Курс «Основы Алгоритмизации и программирование»

Власенко Олег Федосович
SimbirSoft

Лекция 3.

Графика

ЛР 4. Рисование при помощи линий

ЛР 5. Рисование при помощи плоских фигур

Термины

Операционная система (ОС)

https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

Операцио́нная систе́ма, сокр. **ОС** (англ. *operating system, OS*) — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.
В логической структуре типичной вычислительной системы операционная система занимает положение между устройствами с их микроархитектурой, машинным языком и, возможно, собственными (встроенными) микропрограммами (драйверами) — с одной стороны — и прикладными программами с другой.
Разработчикам программного обеспечения операционная система позволяет абстрагироваться от деталей реализации и функционирования устройств, предоставляя минимально необходимый набор функций (см.: интерфейс программирования приложений).
В большинстве вычислительных систем операционная система является основной, наиболее важной (а иногда и единственной) частью системного программного обеспечения. С 1990-х годов наиболее распространёнными операционными системами являются системы семейства Windows, Unix и UNIX-подобные системы.

Windows

https://ru.wikipedia.org/wiki/Windows

Windows (с англ. — *«Окна»*) — группа семейств коммерческих проприетарных операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса.

Рабочий стол Windows 11 — панель задач и главное меню по умолчанию отцентрированы

Разработчик Microsoft

Интерфейс

https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81

Интерфе́йс (от англ. *interface*) — граница между двумя функциональными объектами, требования к которой определяются стандартом[1]; совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы[2].

Интерфейсы в информатике и вычислительной технике

[править | править код]

В наиболее общем смысле интерфейсом называется общая граница, через которую передаётся информация (стандарт ISO 24765)[3].

В вычислительной системе взаимодействие может осуществляться на пользовательском, программном и аппаратном уровнях.

Аппаратные интерфейсы портативного компьютера: сетевой разъем Ethernet (в центре), слева часть порта VGA, справа вверху разъем DisplayPort, справа внизу USB 2.0

GUI

https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F

Графи́ческий интерфе́йс по́льзователя (ГИП), **графи́ческий по́льзовательский интерфе́йс (ГПИ)** (англ. *graphical user interface,* ***GUI***) — система средств для взаимодействия пользователя с электронными устройствами, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.).

API

https://ru.wikipedia.org/wiki/API

API (**программный интерфейс приложения**) (англ. ***a****pplication* ***p****rogramming* ***i****nterface*, *API* [эй-пи-ай][1]) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола (например, RFC[2]), программного каркаса (фреймворка)[3] или стандарта вызовов функций операционной системы[4]. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

Windows API

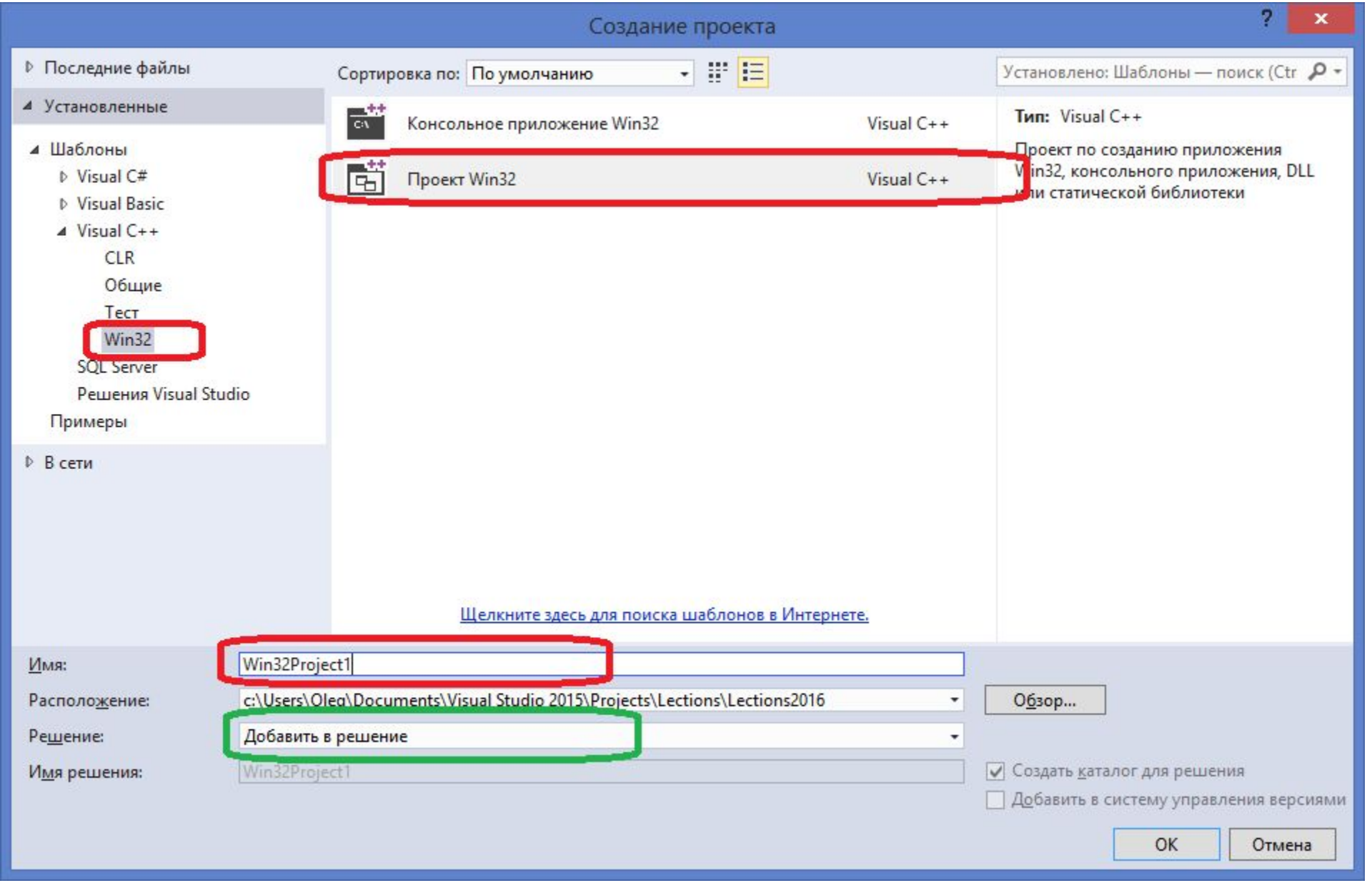
https://ru.wikipedia.org/wiki/Windows_API

