уровня на вкладке «Удаленные репозитории» из раздела «Push & Pull» в игре Learn Git Branching.

Решение уровней раздела «Введение»

Прошла 1 уровень раздела «Введение» (Рисунок 63).

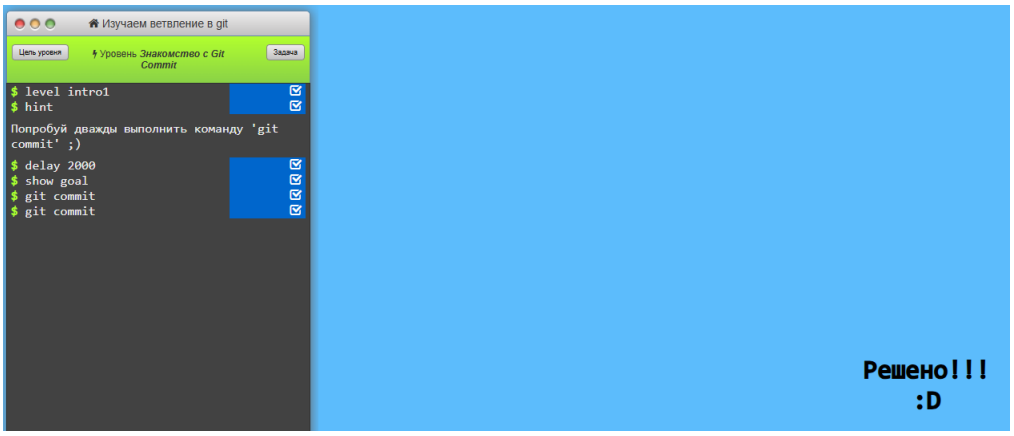


Рисунок 63

Используемые команды git commit, git commit (создание двух коммитов).

Прошла 2 уровень из раздела «Введение» (Рисунок 64).

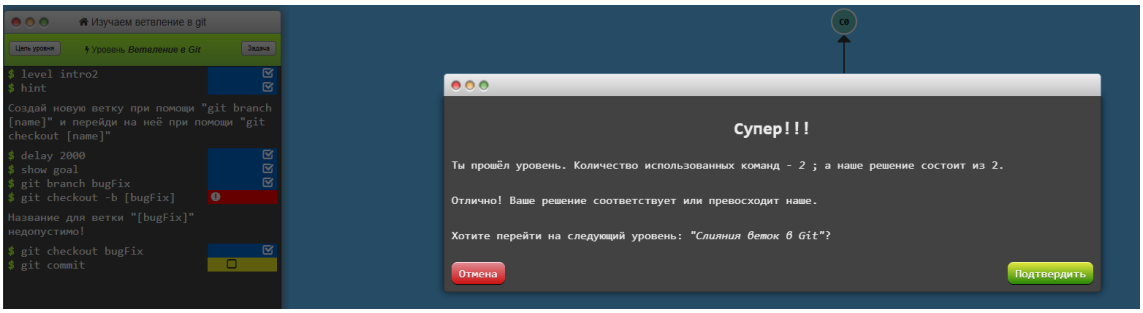


Рисунок 64

Используемые команды: git branch bugFix, git checkout bugFix (создала и переключилась на новую ветку bugFix).

Прошла 3 уровень из раздела «Введение» (Рисунок 65).

