

# Основы программирования

ФИСТ 1 курс

Власенко

Олег

Федосович

Лекция 2

Создание проекта в VS.

Цикл do ... while.

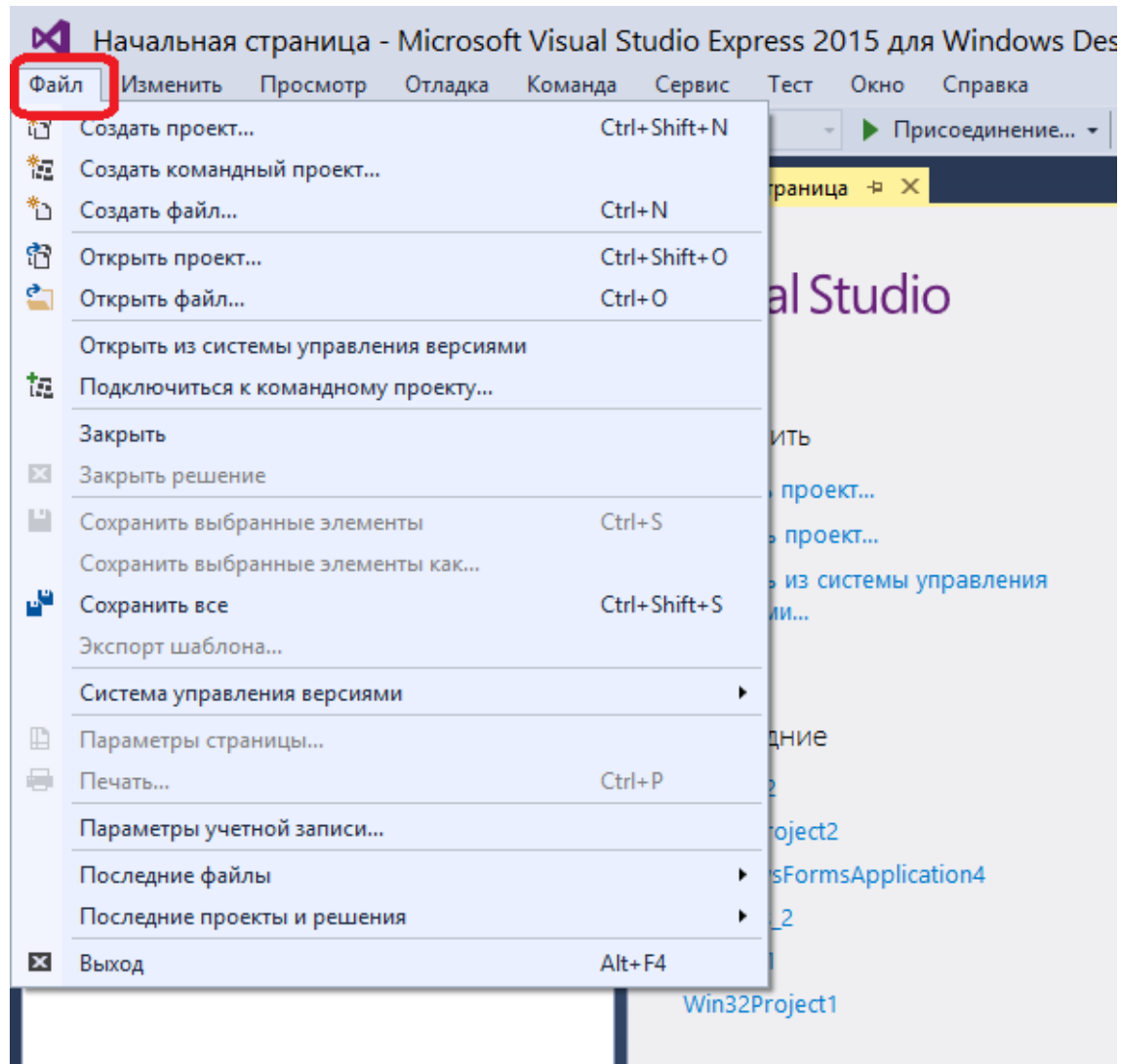
Приложение WinAPI.

Знакомство с графикой в WinAPI: рисование линий, прямоугольников,  
эллипсов.

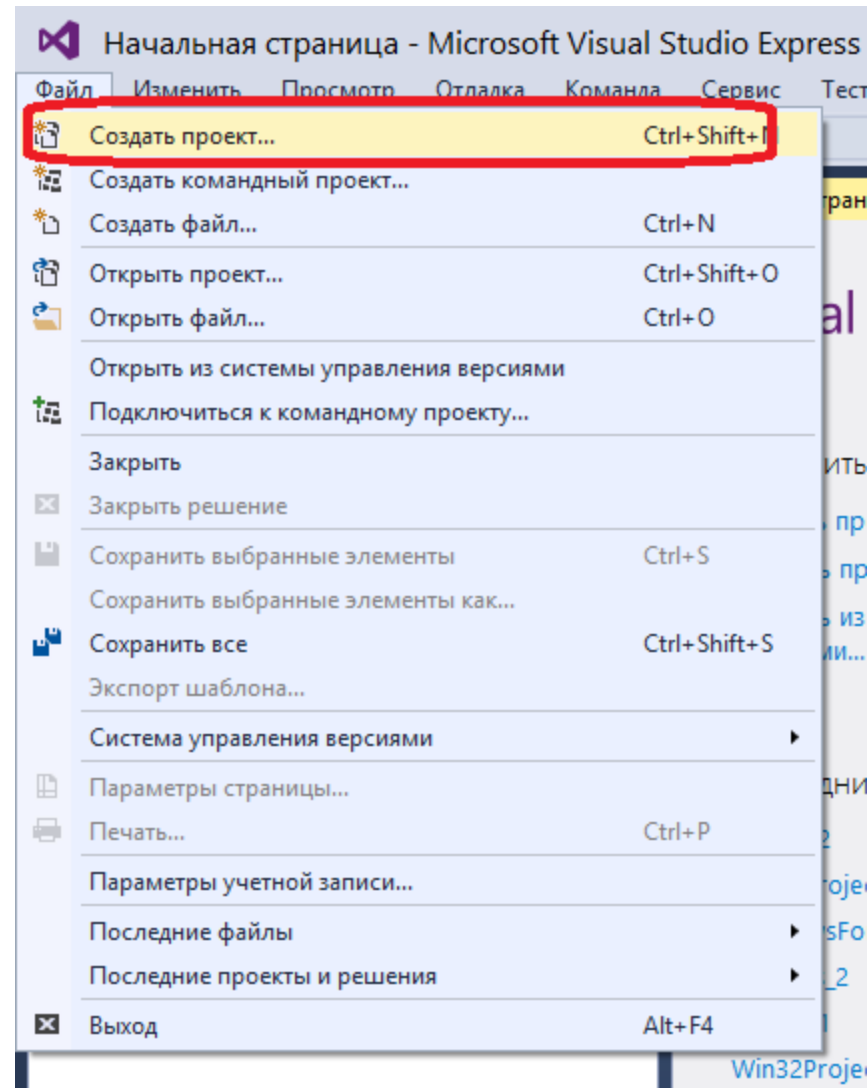
Рисование рисунков по вычисленным координатам.

Стили линий, стили заливки. Struct.

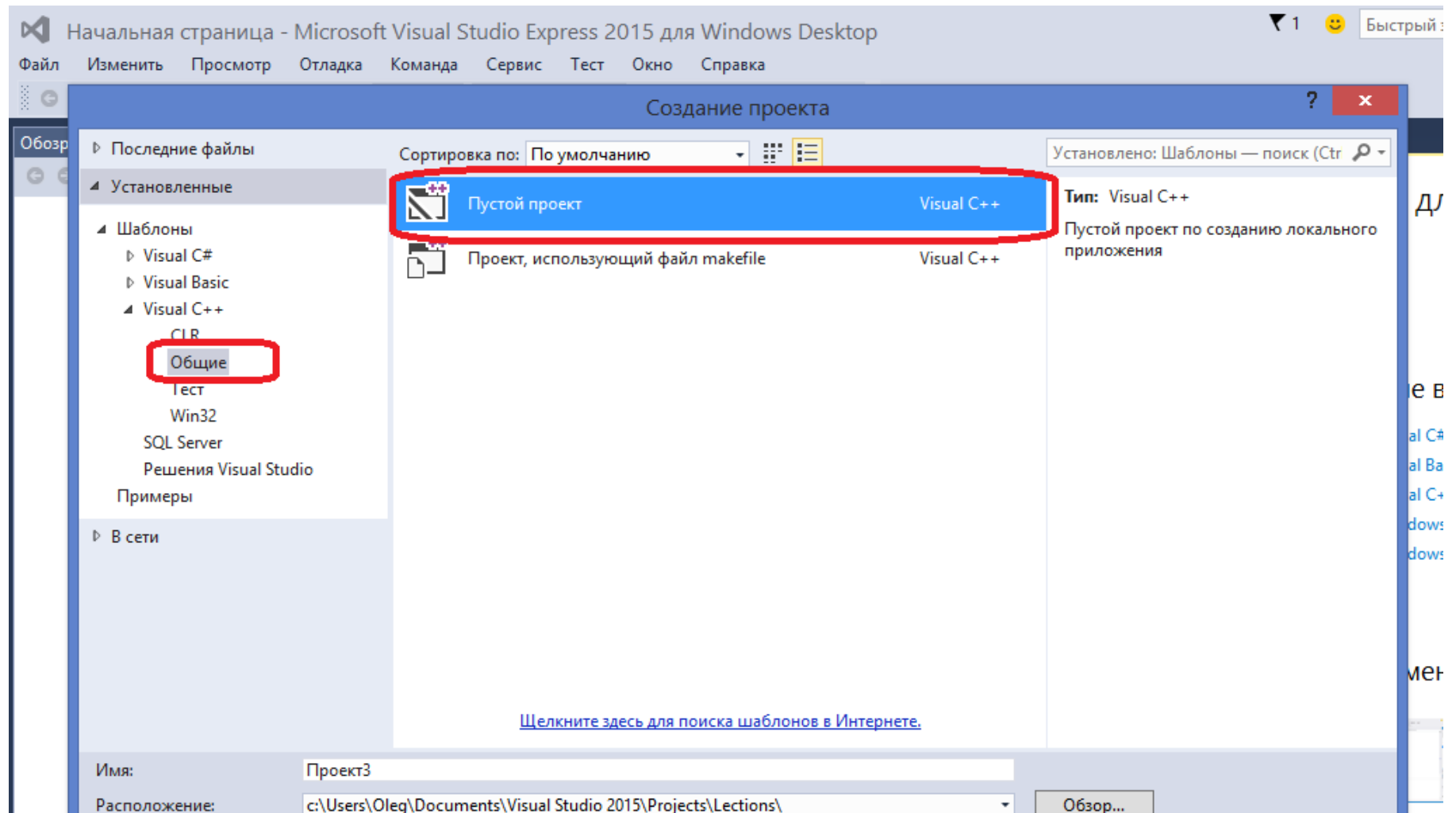
# Создание нового проекта в VS (1)



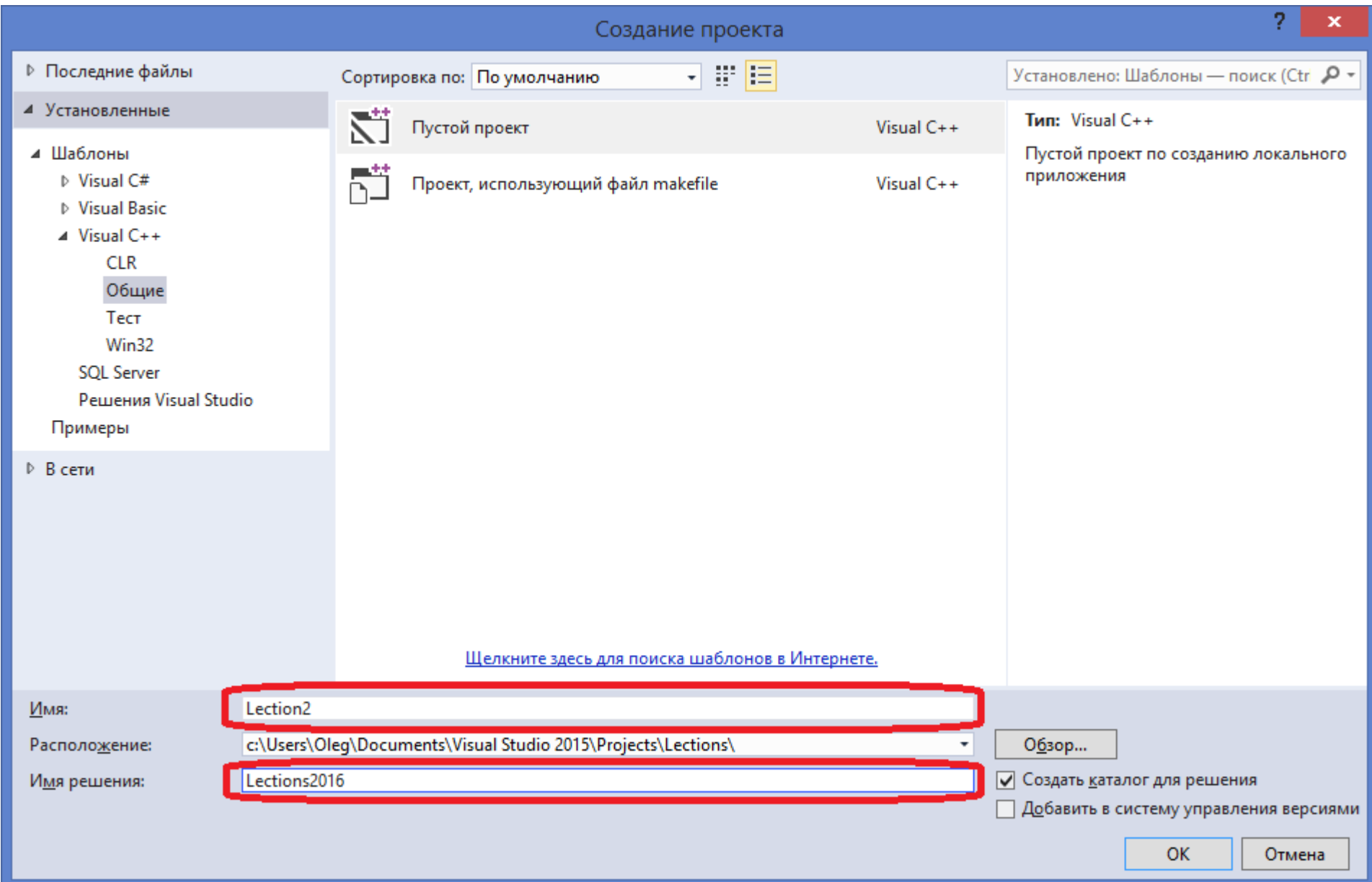
# Создание нового проекта в VS (2)



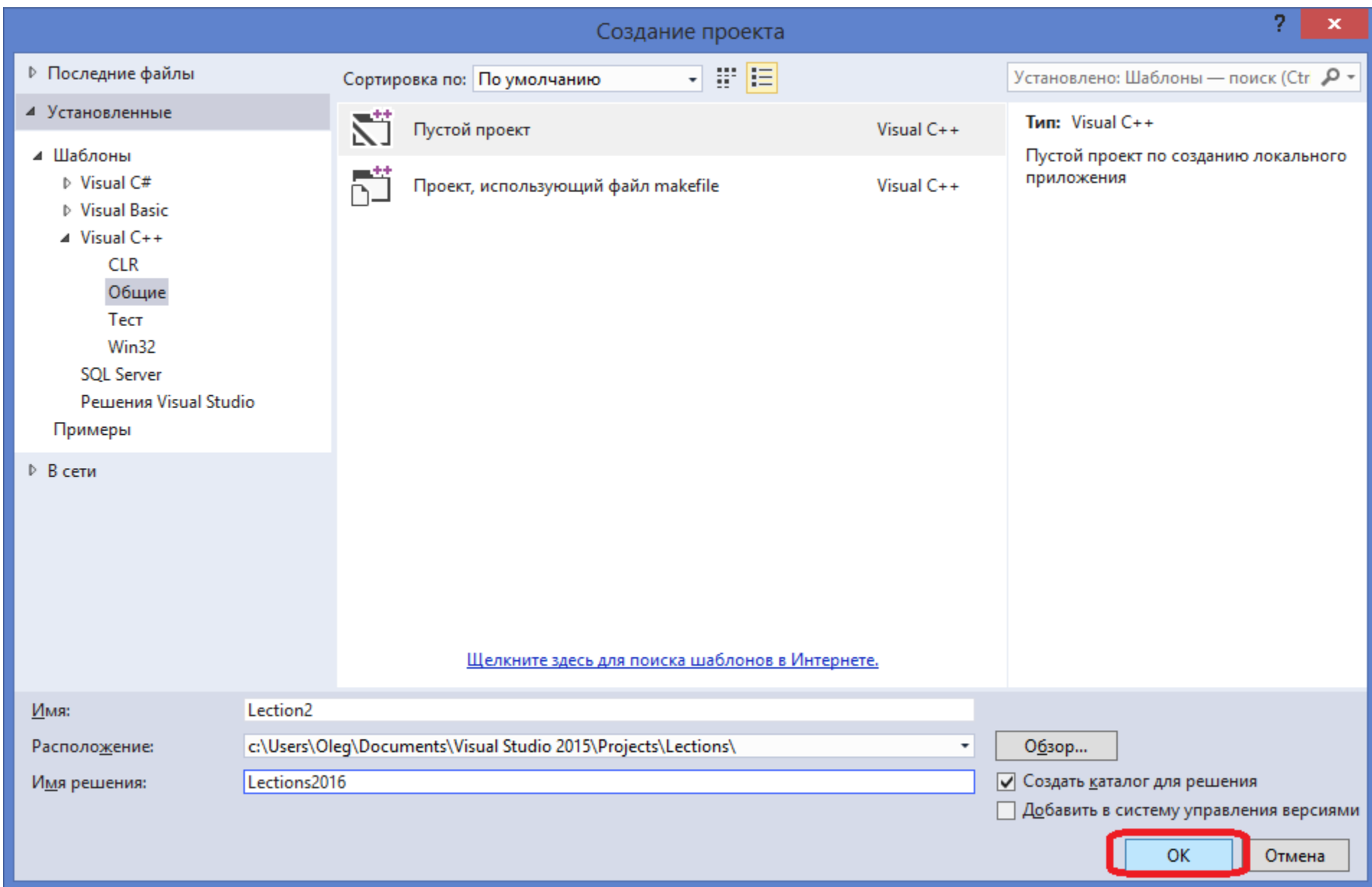
# Создание нового проекта в VS (3)