- **Windows API** (англ. *application programming interfaces*) — общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств *Microsoft Windows* корпорации «Майкрософт». Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой *Windows*. Для создания программ, использующих *Windows API*, корпорация «Майкрософт» выпускает комплект разработчика программного обеспечения, который называется *Platform SDK* и содержит документацию, набор библиотек, утилит и других инструментальных средств для разработки.
- *Windows API* спроектирован для использования в языке Си для написания прикладных программ, предназначенных для работы под управлением операционной системы MS Windows. Работа через *Windows API* — это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ. Более низкий уровень доступа, необходимый только для драйверов устройств, в текущих версиях *Windows* предоставляется через *Windows Driver Model*.
- Windows API представляет собой множество функций, структур данных и числовых констант, следующих соглашениям языка Си.

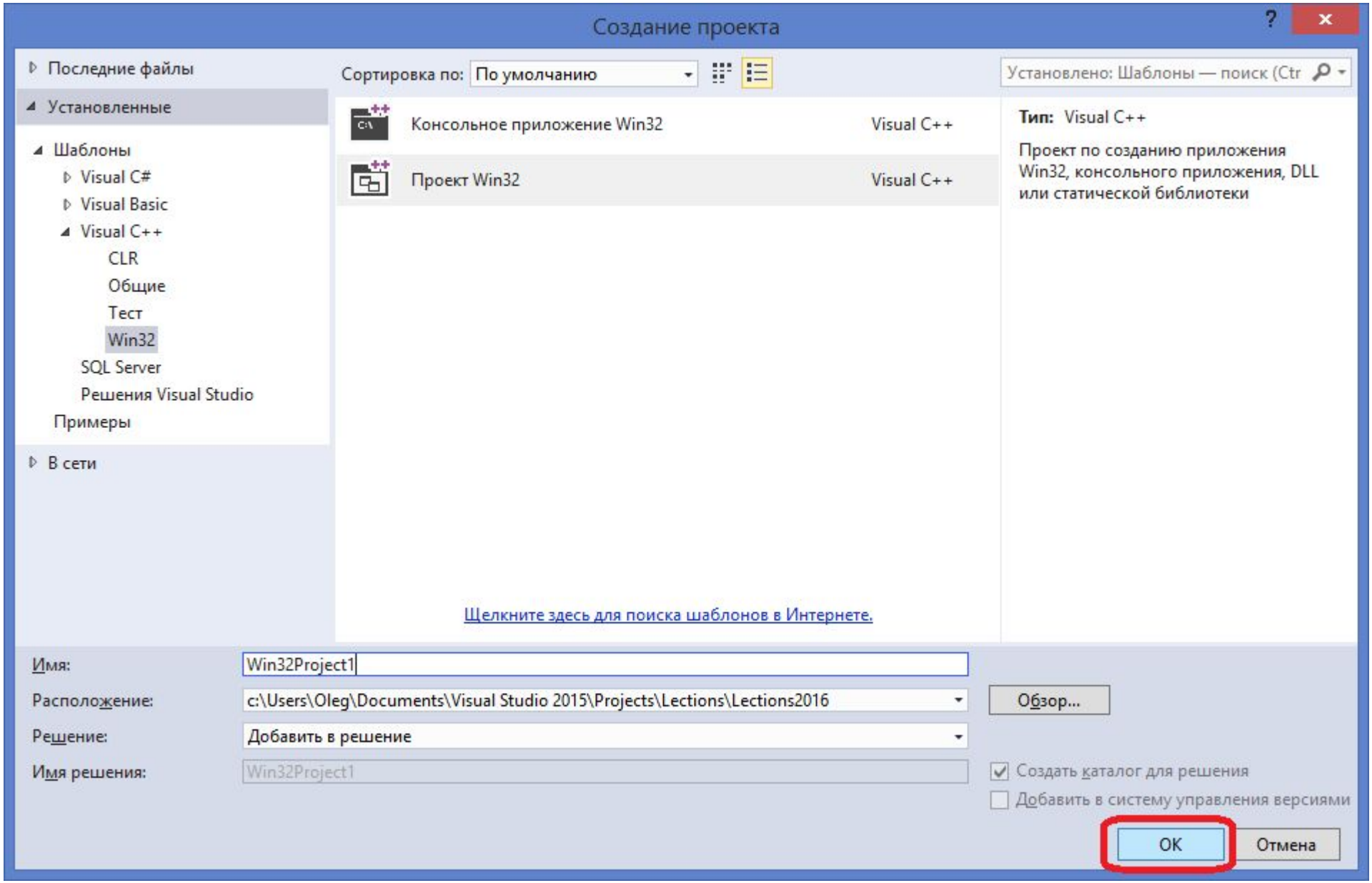
Графика

Создание win32 приложения в VS

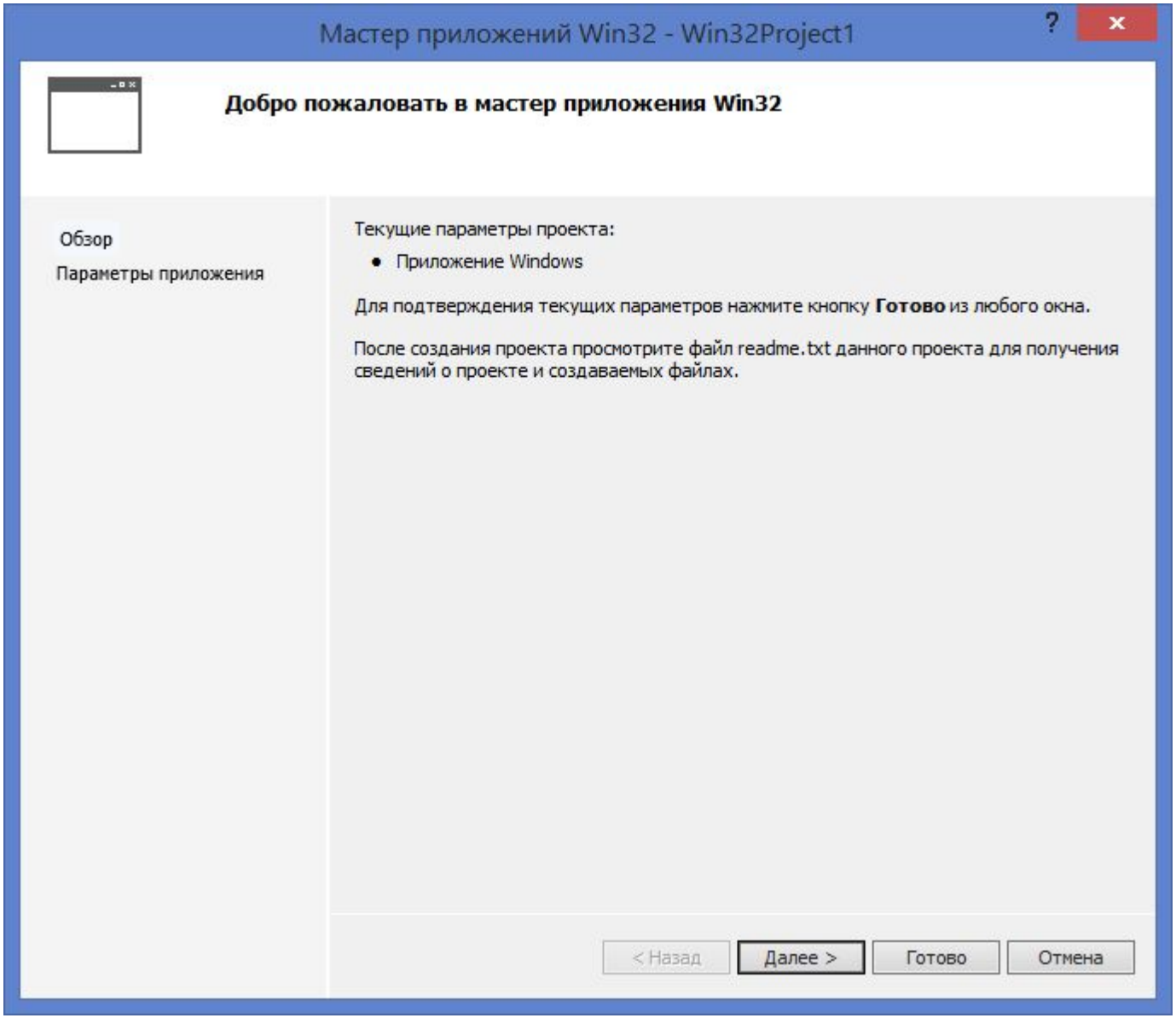
Создание win32 приложения в VS



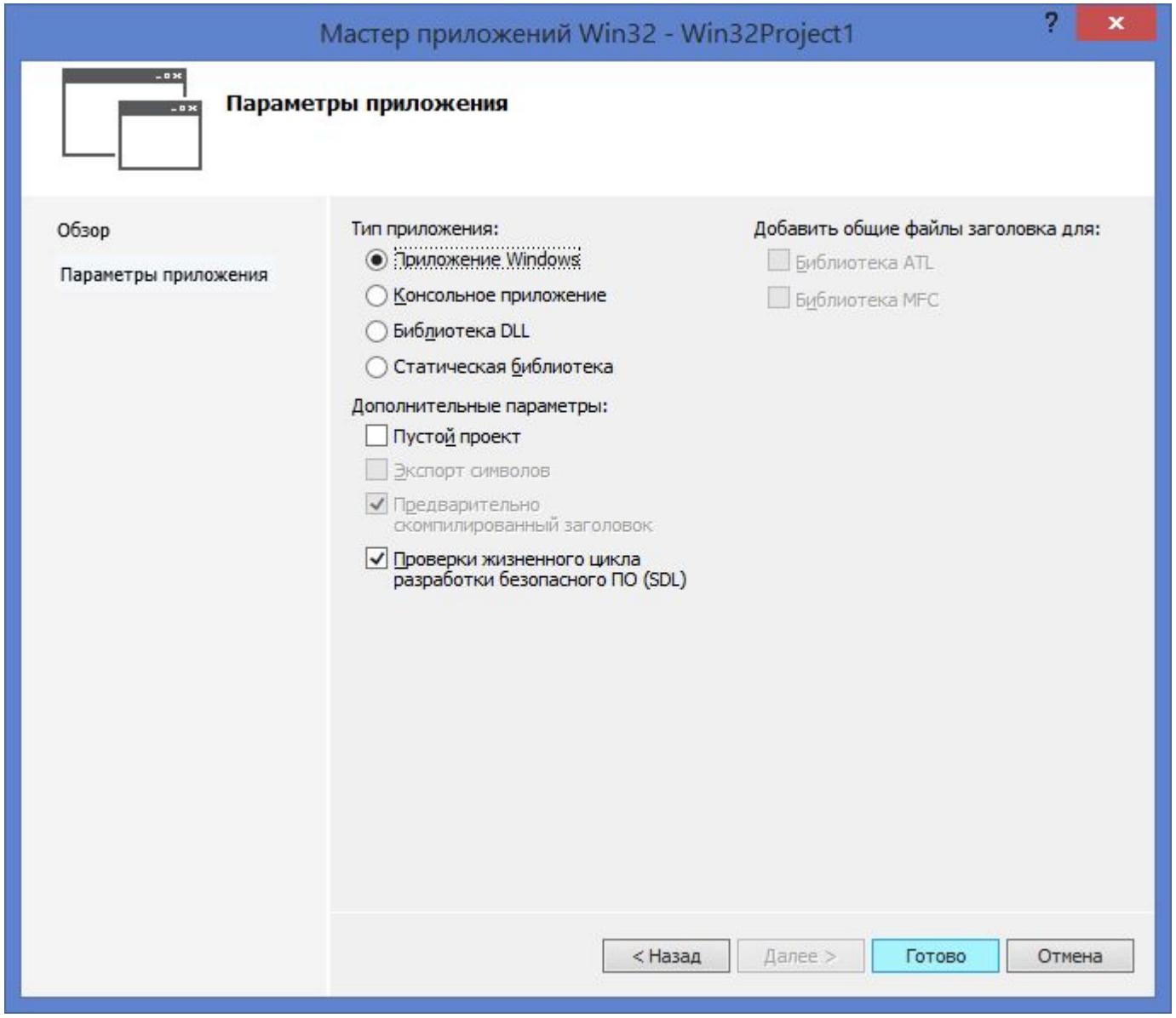
Создание win32 приложения в VS (2)



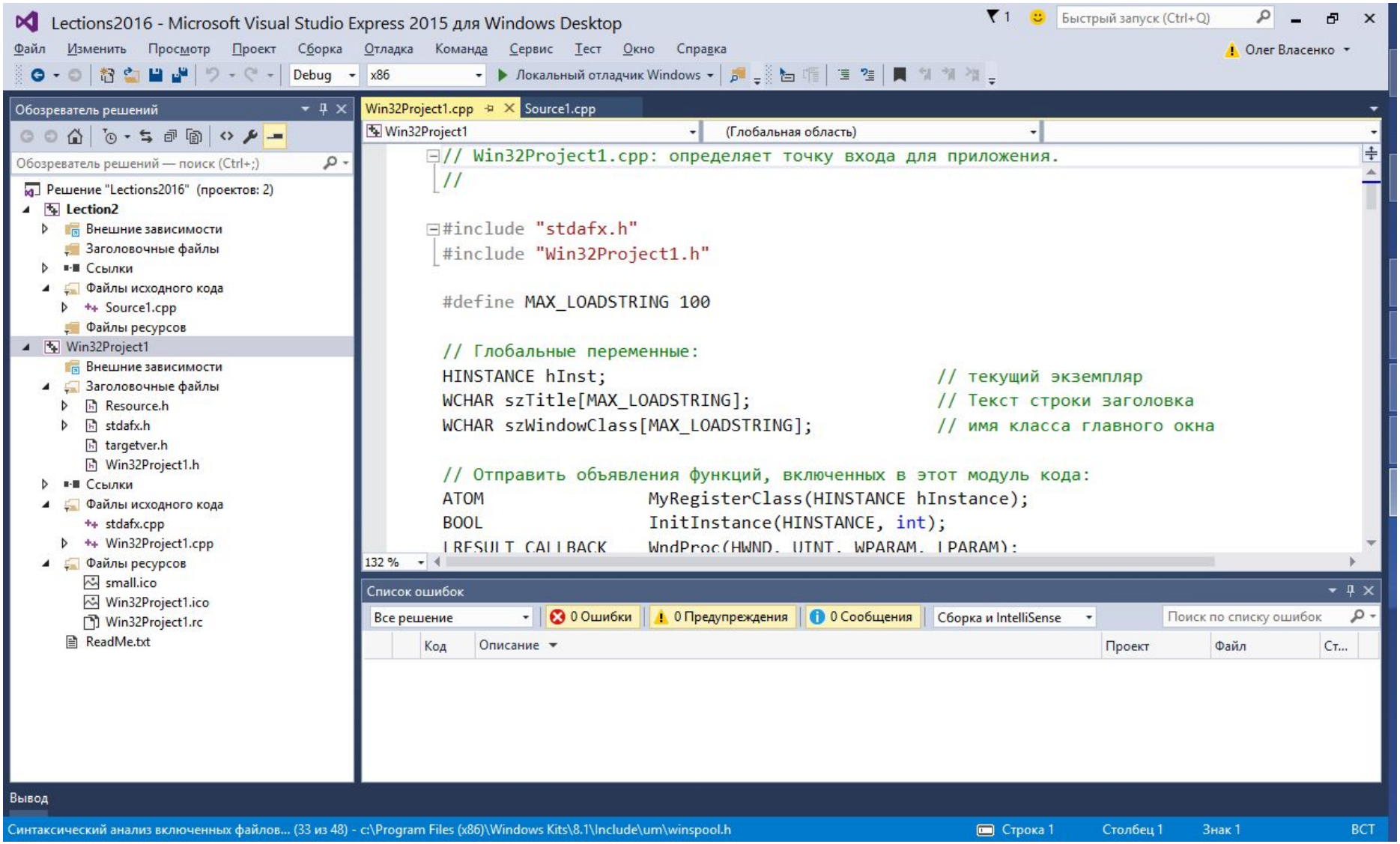
Создание win32 приложения в VS (3)