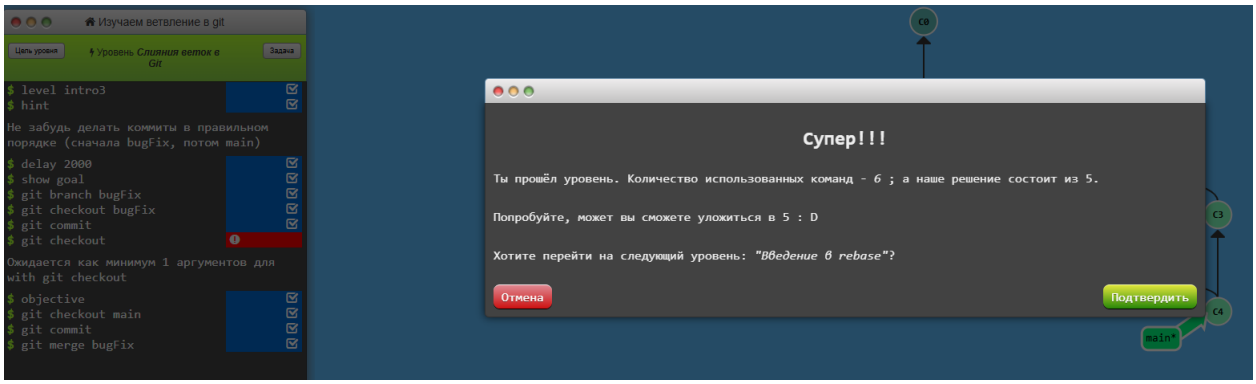


Рисунок 65

Используемые команды: git commit; git checkout main; git commit; git merge bugfix (Объединила изменения из двух разных веток. Использовала команду git merge).

Прошла 4 уровень из раздела «Введение» (Рисунок 66).

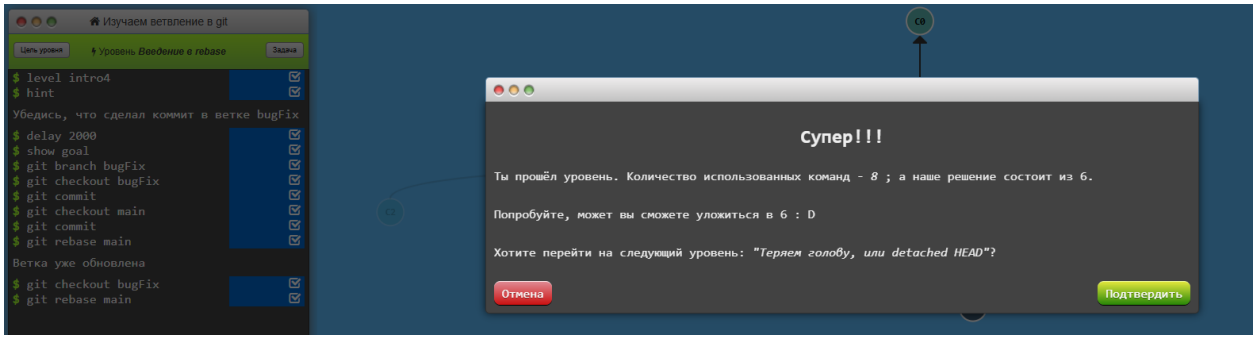


Рисунок 66

Используемые команды: git commit; git checkout main; git commit; git checkout bugfix; git rebase main (Второй способ объединения изменений в двух ветках. При «rebase» Git копирует набор коммитов и переносит их в другое место).

Раздел «Едем дальше»

Прошла 1 уровень из раздела «Едем дальше» (Рисунок 67).

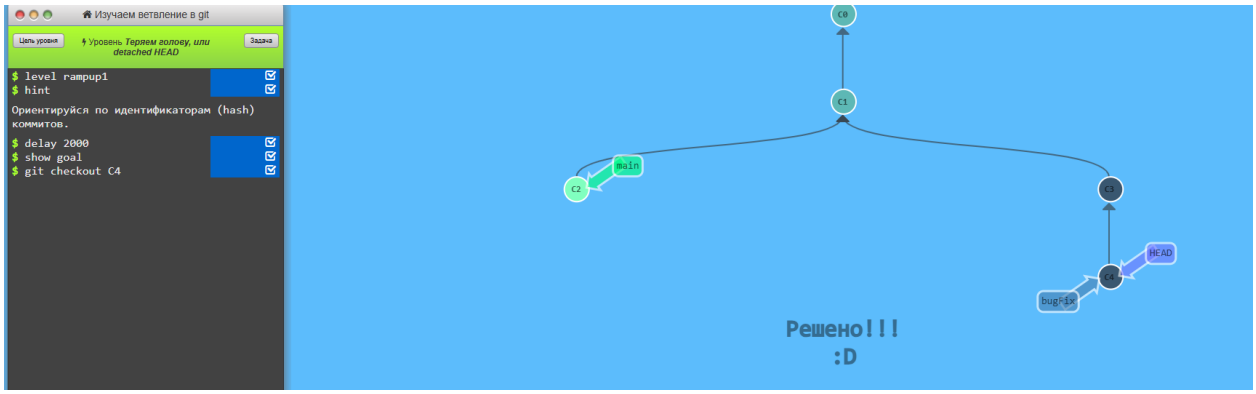


Рисунок 67

Используемые команды: git checkout C4 (Отделила HEAD от ветки bugFix и присвоила его последнему коммиту в этой же ветке).