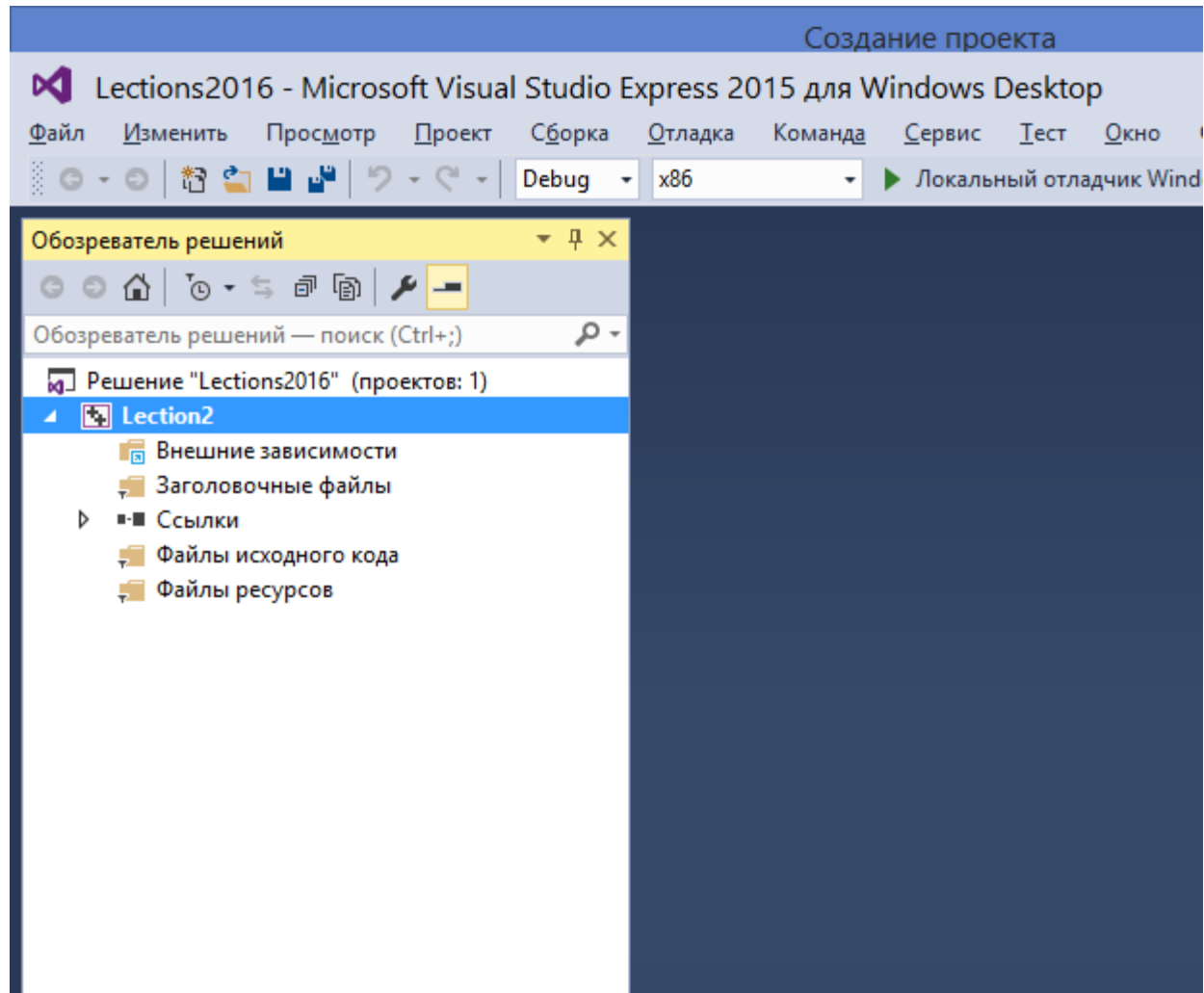
# Создание нового проекта в VS (4)



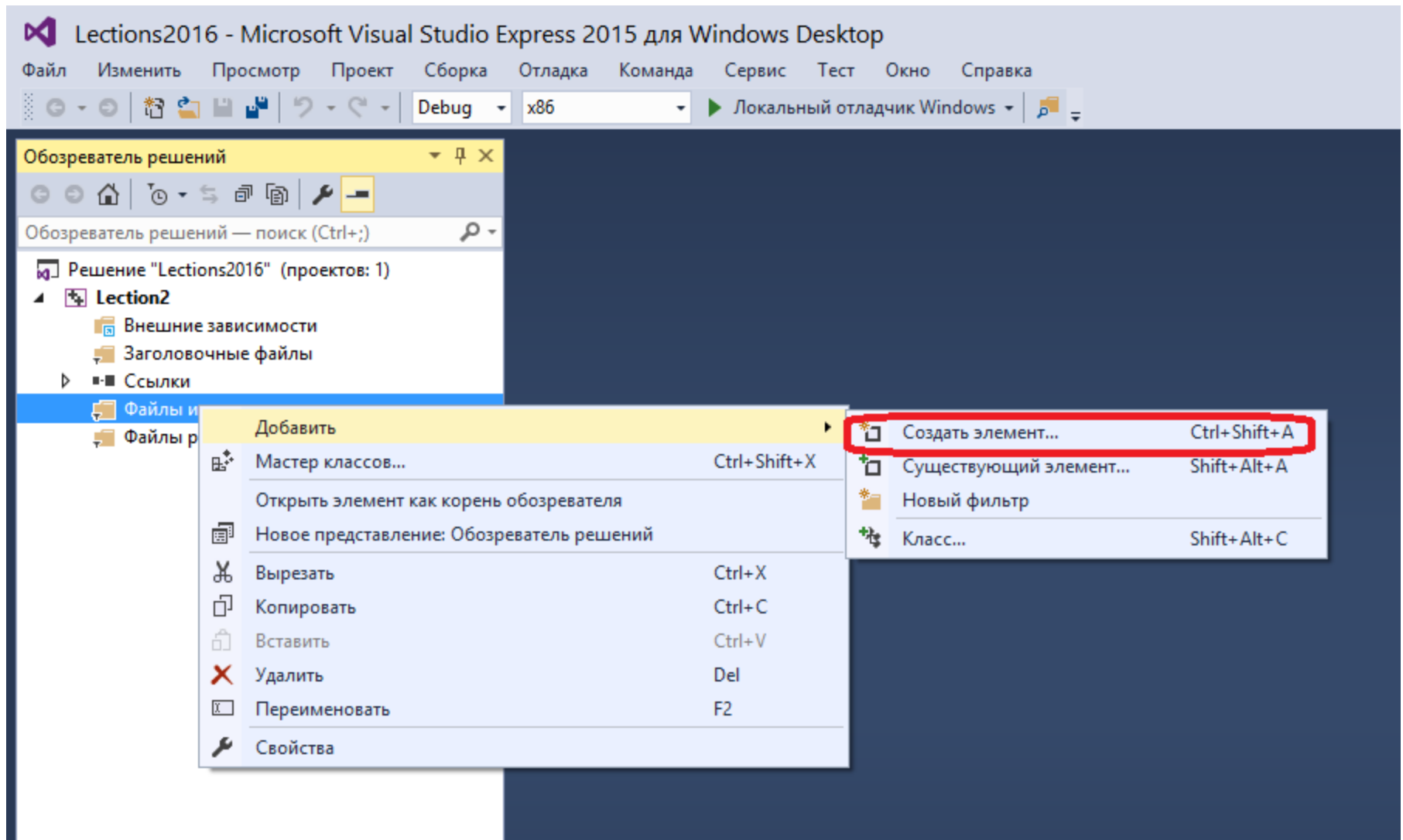
# Создание нового проекта в VS (5)