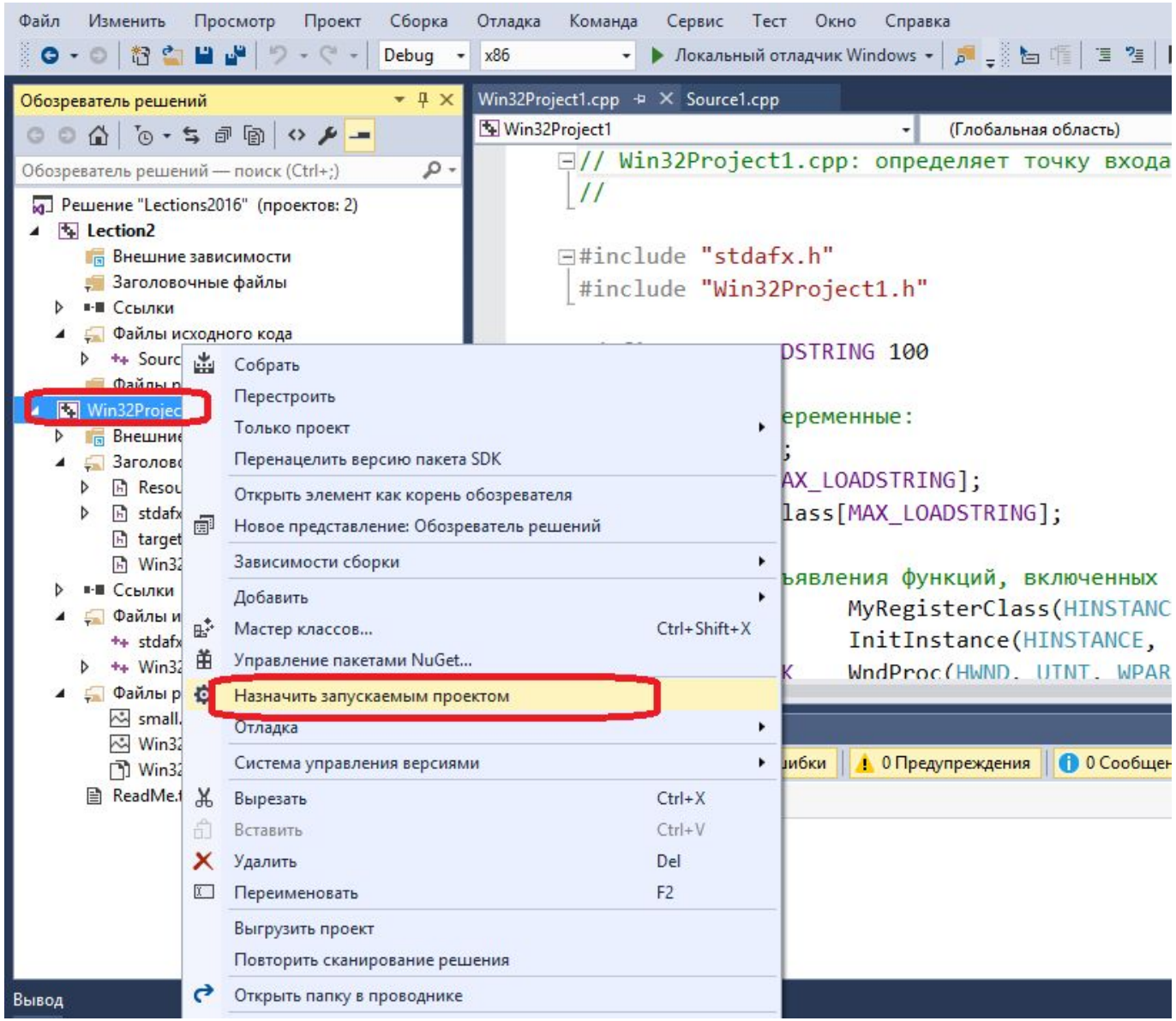
Создание win32 приложения в VS (4)