Прошла 2 уровень из раздела «Едем дальше» (Рисунок 68).

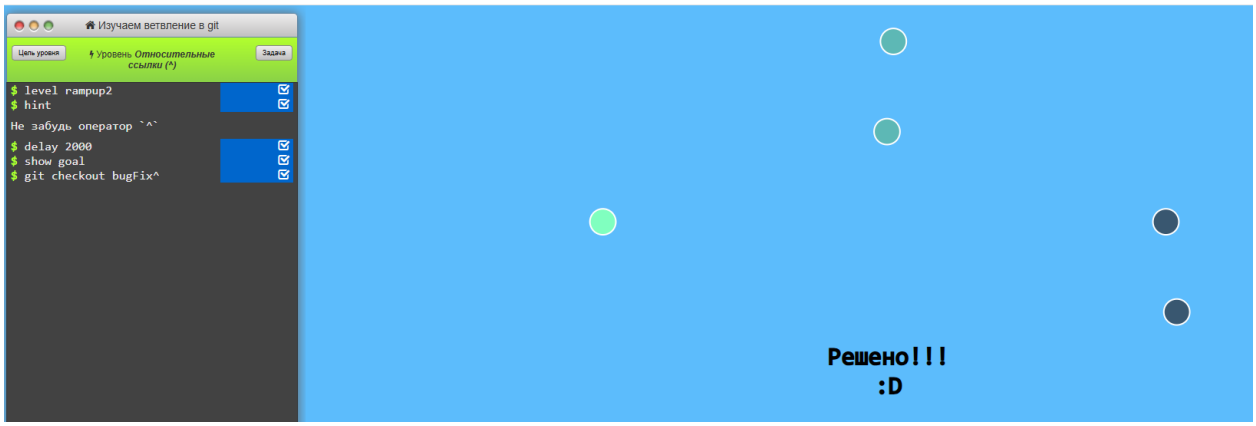


Рисунок 68

Используемые команды: git checkout HEAD^ (Переместила на первого родителя ветки).

Прошла 3 уровень из раздела «Едем дальше» (Рисунок 69).

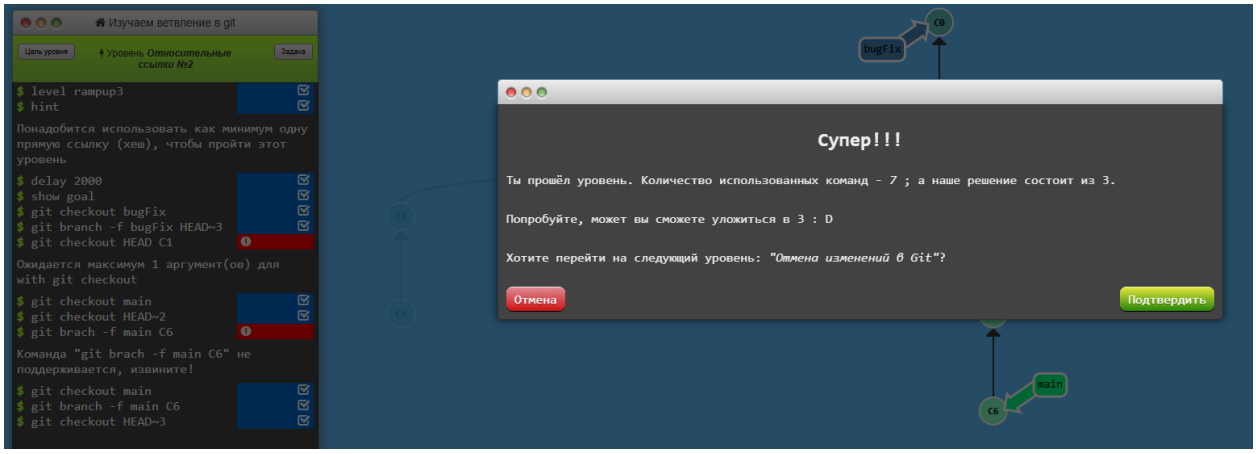


Рисунок 69

Используемые команды: git branch -f main C6; git checkout HEAD~1; git branch -f bugfix HEAD~1 (Создала ветку main с именем C6. Перешла к предыдущему коммиту, создала ветку bugFix с именем, соответствующим предыдущей ветке HEAD~1).