# Создание нового проекта в VS (6) – проект создан!

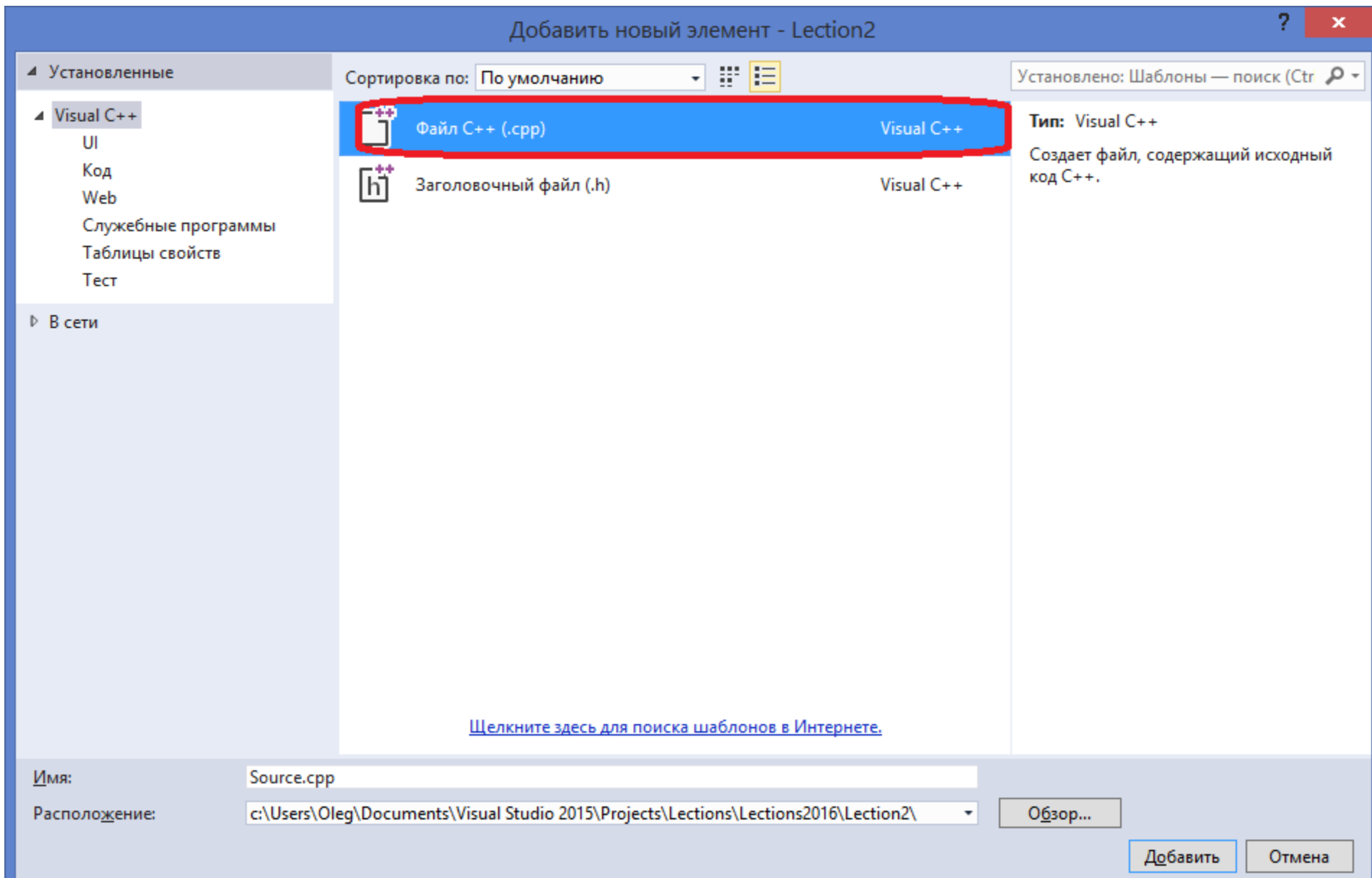


# Создание нового файла с кодом (1)

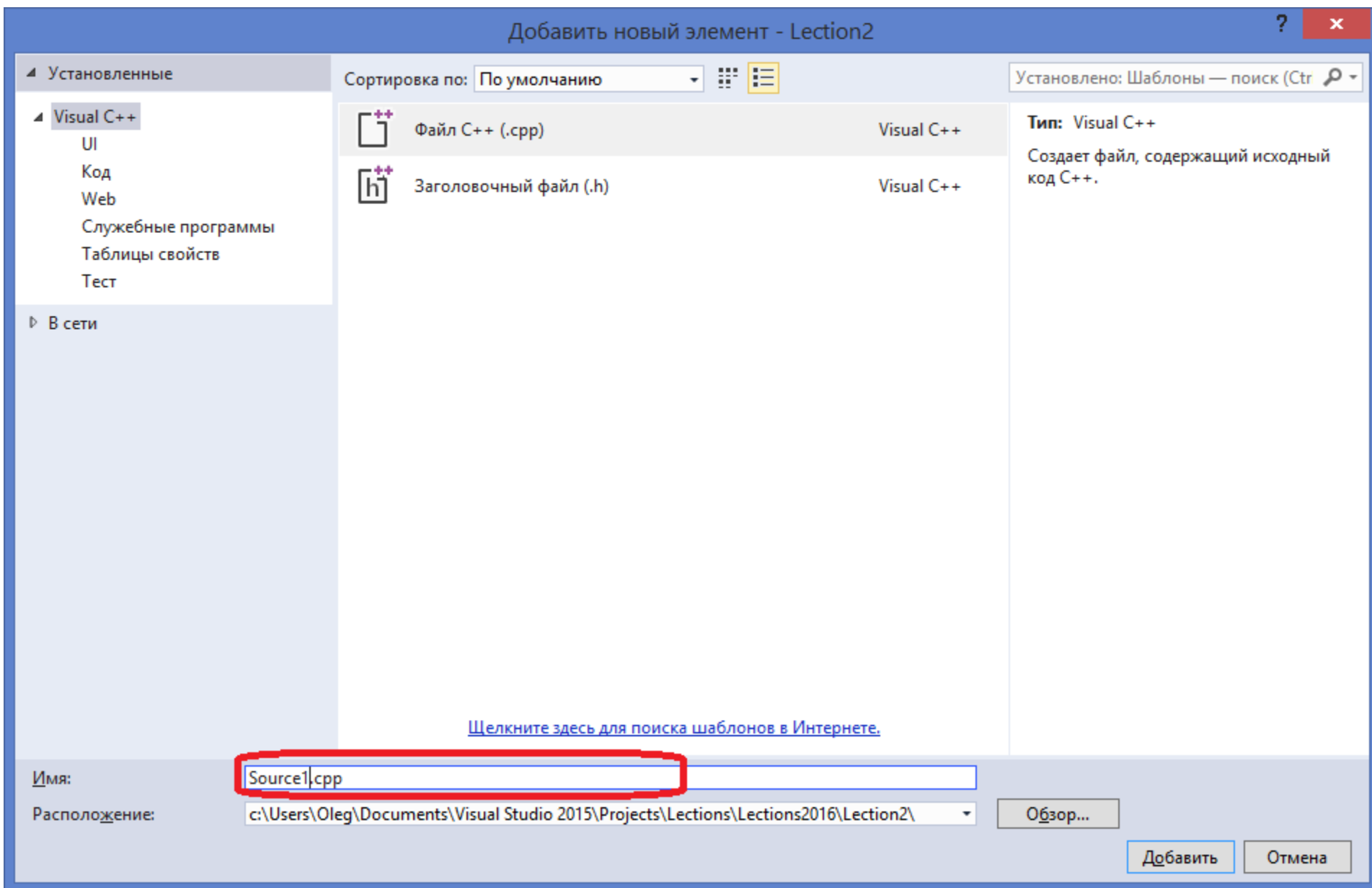




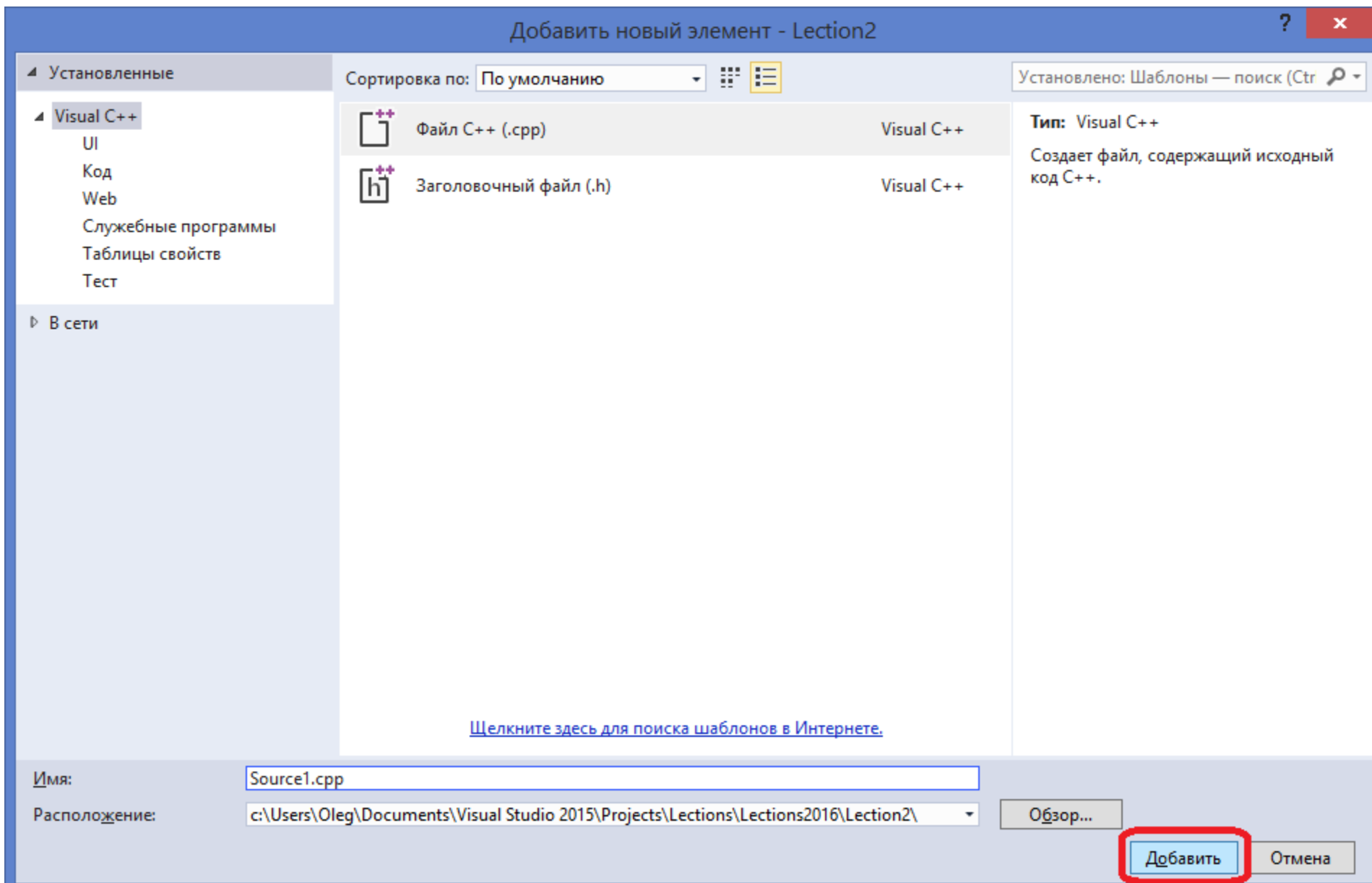
# Создание нового файла с кодом (2)



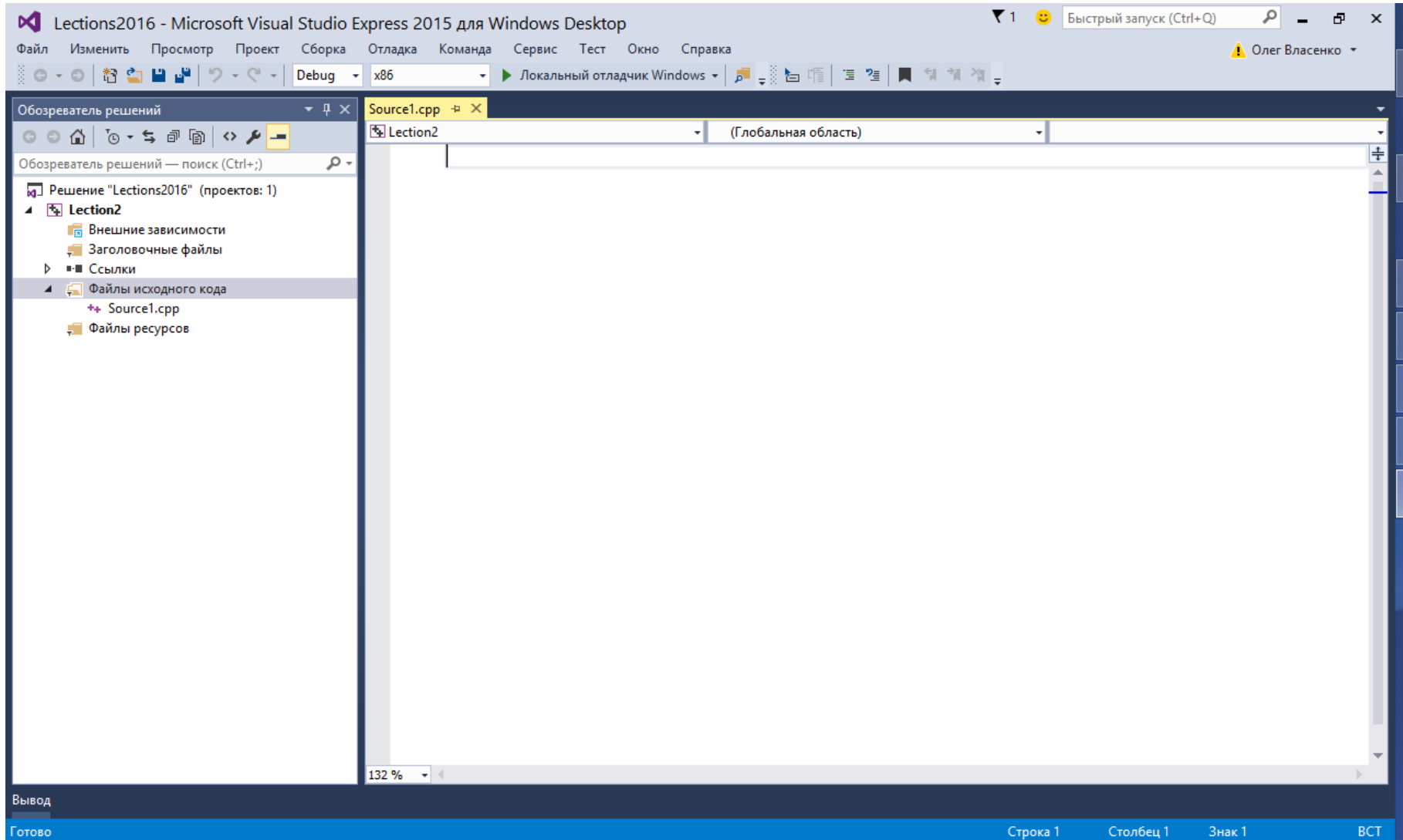
# Создание нового файла с кодом (3)



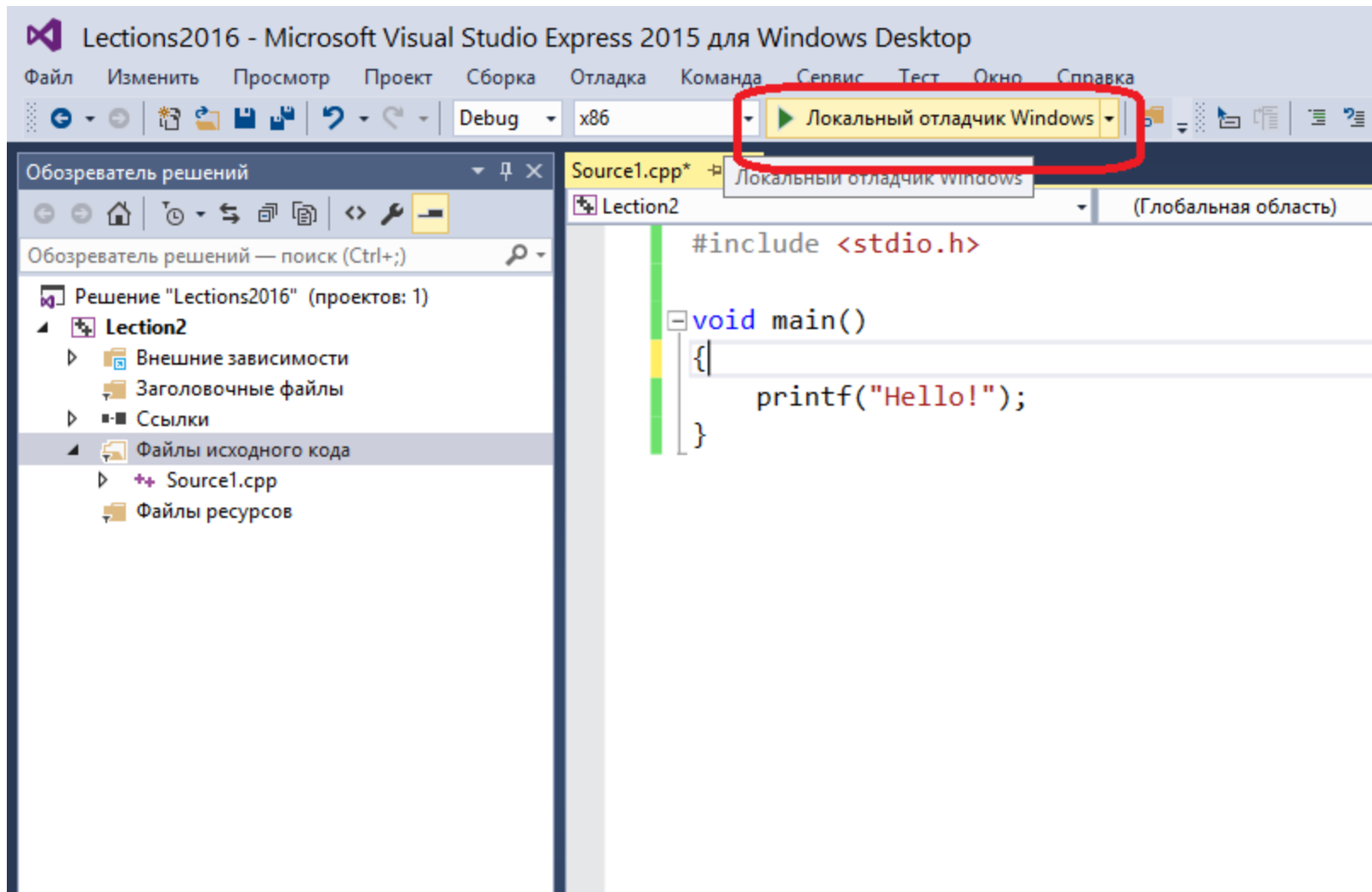
# Создание нового файла с кодом (4)



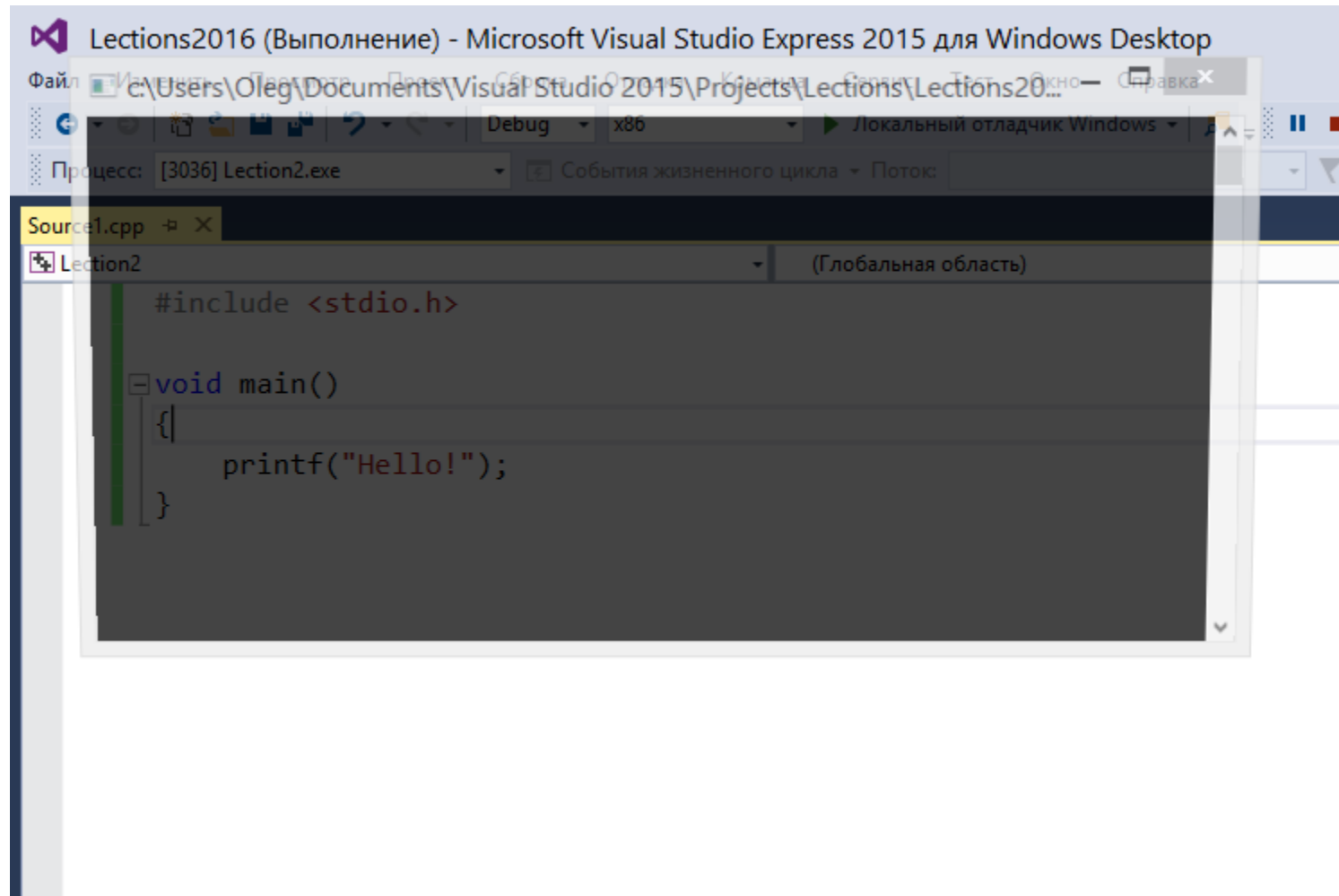
# Создание нового файла с кодом (5) – создано!