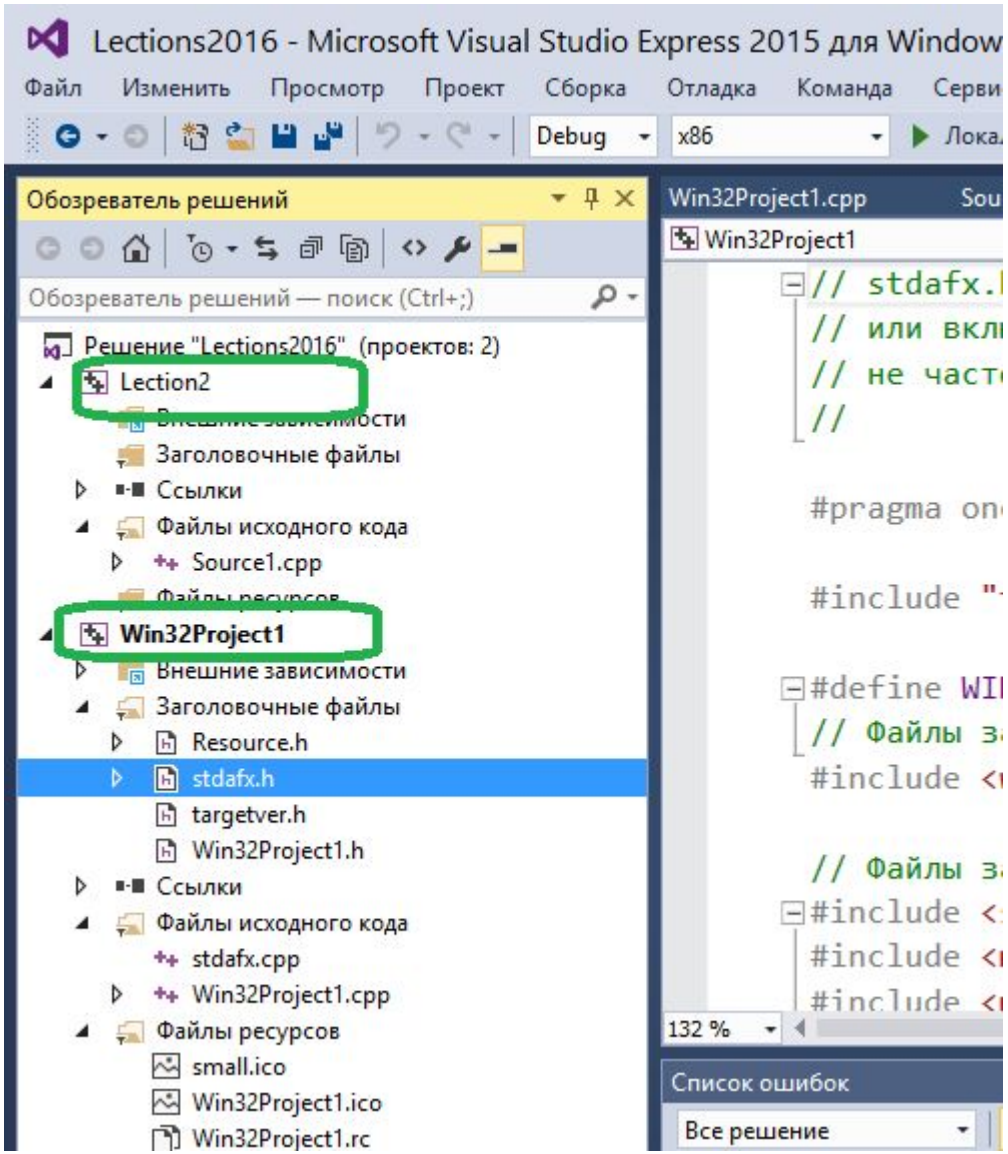
Создание win32 приложения в VS (5)



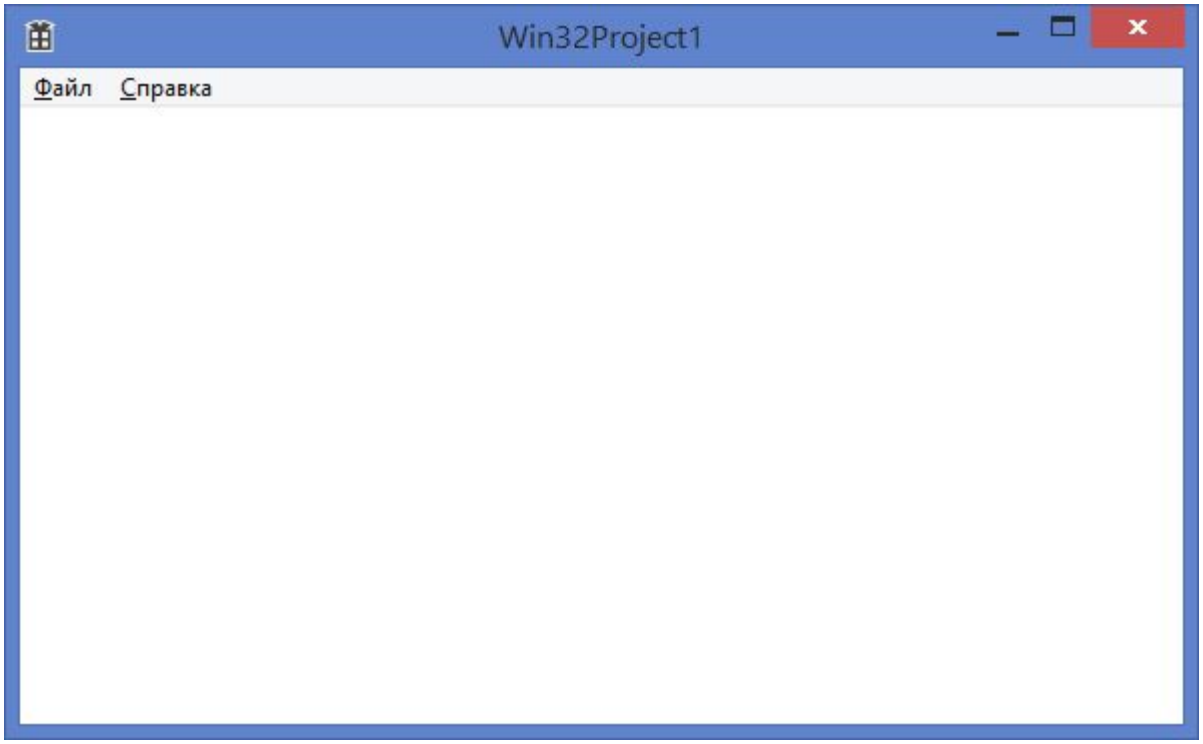
Создание win32 приложения в VS (6)



Создание win32 приложения в VS (7) – создано!