Прошла 4 уровень из раздела «Едем дальше» (Рисунок 70).

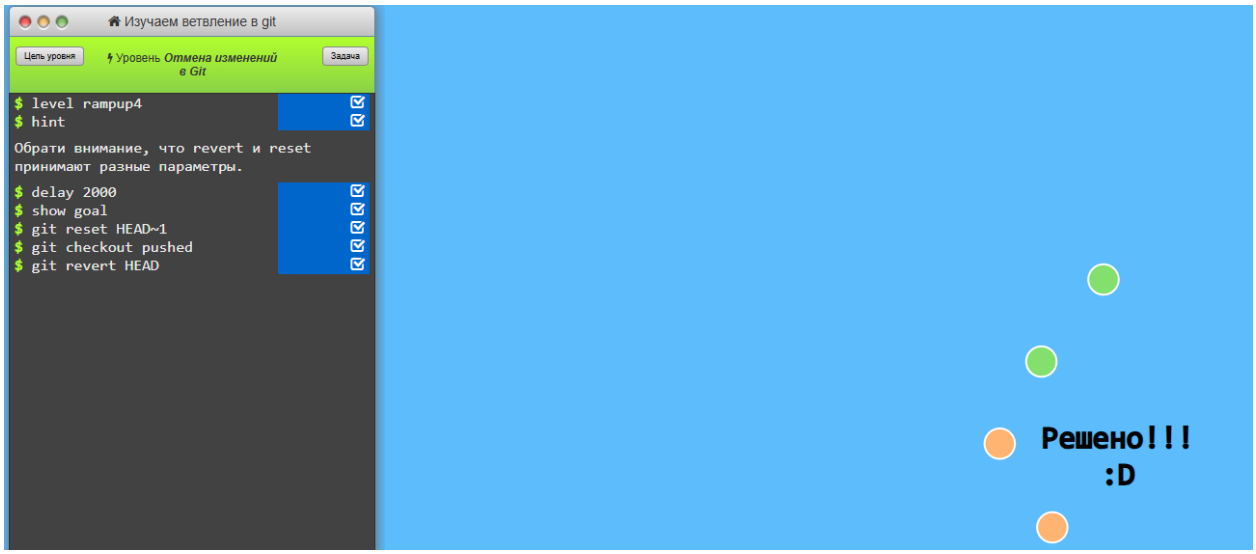


Рисунок 70

Используемые команды: git reset HEAD~1; git checkout pushed; git revert HEAD (Заменила файлы в рабочей копии на файлы предка переданного коммита, далее создала коммит, чтобы сохранить изменения, вернулась к прежнему состоянию файлов в рабочей области).

Вкладка «Основы» завершена (по заданию) (Рисунок 71).