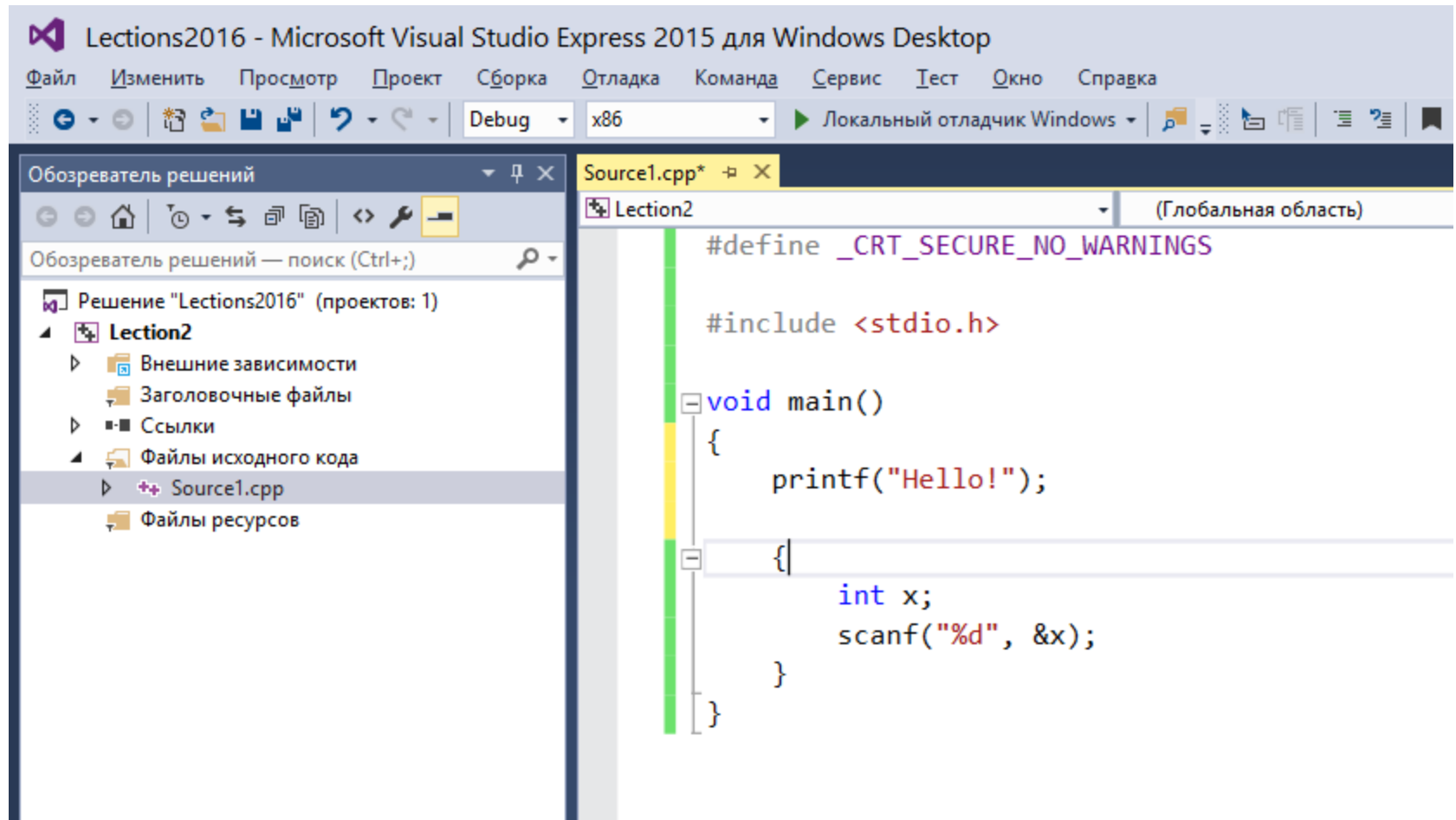
# Набор текста программы и запуск



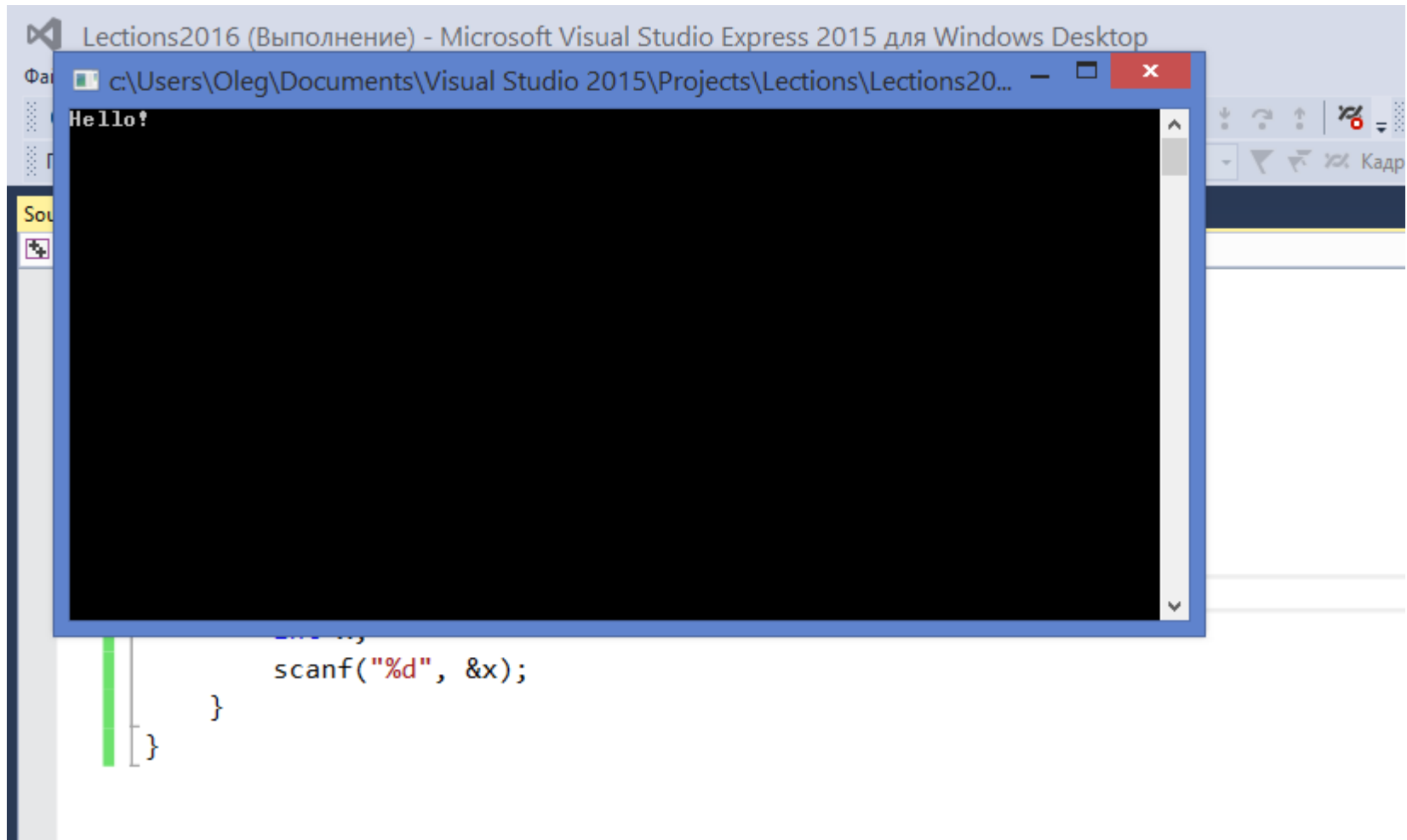
# Набор текста программы и запуск (2)



# Программа, которая ждет ввода



# Программа, которая ждет ввода (2)





# Задача 1

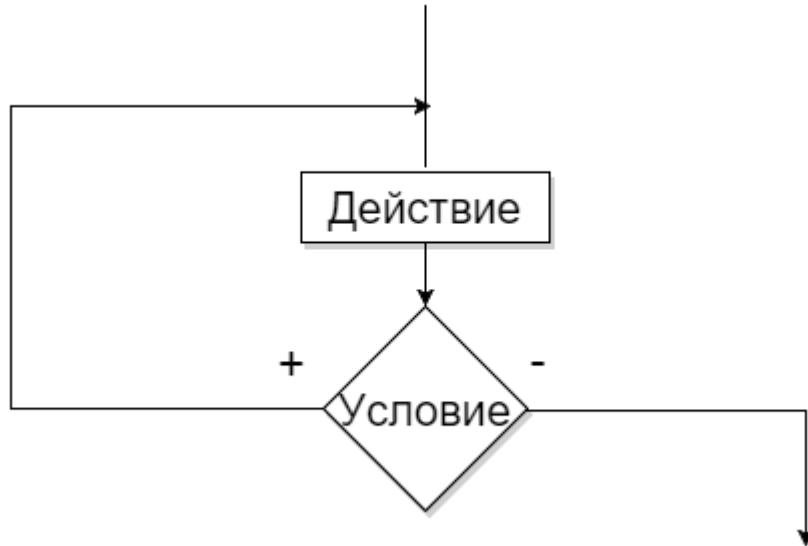
Создать программу, которая подсчитывает сумму введенных чисел.

Все числа положительные целые. Завершение ввода – 0.

Пример ввода: 10 20 25 0

Вывод: 55

# Цикл с постусловием do while

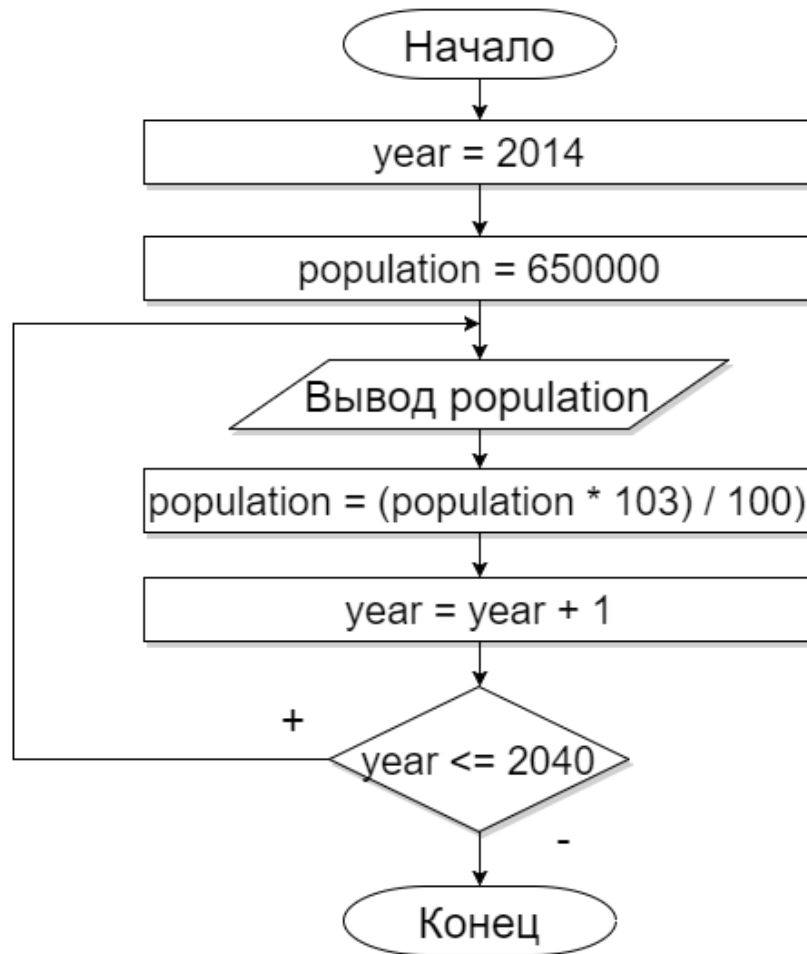


```
do {  
    Действие;  
} while (Условие);
```

# Пример для цикла do while

Население города увеличивается на 3% каждый год. В 2014 году население города составляло 650 000 человек. Напишите программу, которая выведет на экран предсказываемую численность населения города в каждом году, вплоть до 2040.

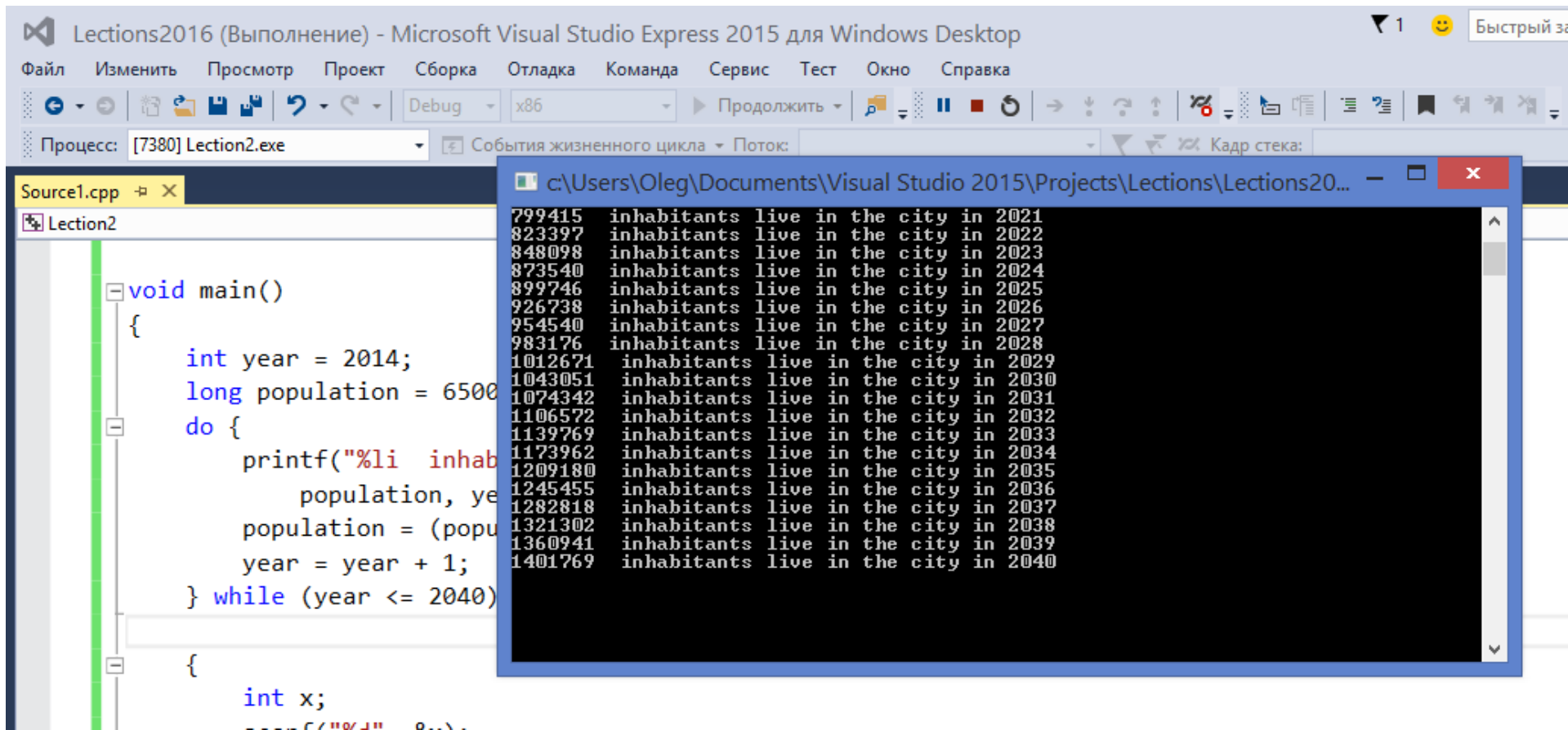
# Блок-схема



# Программа

```
void main() {  
    int year = 2014;  
    long population = 650000;  
    do {  
        printf("%li inhabitants live in the city in %i\n",  
               population, year);  
        population = (population * 103) / 100;  
        year = year + 1;  
    } while (year <= 2040);  
}
```

# Программа в работе



The screenshot shows the Microsoft Visual Studio Express 2015 IDE. The main window displays the source code for `Source1.cpp`, which contains a `main` function. The code calculates the population of a city from 2014 to 2040, assuming a constant growth rate. The output window shows the results for each year.

```
void main()
{
    int year = 2014;
    long population = 6500;
    do {
        printf("%li inhabitants live in the city in %d\n", population, year);
        population = (population * 1.02);
        year = year + 1;
    } while (year <= 2040)
```

The output window displays the following results:

Year	Population
2021	799415
2022	823397
2023	848098
2024	873540
2025	899746
2026	926738
2027	954540
2028	983176
2029	1012671
2030	1043051
2031	1074342
2032	1106572
2033	1139769
2034	1173962
2035	1209180
2036	1245455
2037	1282818
2038	1321302
2039	1360941
2040	1401769

## Задача 1 (2)

Создать программу, которая подсчитывает сумму введенных чисел.

Все числа положительные целые. Завершение ввода – 0.