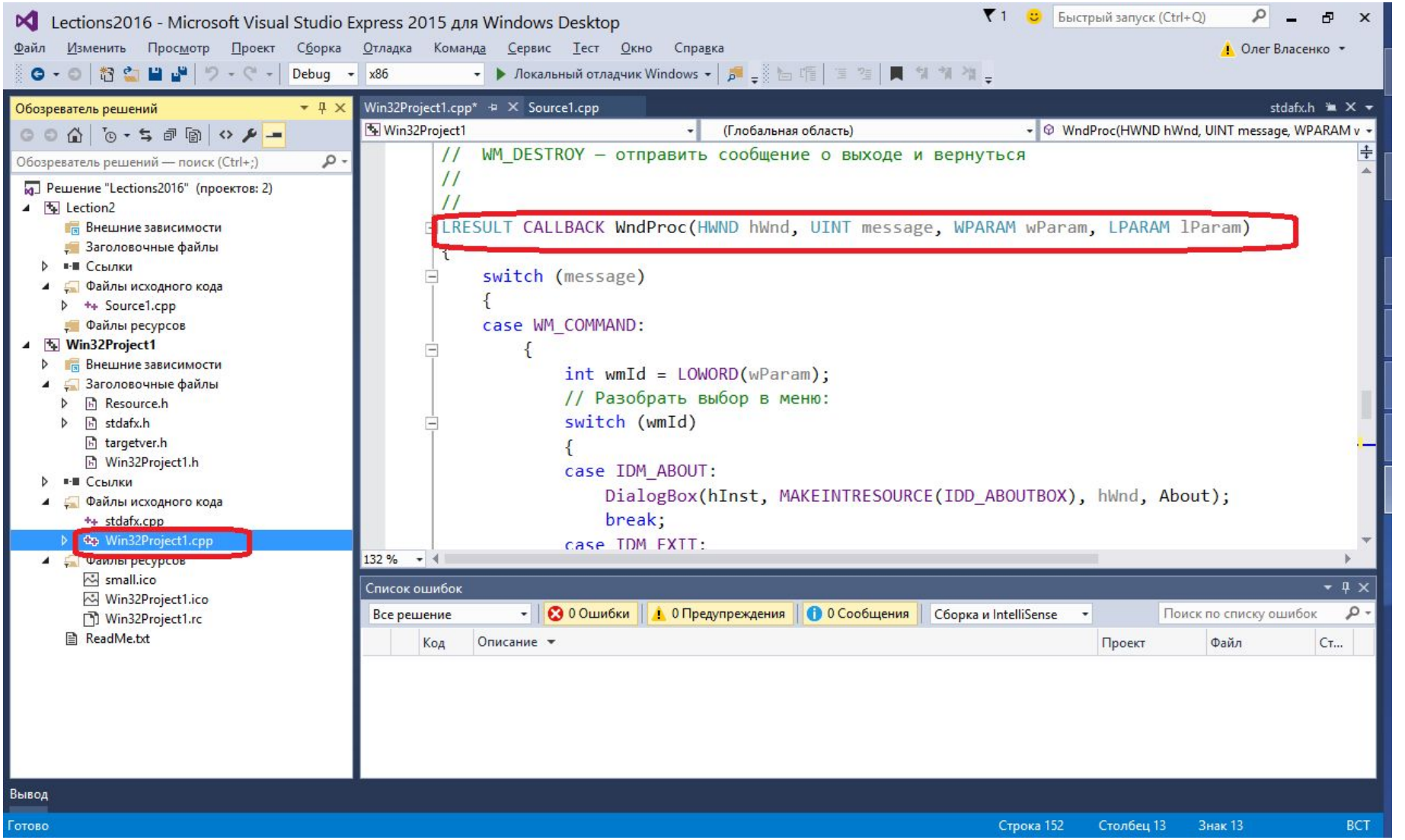
Запущенное win32 приложение!



Где в коде рисовать картинки?