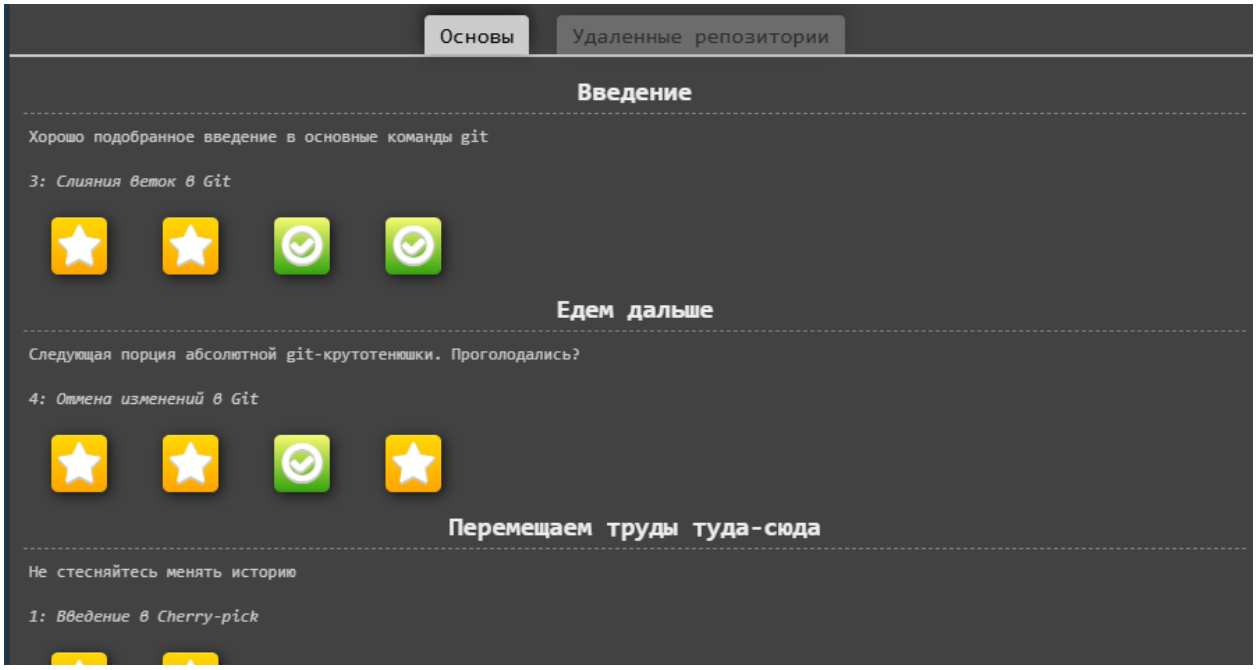


Рисунок 71

Раздел «Push&Pull» на вкладке «Удаленные репозитории»

Прошла 1 уровень из раздела «Push&Pull» (Рисунок 72).

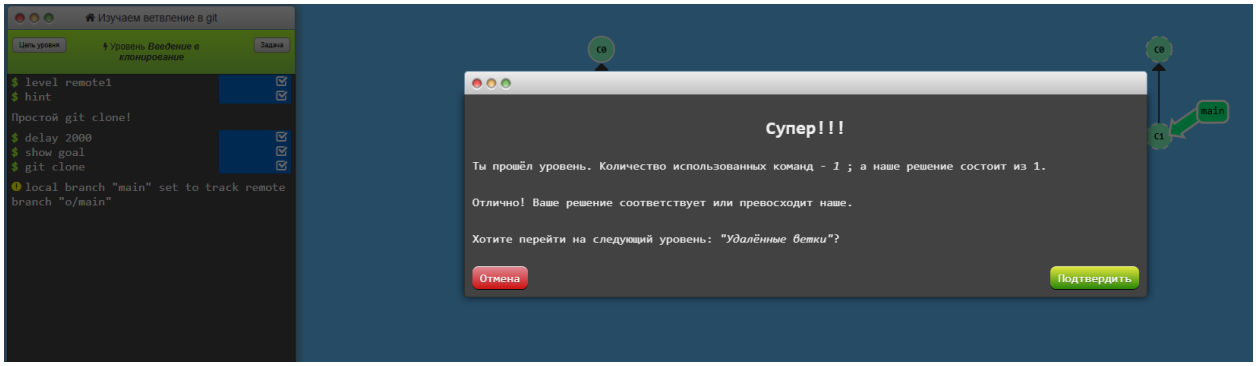


Рисунок 72

Используемые команды: git clone (Создала клон репозитория).

Прошла 2 уровень из раздела «Push&Pull» (Рисунок 73).

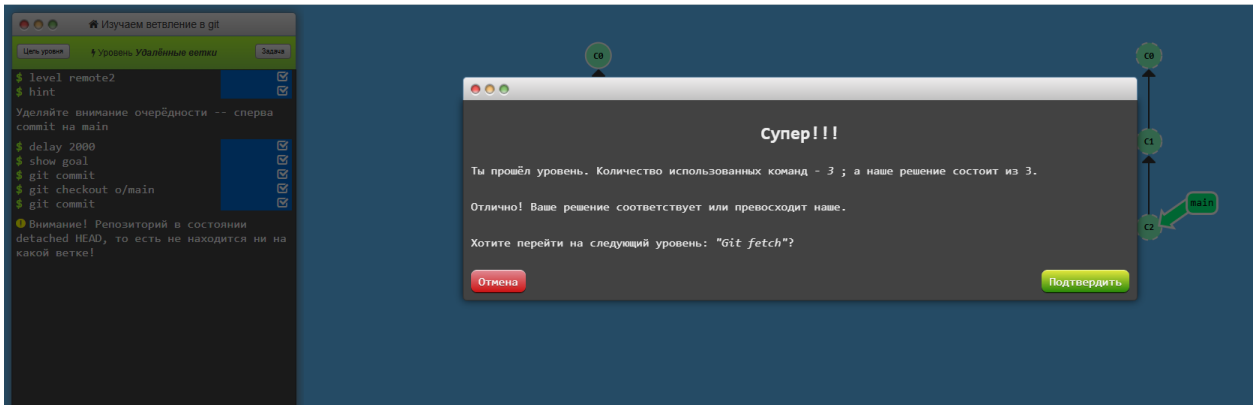


Рисунок 73

Используемые команды: git commit; git checkout o/main; git commit (Создала коммит, перешла к удаленным веткам, создала коммит).