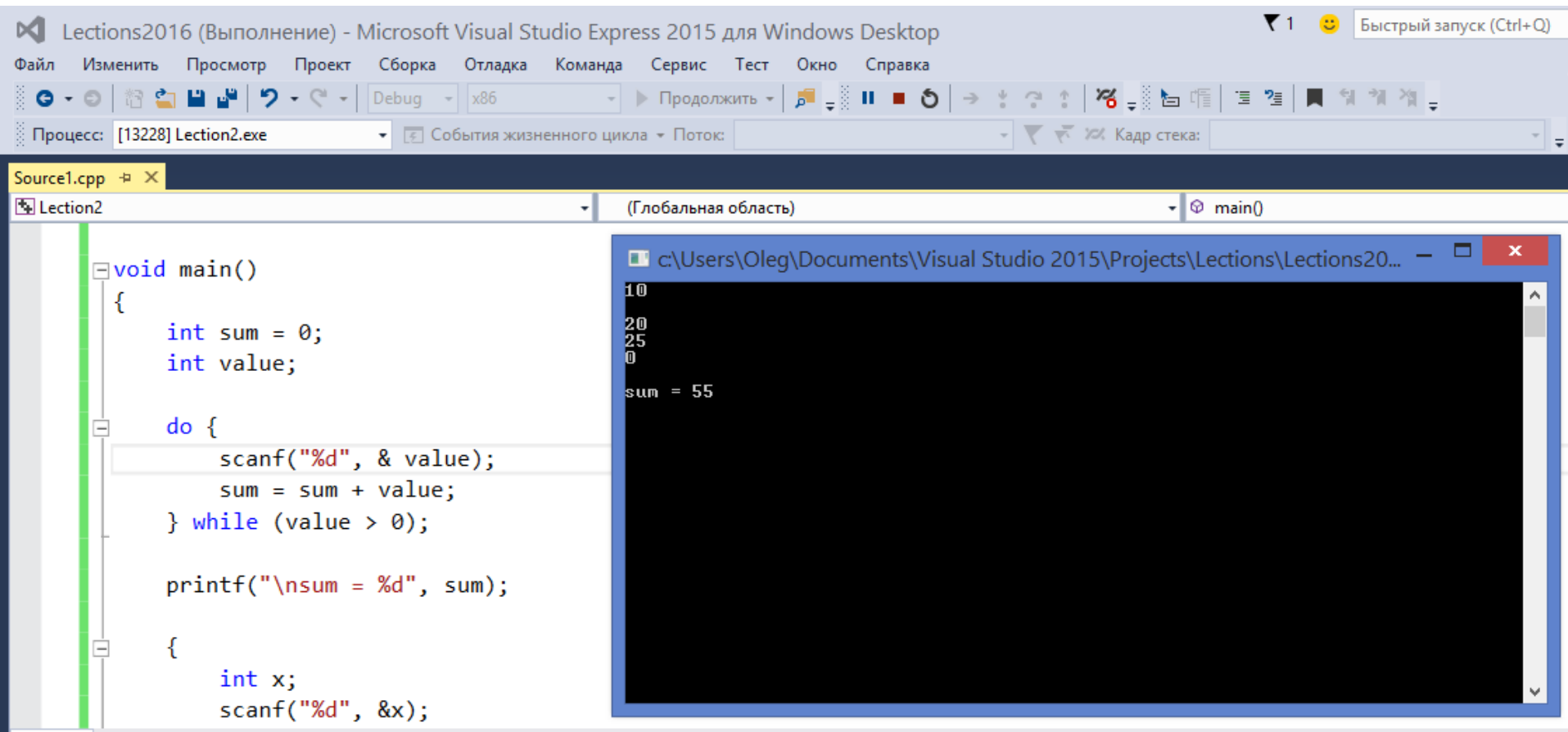
Пример ввода: 10 20 25 0

Вывод: 55

**Нужно использовать цикл do while для реализации.**

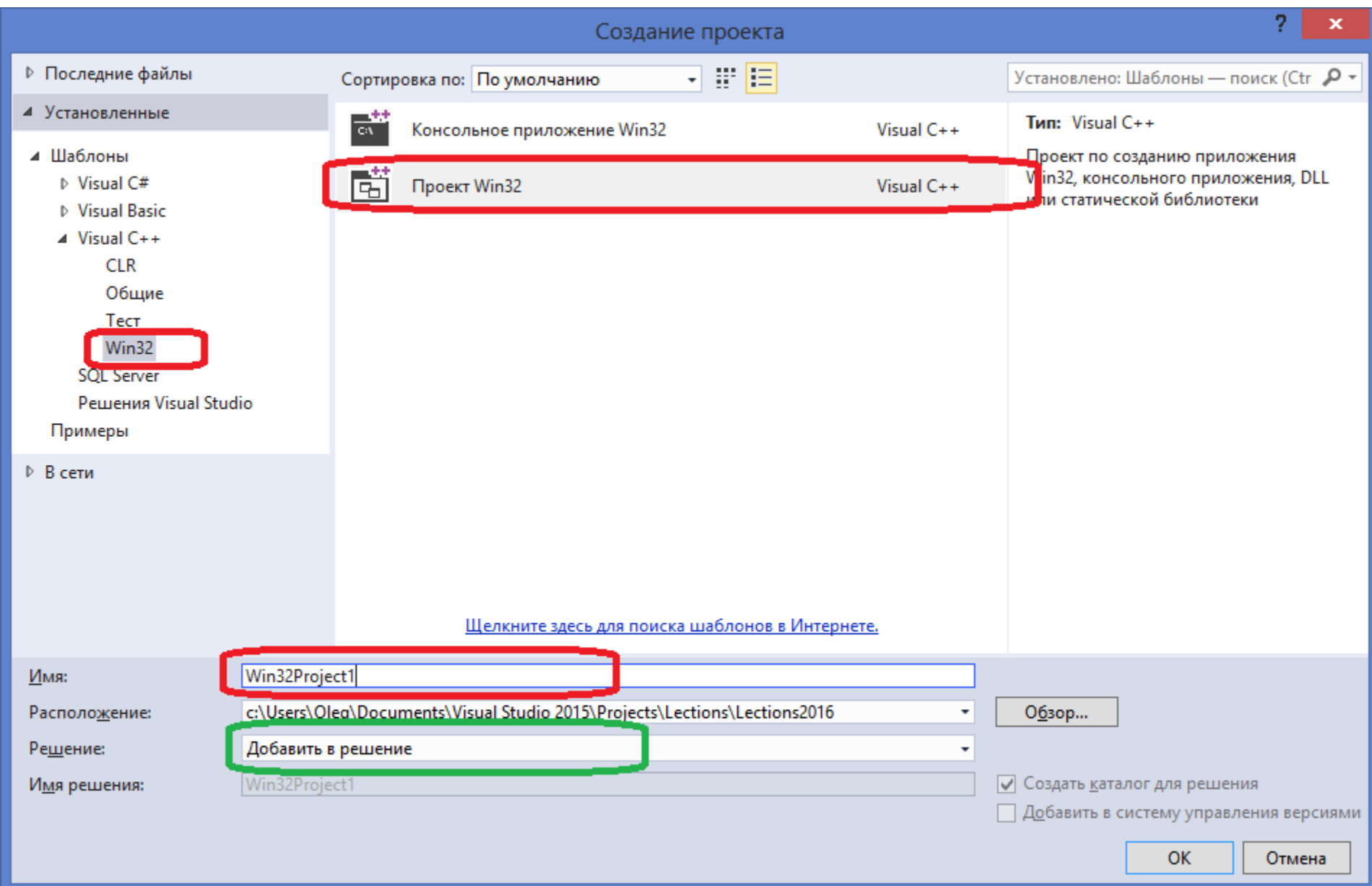
```
void main()
{
    int sum = 0;
    int value;
    do {
        ???
    } while (value > 0);
    printf("\nsum = %d", sum);
}
```

# Задача 1 (3): программа в работе

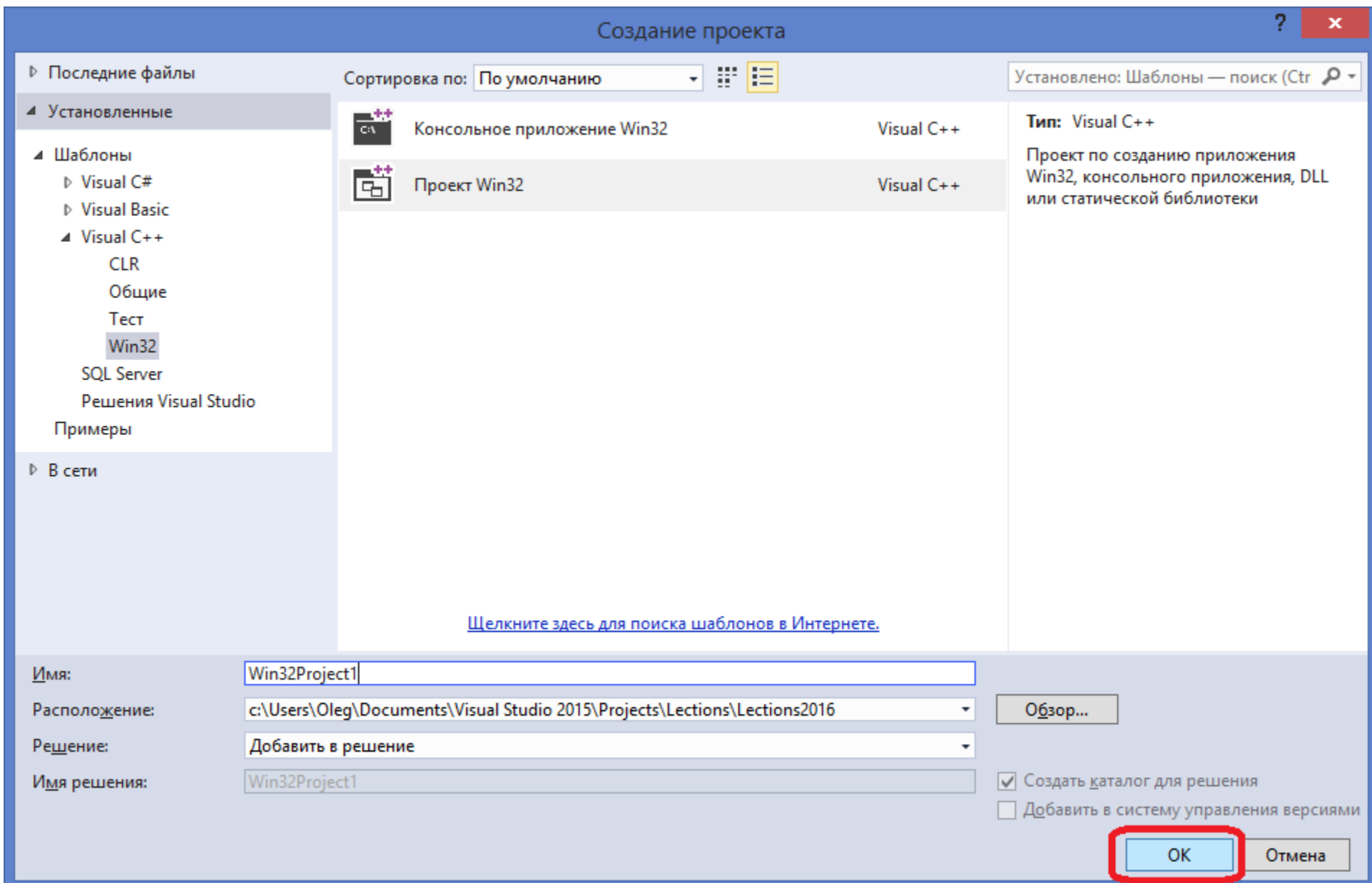




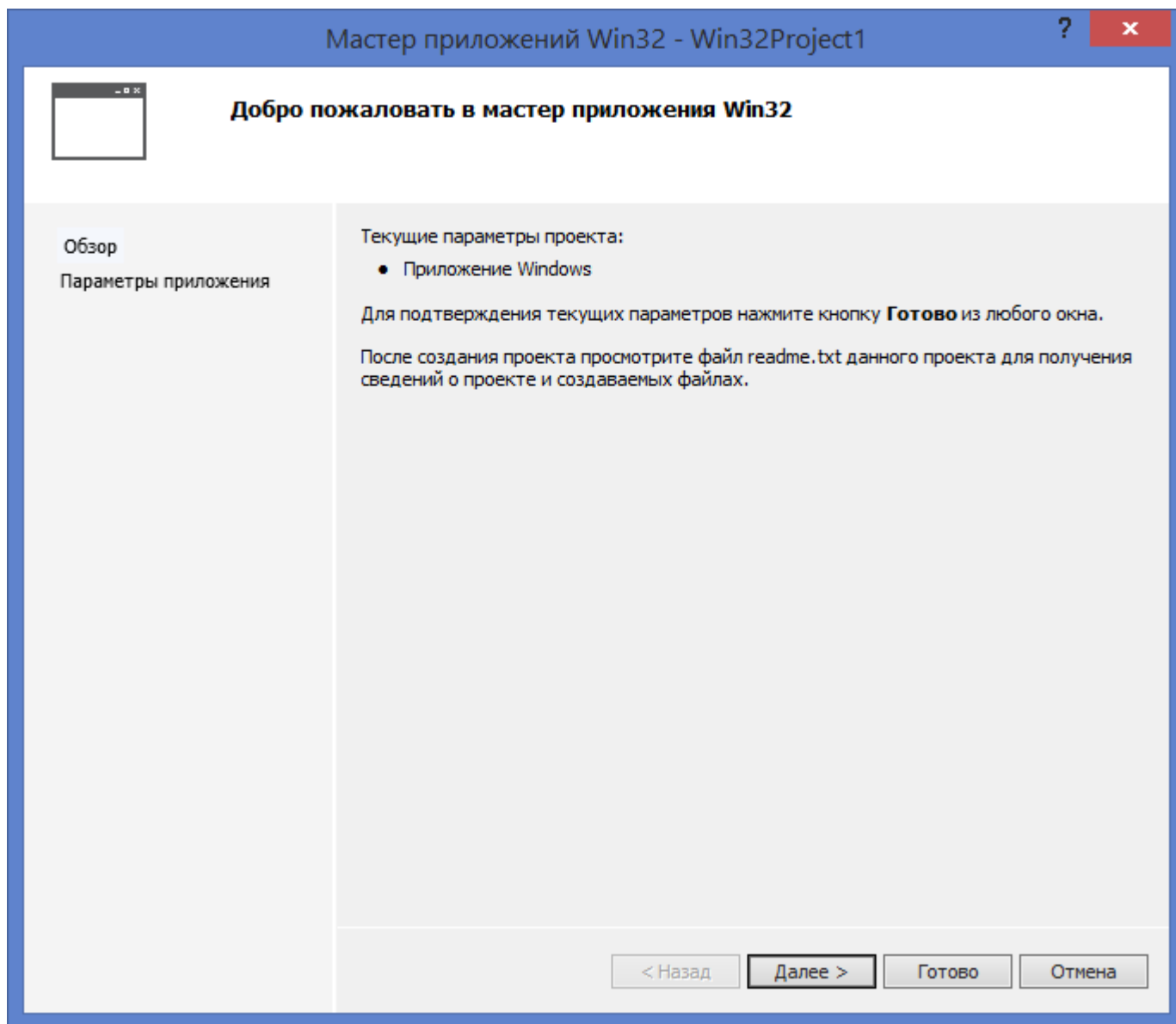
# Создание win32 приложения в VS



# Создание win32 приложения в VS (2)



# Создание win32 приложения в VS (3)



# Создание win32 приложения в VS (4)

Мастер приложений Win32 - Win32Project1

Параметры приложения

Обзор  
Параметры приложения

Тип приложения:

- ☒ Приложение Windows
- ☐ Консольное приложение
- ☐ Библиотека DLL
- ☐ Статическая библиотека

Дополнительные параметры:

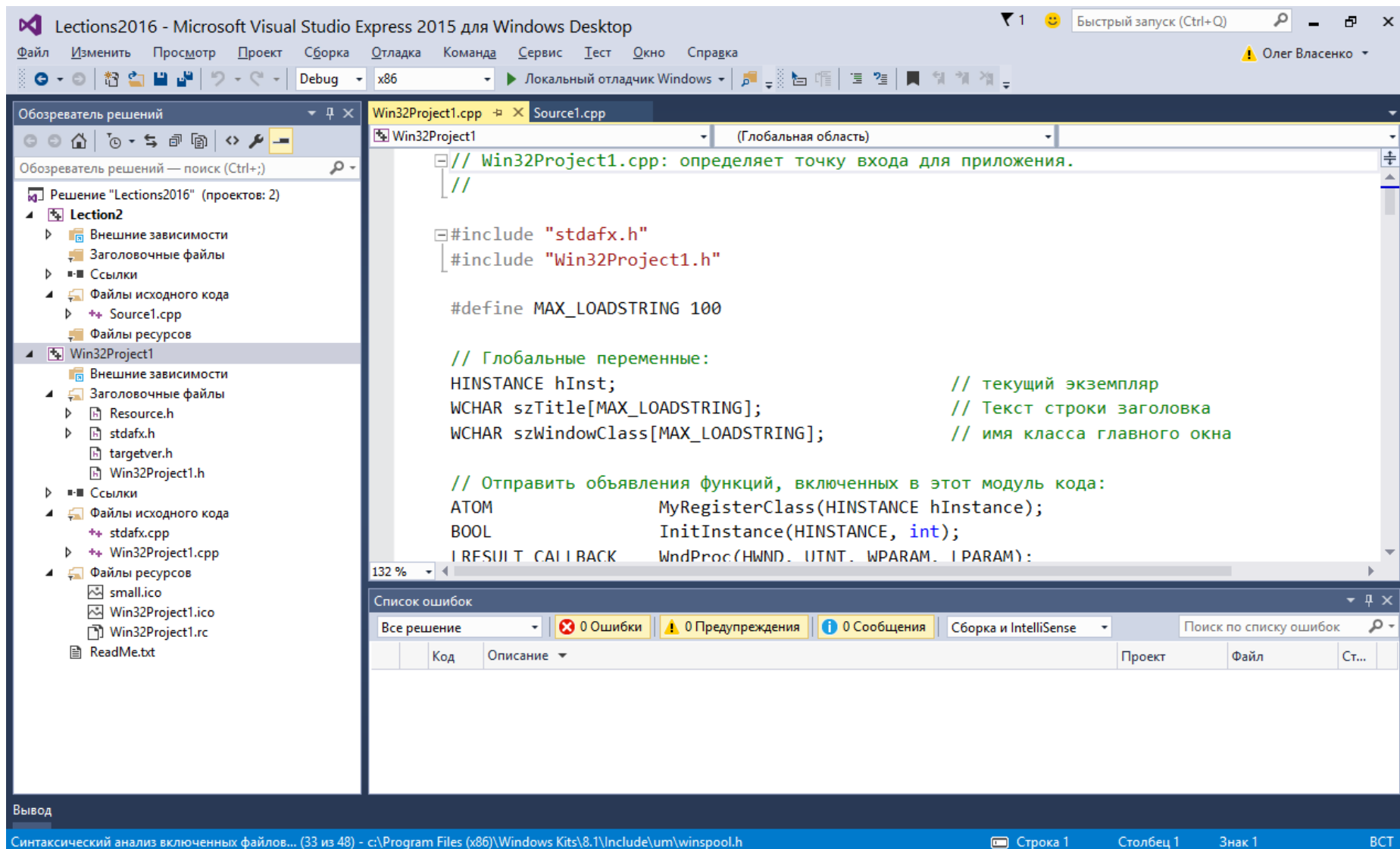
- ☐ Пустой проект
- ☐ Экспорт символов
- ☒ Предварительно скомпилированный заголовок
- ☒ Проверки жизненного цикла разработки безопасного ПО (SDL)

Добавить общие файлы заголовка для:

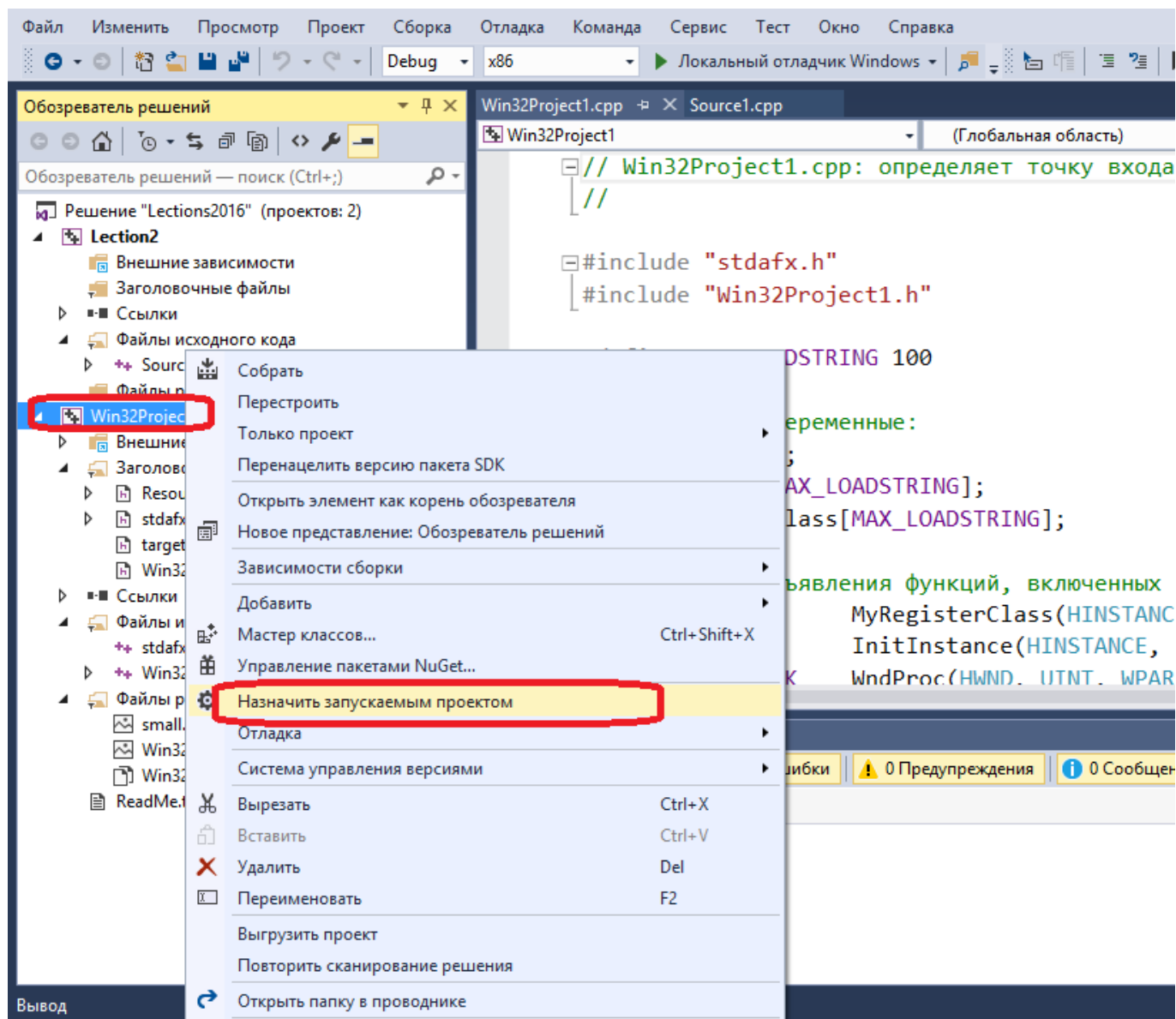
- ☐ Библиотека ATL
- ☐ Библиотека MFC

< Назад    Далее >    **Готово**    Отмена

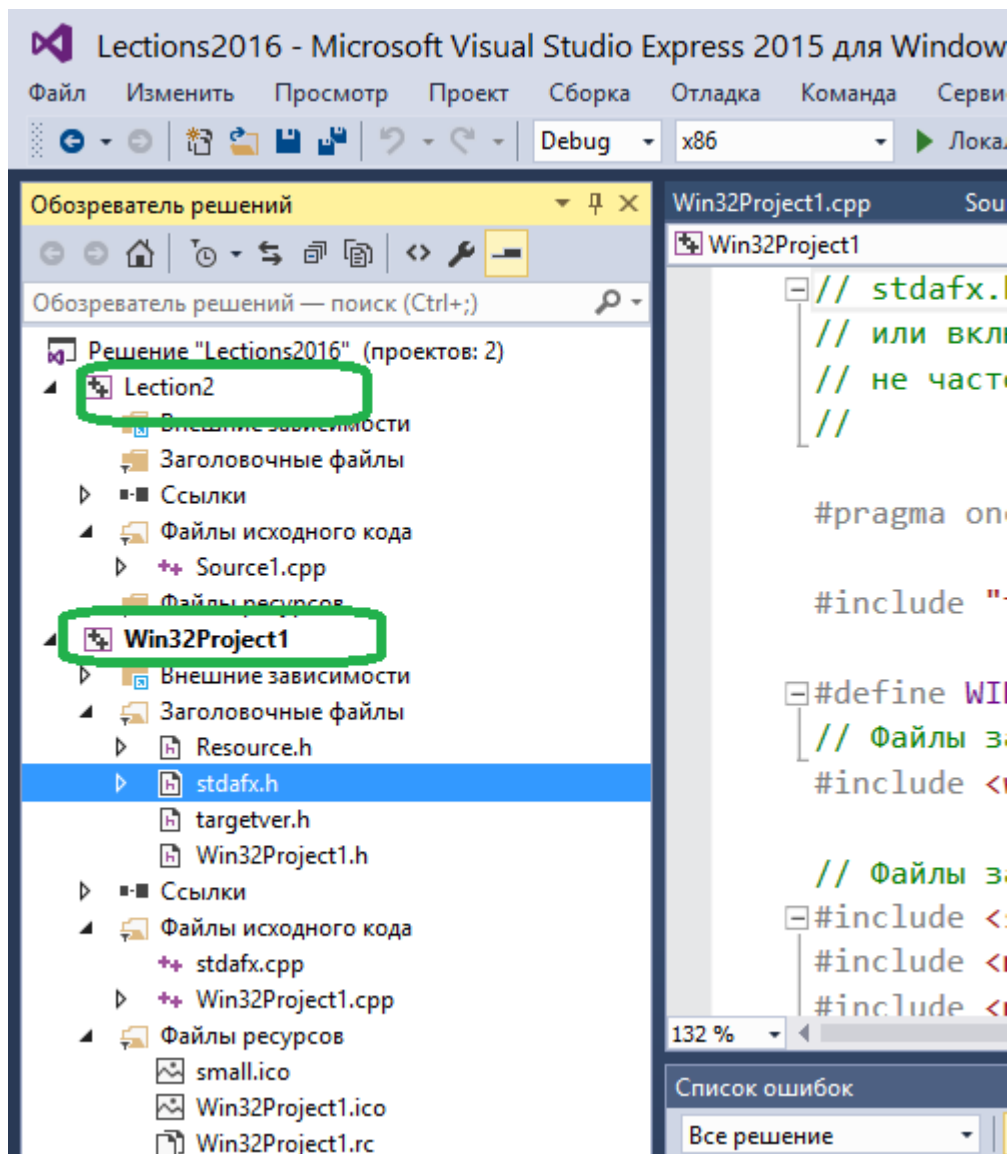
# Создание win32 приложения в VS (5)



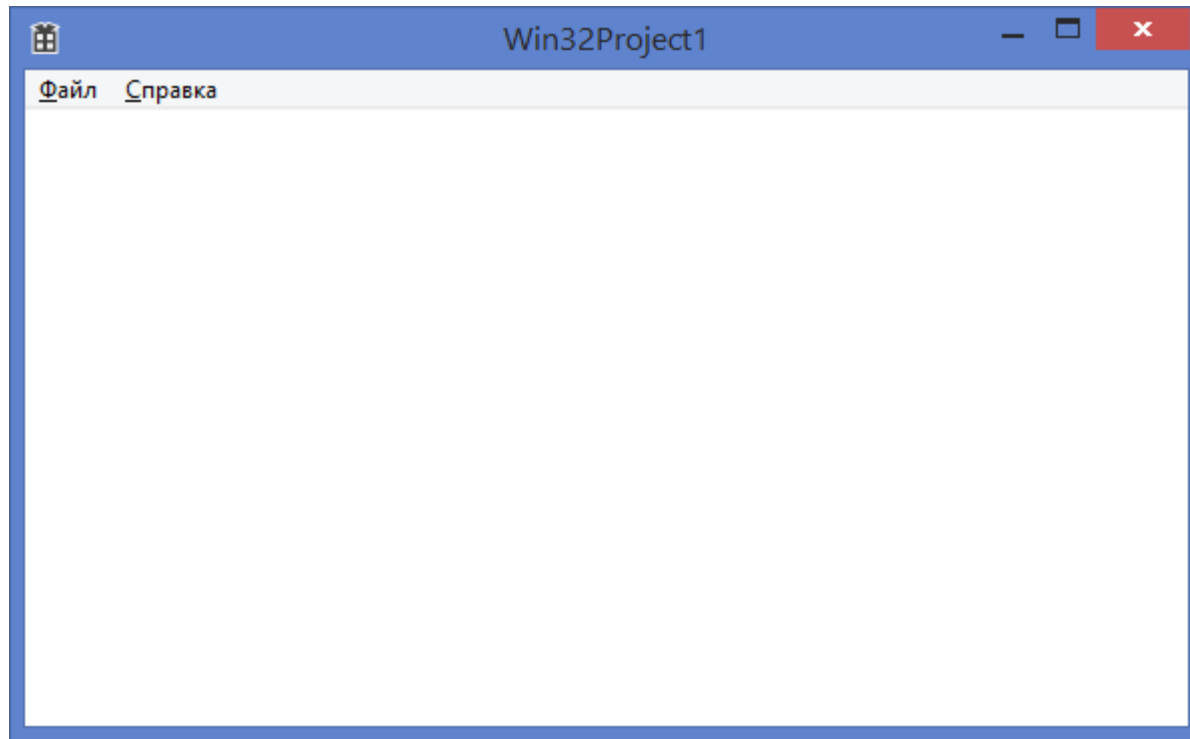
# Создание win32 приложения в VS (6)



# Создание win32 приложения в VS (7) – создано!



# Запущенное win32 приложение!

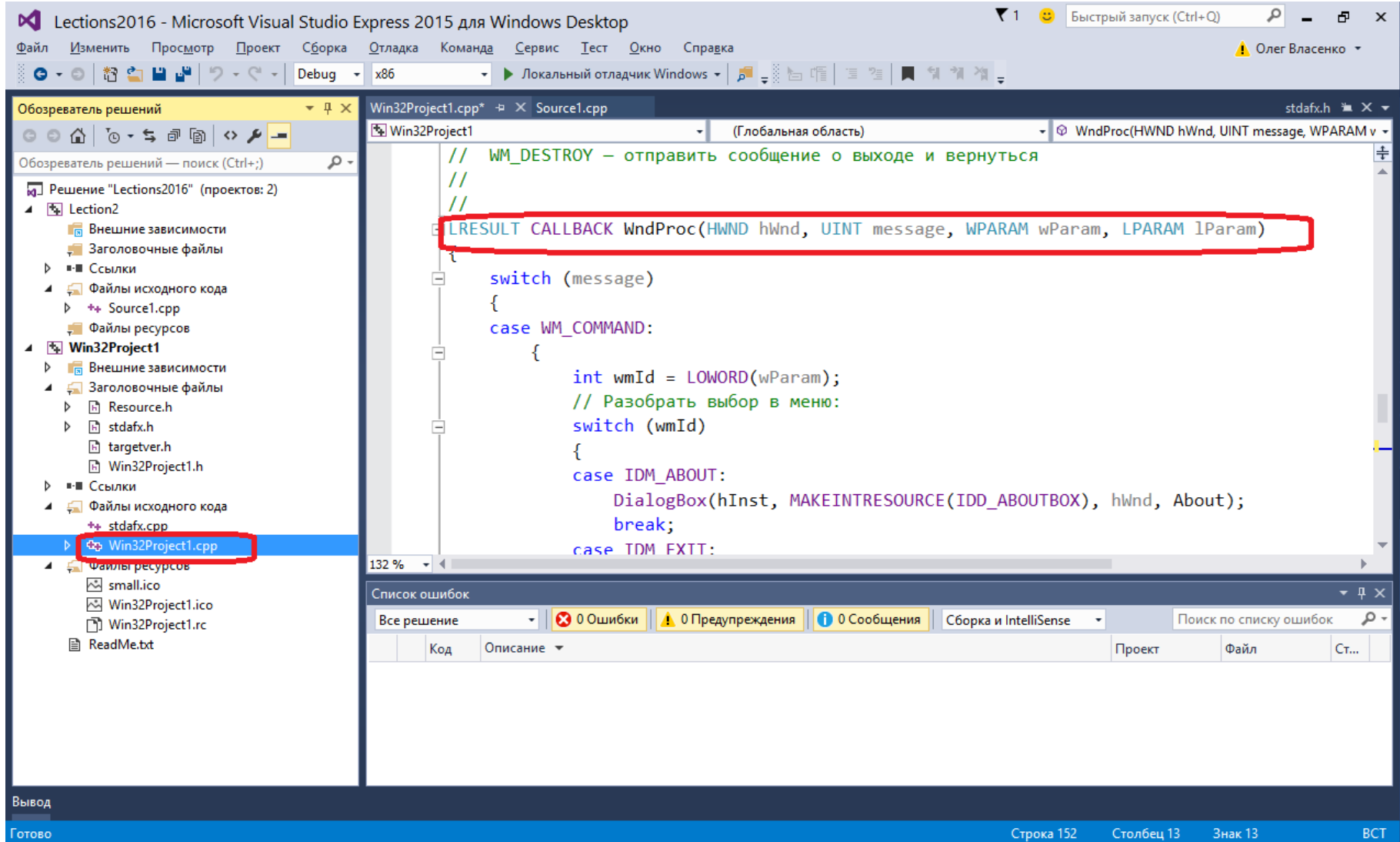




# Где в коде рисовать картинки?

Файл Win32Project1.cpp

Функция WndProc()



# Где в коде рисовать картинки? (2)

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
        ...
        case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hWnd, &ps);
            // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
            EndPaint(hWnd, &ps);
        }
        break;
        ...
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

# Рисуем прямоугольник

```
case WM_PAINT:
```

```
{
```

```
    PAINTSTRUCT ps;
```

```
    HDC hdc = BeginPaint(hWnd, &ps);
```

```
    // RECT - Структура, в которой хранятся параметры прямоугольника
```

```
    RECT rect;
```

```
    //Определяем размер клиентской области окна
```

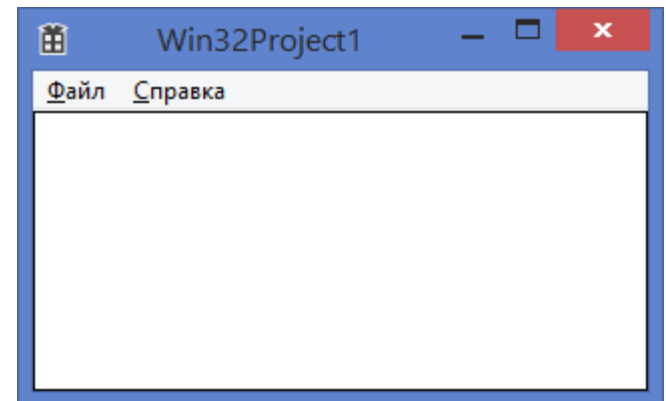
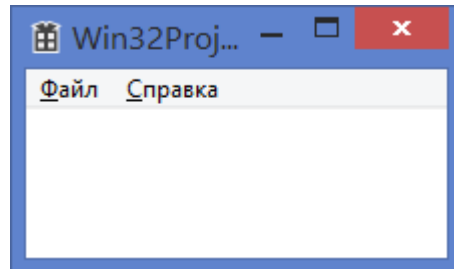
```
    GetClientRect(hWnd, &rect);
```

```
    // Рисуем прямоугольник по границам клиентской области окна
```

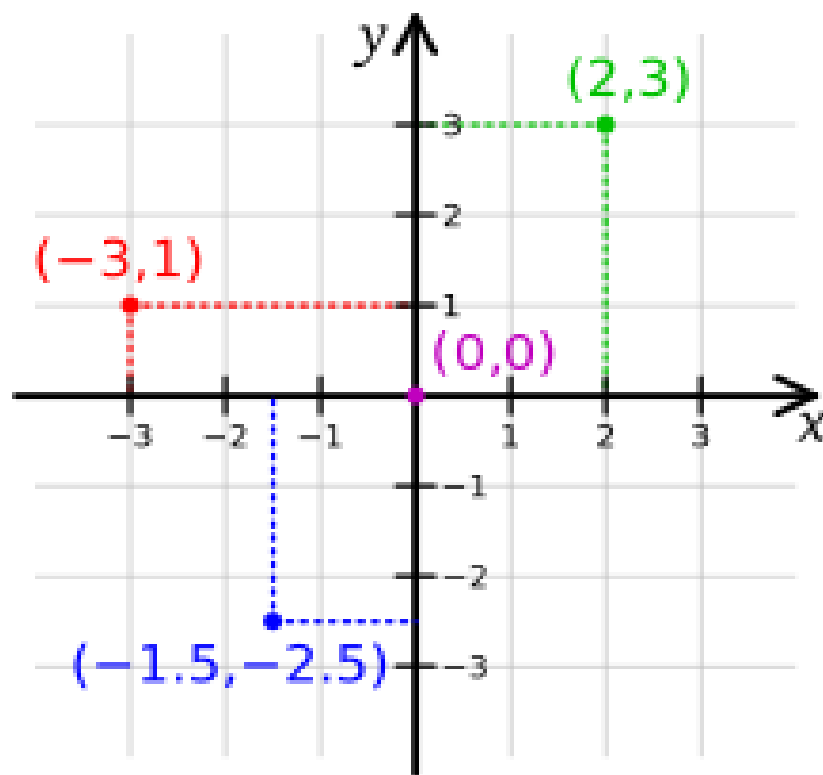
```
    Rectangle(hdc, rect.left, rect.top, rect.right, rect.bottom);
```

```
    EndPaint(hWnd, &ps);
```

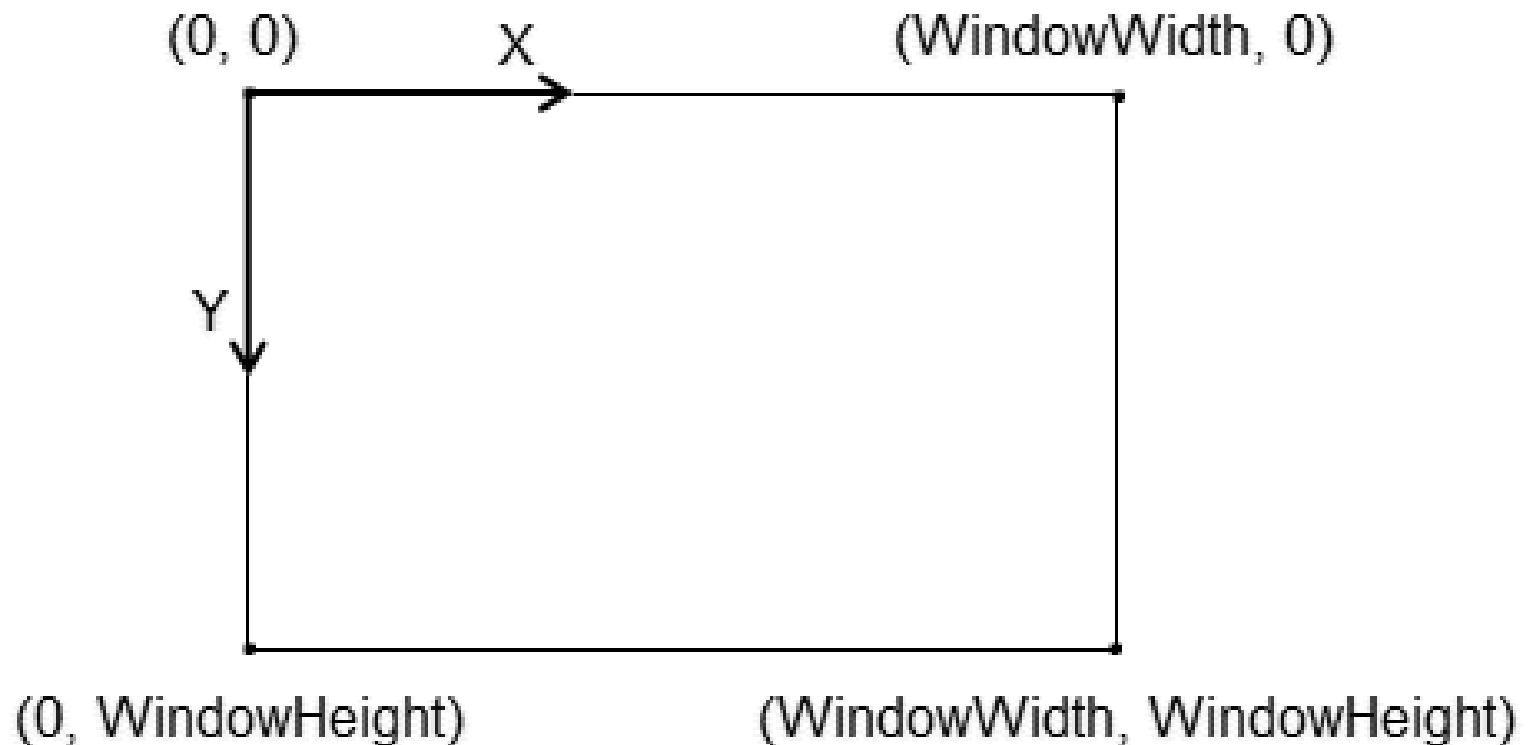
```
}
```



# Точки в Декартовой системе координат



# Экранная система координат



# Рисуем линии

```
case WM_PAINT:
```

```
{
```

```
    PAINTSTRUCT ps;
```

```
    HDC hdc = BeginPaint(hWnd, &ps);
```

```
// Перемещаем "курсор" рисования линии в точку (x = 10, y = 30)
```

```
    MoveToEx(hdc, 10, 30, NULL);
```

```
// Рисуем линию из текущей позиции курсора в точку (x = 10, y = 100)
```

```
// "Курсор" после отрисовки находится в новой точке (x = 10, y = 100)
```

```
    LineTo(hdc, 10, 100);
```

```
// Рисуем линию от предыдущей точки (x = 10, y = 100) до точки (x = 150, y = 100)
```

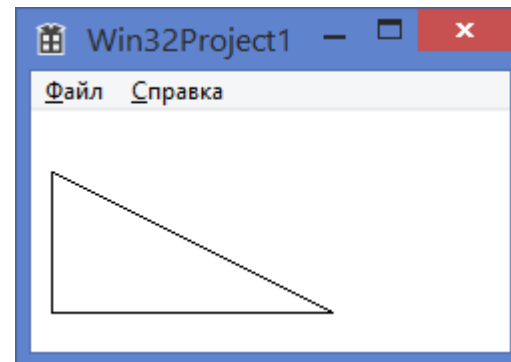
```
    LineTo(hdc, 150, 100);
```

```
// Рисуем линию от предыдущей точки (x = 150, y = 100) до точки (x = 10, y = 30)
```

```
    LineTo(hdc, 10, 30);
```

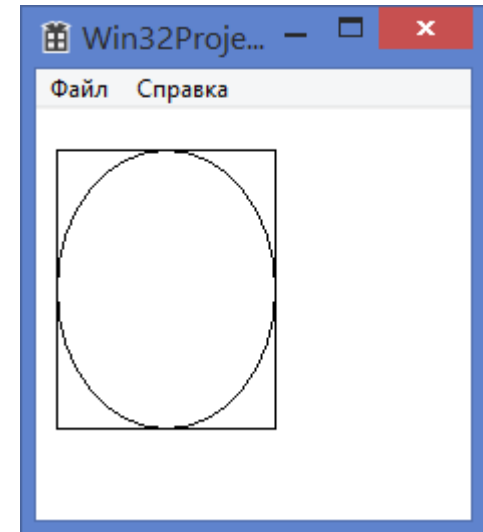
```
    EndPaint(hWnd, &ps);
```

```
}
```

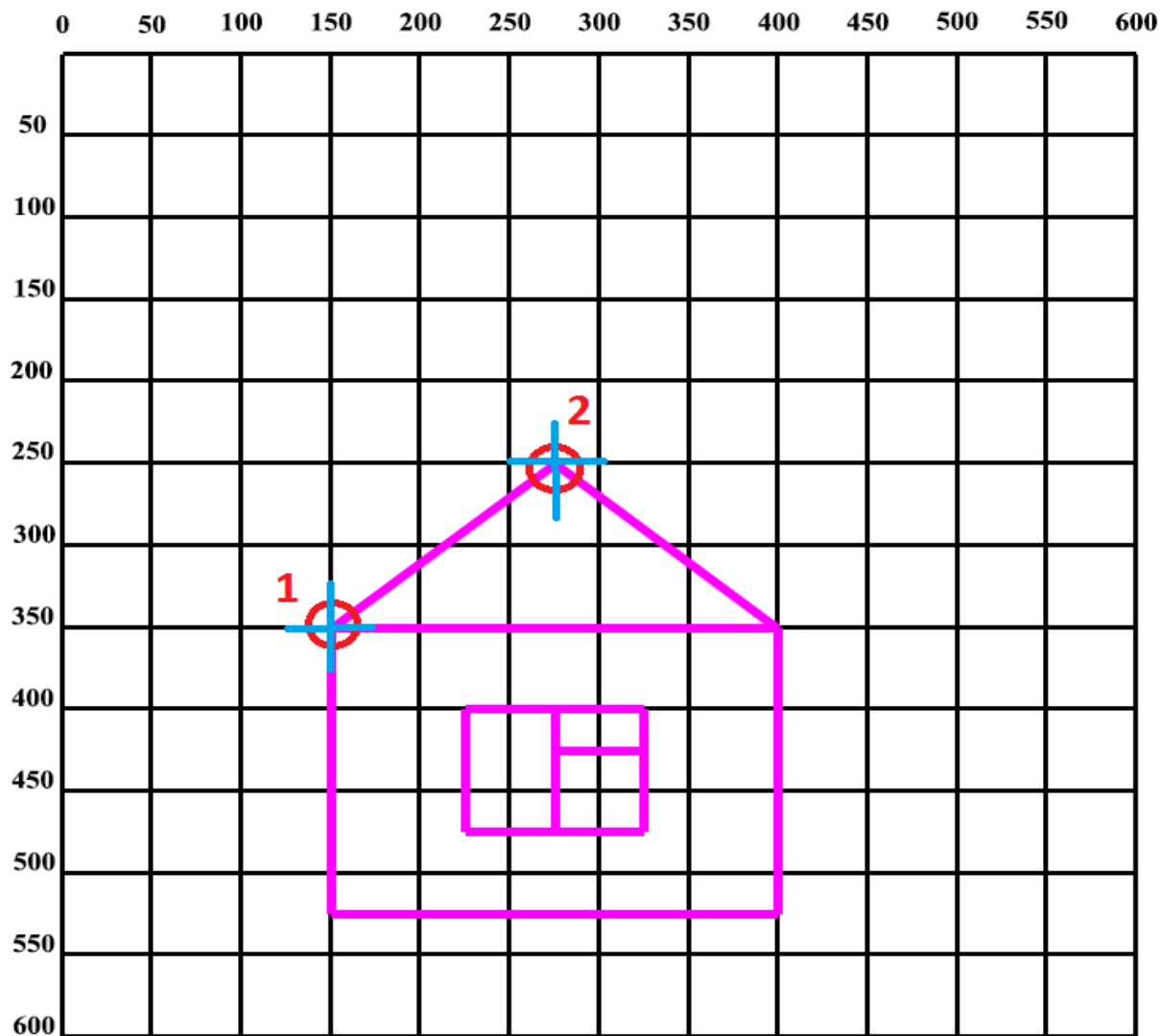


# Рисуем эллипс

```
case WM_PAINT:  
    {  
        PAINTSTRUCT ps;  
        HDC hdc = BeginPaint(hWnd, &ps);  
  
        // Рисуем фиксированный прямоугольник  
        Rectangle(hdc, 10, 20, 120, 160);  
  
        // Рисуем эллипс, вписанный в прямоугольник  
        Ellipse(hdc, 10, 20, 120, 160);  
  
        EndPaint(hWnd, &ps);  
    }
```



# Оцифровка точек в координатной сетке





# Рисуем много линий

```
case WM_PAINT:
```

```
{
```

```
    PAINTSTRUCT ps;
```

```
    HDC hdc = BeginPaint(hWnd, &ps);
```

```
    int x1 = 10, y1 = 100;
```

```
    int x2 = 300, y2 = 100;
```

```
    int i = 0;
```

```
    do {
```

```
        MoveToEx(hdc, x1, y1, NULL);
```

```
        LineTo(hdc, x2, y2);
```

```
        y1 = y1 - 5;
```

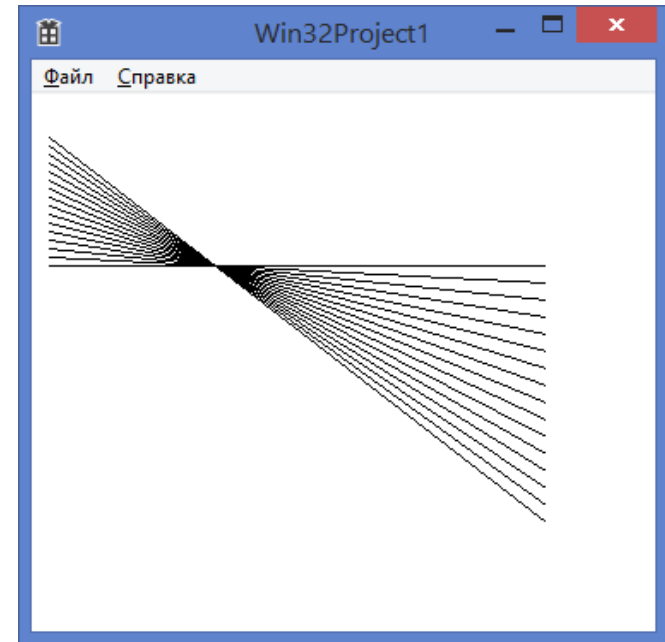
```
        y2 = y2 + 10;
```

```
        i++;
```

```
    } while (i < 16);
```

```
    EndPaint(hWnd, &ps);
```

```
}
```



# Рисуем много линий

```
case WM_PAINT:
```

```
{
```

```
    PAINTSTRUCT ps;
```

```
    HDC hdc = BeginPaint(hWnd, &ps);
```

```
    int x1 = 10, y1 = 100;
```

```
    int x2 = 300, y2 = 100;
```

```
    int i = 0;
```

```
    do {
```

```
        MoveToEx(hdc, x1, y1, NULL);
```

```
        LineTo(hdc, x2, y2);
```

```
        y1 = y1 - 5;
```

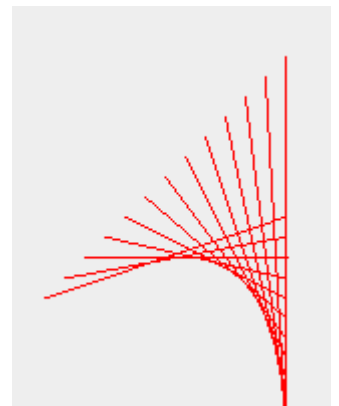
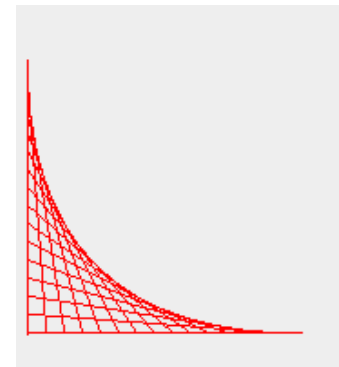
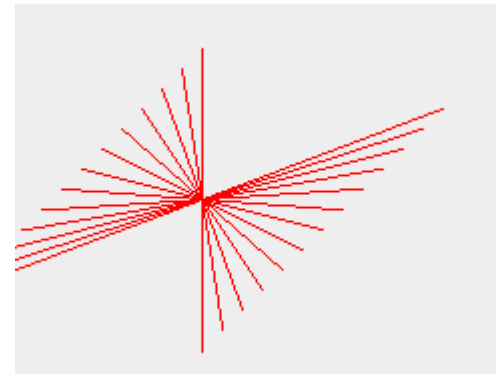
```
        y2 = y2 + 10;
```

```
        i++;
```

```
    } while (i < 16);
```

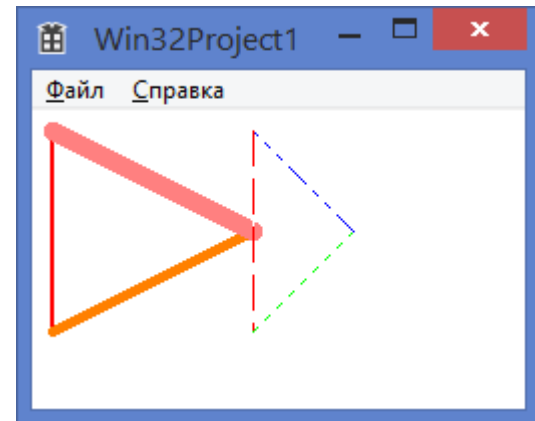
```
    EndPaint(hWnd, &ps);
```

```
}
```



# Такое разное перо

```
case WM_PAINT: {  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);  
    HPEN hPen;  
    hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));  
    SelectObject(hdc, hPen);  
    MoveToEx(hdc, 10, 10, NULL);  
    LineTo(hdc, 10, 110);  
  
    hPen = CreatePen(PS_SOLID, 5, RGB(255, 128, 0));  
    SelectObject(hdc, hPen);  
    LineTo(hdc, 110, 60);  
  
    hPen = CreatePen(PS_SOLID, 10, RGB(255, 128, 128));  
    SelectObject(hdc, hPen);  
    LineTo(hdc, 10, 10);
```



## Такое разное перо (2)

```
hPen = CreatePen(PS_DASH, 1, RGB(255, 0, 0));
```

```
SelectObject(hdc, hPen);
```

```
MoveToEx(hdc, 110, 10, NULL);
```

```
LineTo(hdc, 110, 110);
```

```
hPen = CreatePen(PS_DOT, 1, RGB(0, 255, 0));
```

```
SelectObject(hdc, hPen);
```

```
LineTo(hdc, 160, 60);
```

```
hPen = CreatePen(PS_DASHDOTDOT, 1, RGB(0, 0, 255));
```