Прошла 3 уровень из раздела «Push&Pull» (Рисунок 74).

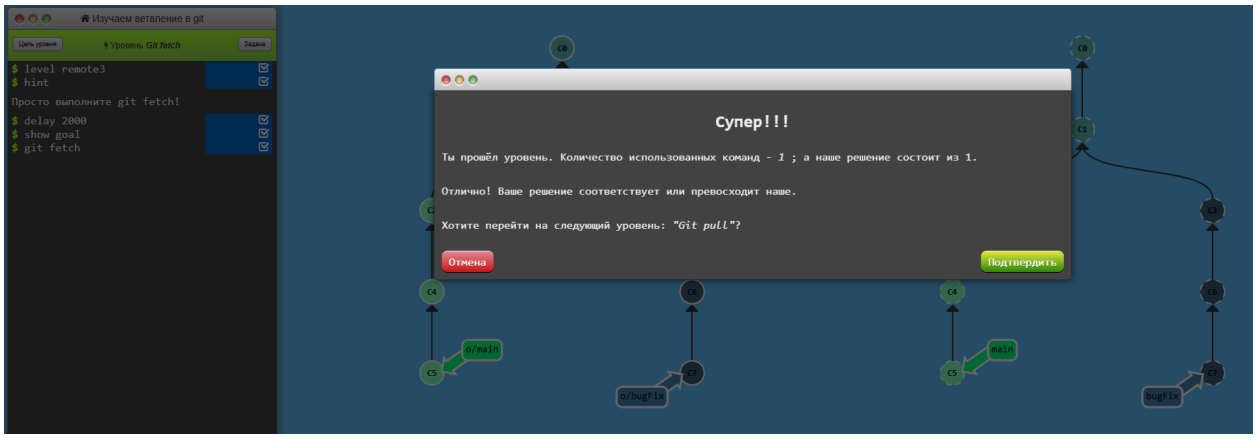


Рисунок 74

Используемые команды: git fetch (Загрузила актуальные данные из удалённого репозитория в локальное хранилище).

Прошла 4 уровень из раздела «Push&Pull» (Рисунок 75).

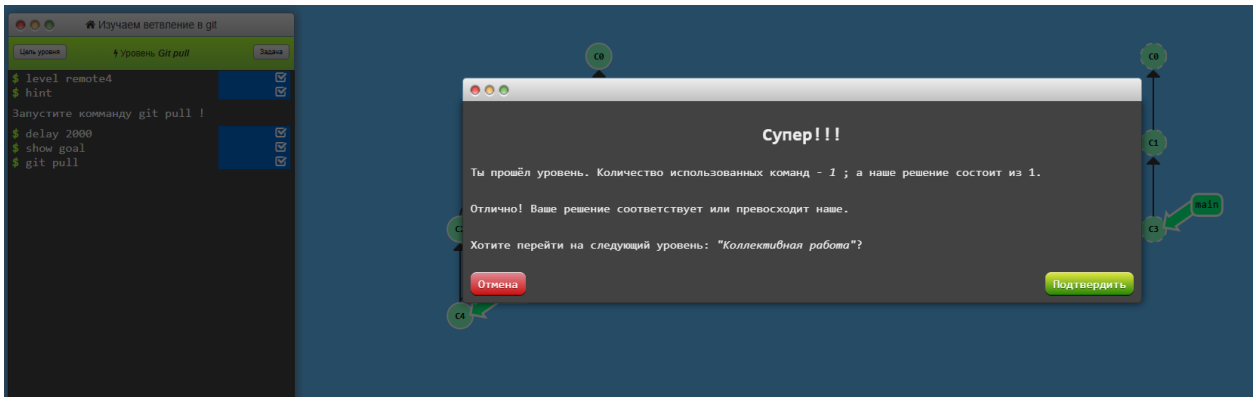


Рисунок 75

Используемые команды: git pull (Загрузила последние изменения из удалённого репозитория, и команда автоматически слила их с текущей локальной веткой).

**Задание 2.**