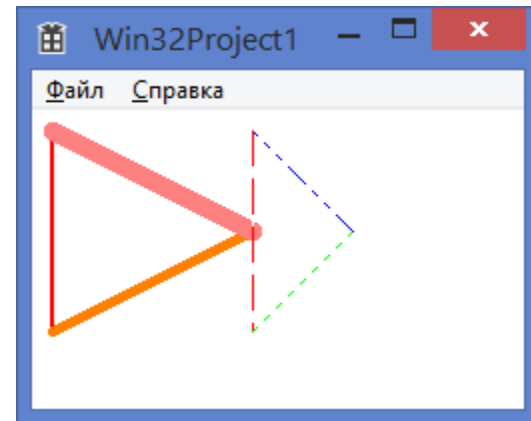
```
SelectObject(hdc, hPen);
```

```
LineTo(hdc, 110, 10);
```

```
DeleteObject(hPen);
```

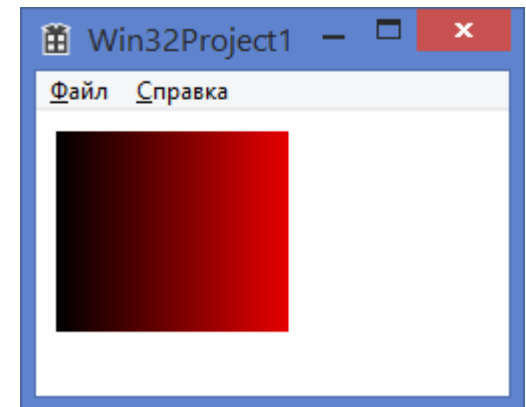
```
EndPaint(hWnd, &ps);
```

```
}
```



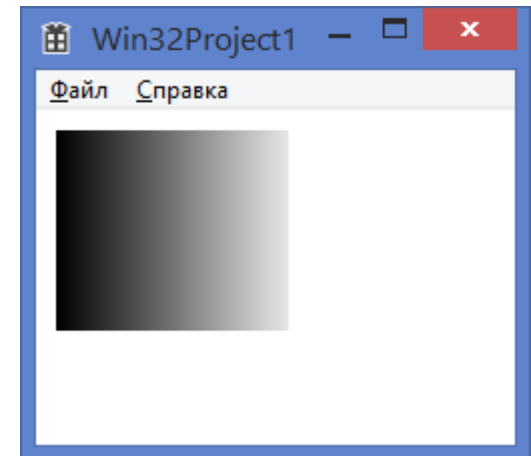
# Цветное перо

```
case WM_PAINT: {  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);  
    HPEN hPen;  
    int x = 10;  
    int r = 0;  
    do {  
        hPen = CreatePen(PS_SOLID, 1, RGB(r, 0, 0));  
        SelectObject(hdc, hPen);  
        MoveToEx(hdc, x, 10, NULL);  
        LineTo(hdc, x, 110);  
        DeleteObject(hPen);  
        x += 1;  
        r += 2;  
    } while (x <= 125);  
    EndPaint(hWnd, &ps);  
}
```



# Цветное перо

```
case WM_PAINT: {  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);  
    HPEN hPen;  
    int x = 10;  
    int r = 0;  
    do {  
        hPen = CreatePen(PS_SOLID, 1, RGB(r, r, r));  
        SelectObject(hdc, hPen);  
        MoveToEx(hdc, x, 10, NULL);  
        LineTo(hdc, x, 110);  
        DeleteObject(hPen);  
        x += 1;  
        r += 2;  
    } while (x <= 125);  
    EndPaint(hWnd, &ps);  
}
```



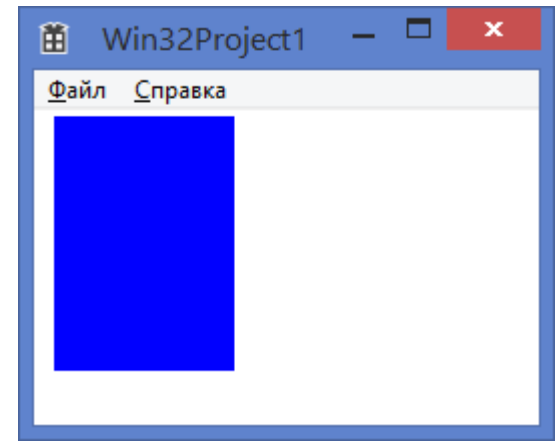
# Кисть

```
case WM_PAINT: {  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);
```

```
    HBRUSH hBrush;  
    hBrush = CreateSolidBrush(RGB(0, 0, 255));  
    SelectObject(hdc, hBrush);  
    RECT rect = { 10, 3, 100, 130 };  
    FillRect(hdc, &rect, hBrush);
```

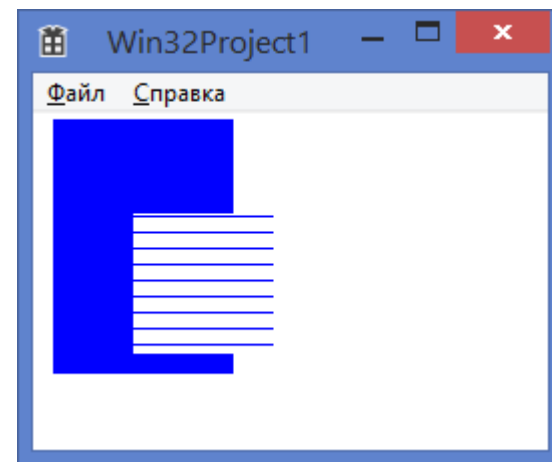
// СЛЕДУЮЩИЙ КОД ВСТАВИТЬ СЮДА!!!

```
    EndPaint(hWnd, &ps);  
}
```



## Кисть (2)

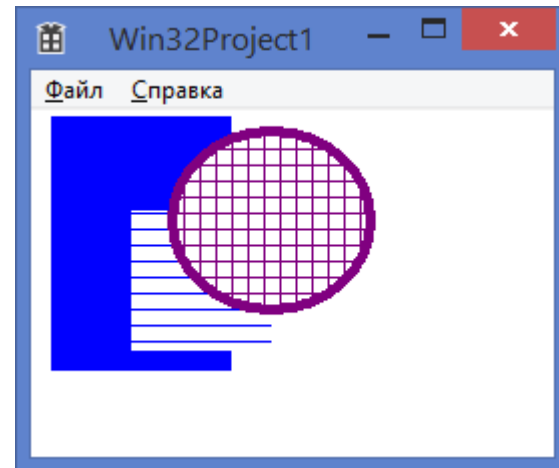
```
hBrush = CreateHatchBrush(HS_HORIZONTAL, RGB(0, 0, 255));  
SelectObject(hdc, hBrush);  
RECT rect2 = { 50, 50, 120, 120 };  
FillRect(hdc, &rect2, hBrush);
```





## Кисть (3)

```
hBrush = CreateHatchBrush(HS_CROSS, RGB(128, 0, 128));  
SelectObject(hdc, hBrush);  
HPEN hPen;  
hPen = CreatePen(PS_SOLID, 5, RGB(128, 0, 128));  
SelectObject(hdc, hPen);  
Ellipse( hdc, 70, 10, 170, 100);  
  
DeleteObject(hBrush);
```



# Рисуем много линий из центра

```
case WM_PAINT:
```

```
{
```

```
    PAINTSTRUCT ps;
```

```
    HDC hdc = BeginPaint(hWnd, &ps);
```

```
    RECT rect;
```

```
    GetClientRect(hWnd, &rect);
```

```
    int cx = rect.right / 2;
```

```
    int cy = rect.bottom / 2;
```

```
    int x = 0;
```

```
    do {
```

```
        MoveToEx(hdc, cx, cy, NULL);
```

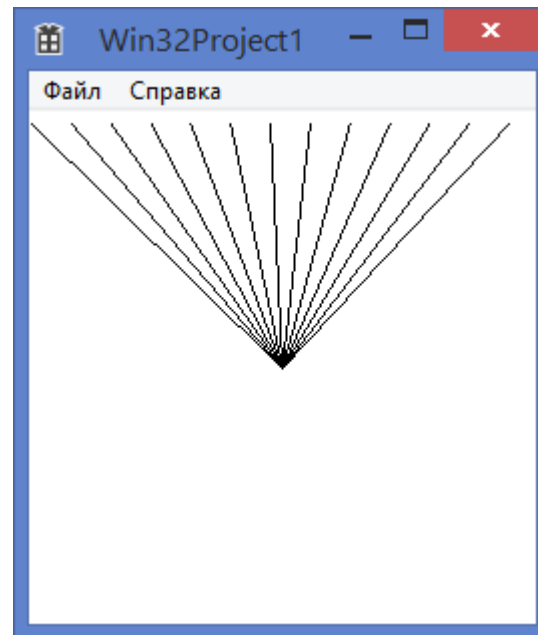
```
        LineTo(hdc, x, 5);
```

```
        x += 20;
```

```
    } while (x < rect.right);
```

```
    EndPaint(hWnd, &ps);
```

```
}
```



# struct

В языке Си, структура (struct) — компози́тный тип данных, инкапсулирующий без сокрытия набор значений различных типов.

([https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0\\_%28%D1%8F%D0%B7%D1%8B%D0%BA\\_%D0%A1%D0%B8%29](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%A1%D0%B8%29) )

**Структура** — это агрегатный тип данных. Она может содержать в себе разнотипные элементы. (<http://cppstudio.com/post/5377/> )

// определение структуры

```
struct str_name
```

```
{
```

```
    int        member_1;
```

```
    float      member_2;
```

```
    char       member_3[256];
```

```
};
```

// объявление переменной-структуры

```
struct str_name struct0;
```

# Пример struct

```
struct tagRECT // определение структуры tagRECT
{
    LONG    left; // поле left
    LONG    top;
    LONG    right;
    LONG    bottom;
};
```

```
int main()
{
    struct tagRECT rect; // объявили переменную-структуру
    rect.left = 10; // поле left получило значение «10»
    rect.top = 20;
    rect.right = 200;
    rect.bottom = 300;
}
```

# struct и typedef

**typedef** struct **name**

{

type atrib1;

type atrib2;

// остальные элементы структуры...

} **newStructName** structVar1;

**struct name** structVar3;

**newStructName** structVar2;

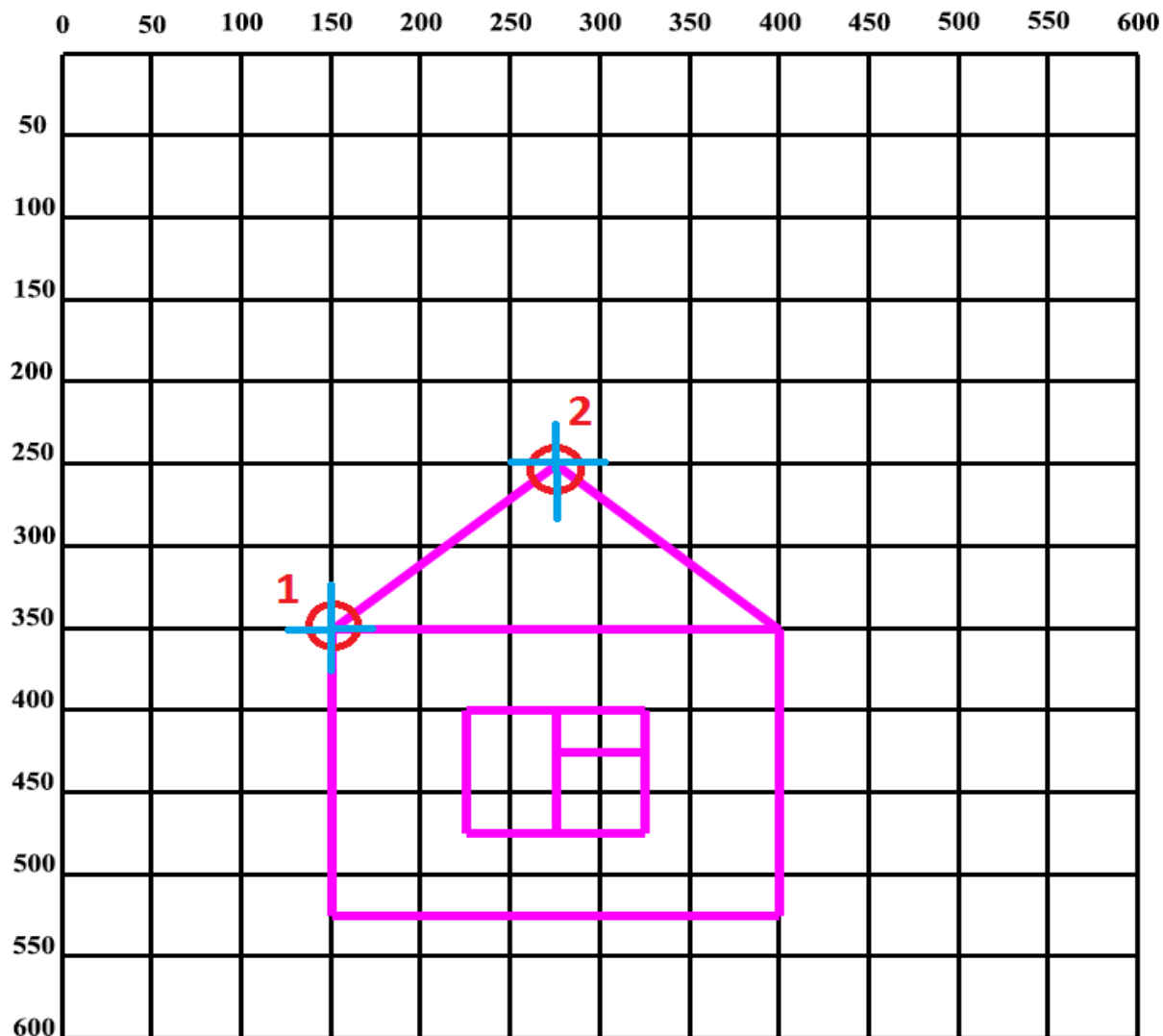
# struct и typedef – пример (RECT)

```
// windef.h
typedef struct tagRECT
{
    LONG    left;
    LONG    top;
    LONG    right;
    LONG    bottom;
} RECT, *PRECT, NEAR *NPRECT, FAR *LPRECT;
...
// Win32Project1.cpp:
// RECT - Структура, в которой хранятся параметры прямоугольника
RECT rect; // struct tagRECT rect;
//Определяем размер клиентской области окна
GetClientRect(hWnd, &rect);
// Рисуем прямоугольник по границам клиентской области окна
Rectangle(hdc, rect.left, rect.top, rect.right, rect.bottom);
```

# Домашнее задание

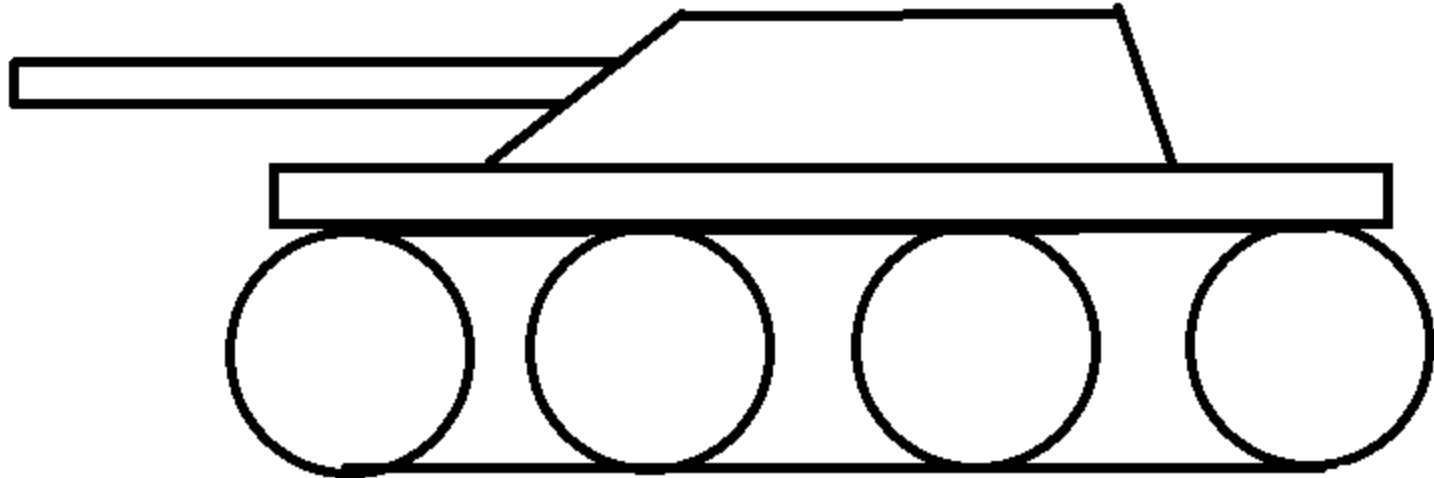
1. Прочитать про рисование в WinAPI  
<http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html>
2. Установить Visual Studio (если ранее не установили).
3. Создать простейшее Win32 приложение
4. Нарисовать домик и танк в этом приложении (вместо танка можно нарисовать что-то более жизнеутверждающее - с эллипсами или дугами)

# Домашнее задание 1 – нарисовать домик





## Домашнее Задание 2 – оцифровать и нарисовать танк



# Источники информации

- **КАК рисовать в Win32 API? -**

<http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html>