Файл Win32Project1.cpp

Функция WndProc()

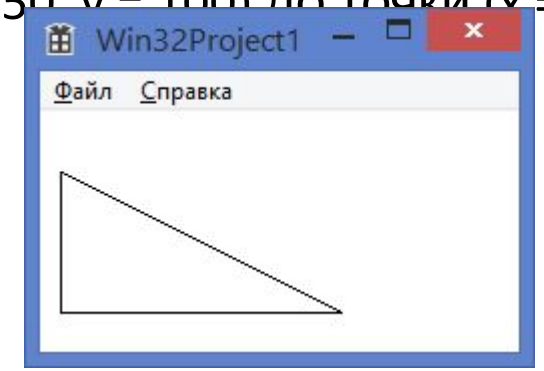


Где в коде рисовать картинки? (2)

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
   switch (message)
   {
…
   case WM_PAINT:
      {
         PAINTSTRUCT ps;
         HDC hdc = BeginPaint(hWnd, &ps);

         EndPaint(hWnd, &ps);
      }
      break;
 …
   default:
      return DefWindowProc(hWnd, message, wParam, lParam);
   }
   return 0;
}
```

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);
    }
```



https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Текущая позиция пера

Для рисования прямых линий (и только для этого) в контексте отображения хранятся координаты текущей позиции пера . Для изменения текущей позиции пера в Windows версии 3.1 есть две функции с именами MoveTo и MoveToEx . Для совместимости с 32-разрядными версиями Windows, такими, как Windows NT, в новых приложениях следует использовать функцию MoveToEx:

```
BOOL WINAPI MoveToEx(
  HDC hdc,          // идентификатор контекста отображения
  int x,            // x-координата
  int y,            // y-координата
  POINT FAR* lppt); // указатель на структуру POINT
```

Для контекста отображения hdc эта функция устанавливает текущую позицию пера, равную (x,y). В структуру типа POINT, на которую указывает параметр lppt, после возврата из функции будут записаны старые координаты пера.

Функция MoveToEx возвращает TRUE при нормальном завершении или FALSE при ошибке.

https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Рисование прямой линии

Для того чтобы нарисовать прямую линию, приложение должно воспользоваться функцией LineTo :

```
BOOL WINAPI LineTo(HDC hdc, int xEnd, int yEnd);
```

Эта функция рисует линию из текущей позиции пера, установленной ранее функцией MoveTo или MoveToEx, в точку с координатами (xEnd,yEnd). После того как линия будет нарисована, текущая позиция пера станет равной (xEnd,yEnd).

Функция LineTo возвращает TRUE при нормальном завершении или FALSE при ошибке.

Таким образом, для того чтобы нарисовать прямую линию, приложение должно сначала с помощью функции MoveToEx установить текущую позицию пера в точку, которая будет началом линии, а затем вызвать функцию LineTo, передав ей через параметры xEnd и yEnd координаты конца линии.

Особенностью функции LineTo является то, что она немного не дорисовывает линию - эта функция рисует всю линию, не включая ее конец, т. е. точку (xEnd,yEnd). Это иллюстрируется на рис. 2.11.

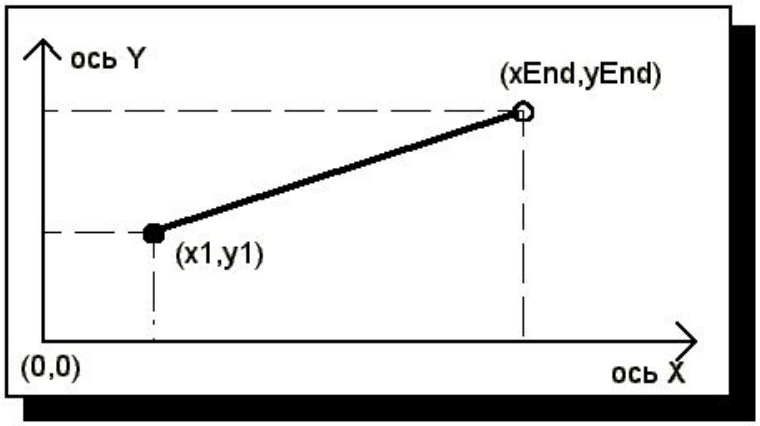
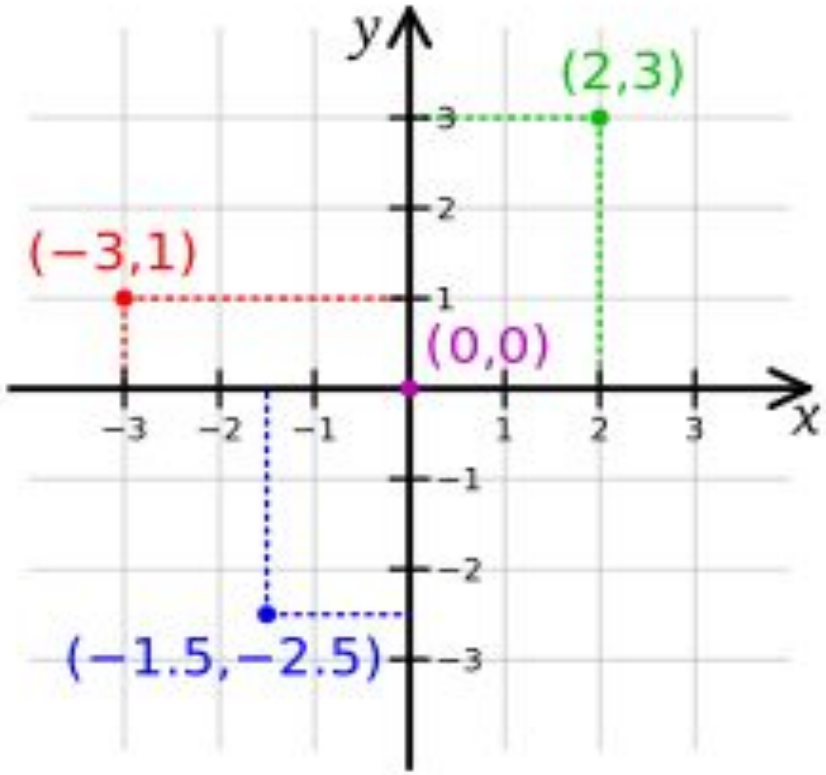
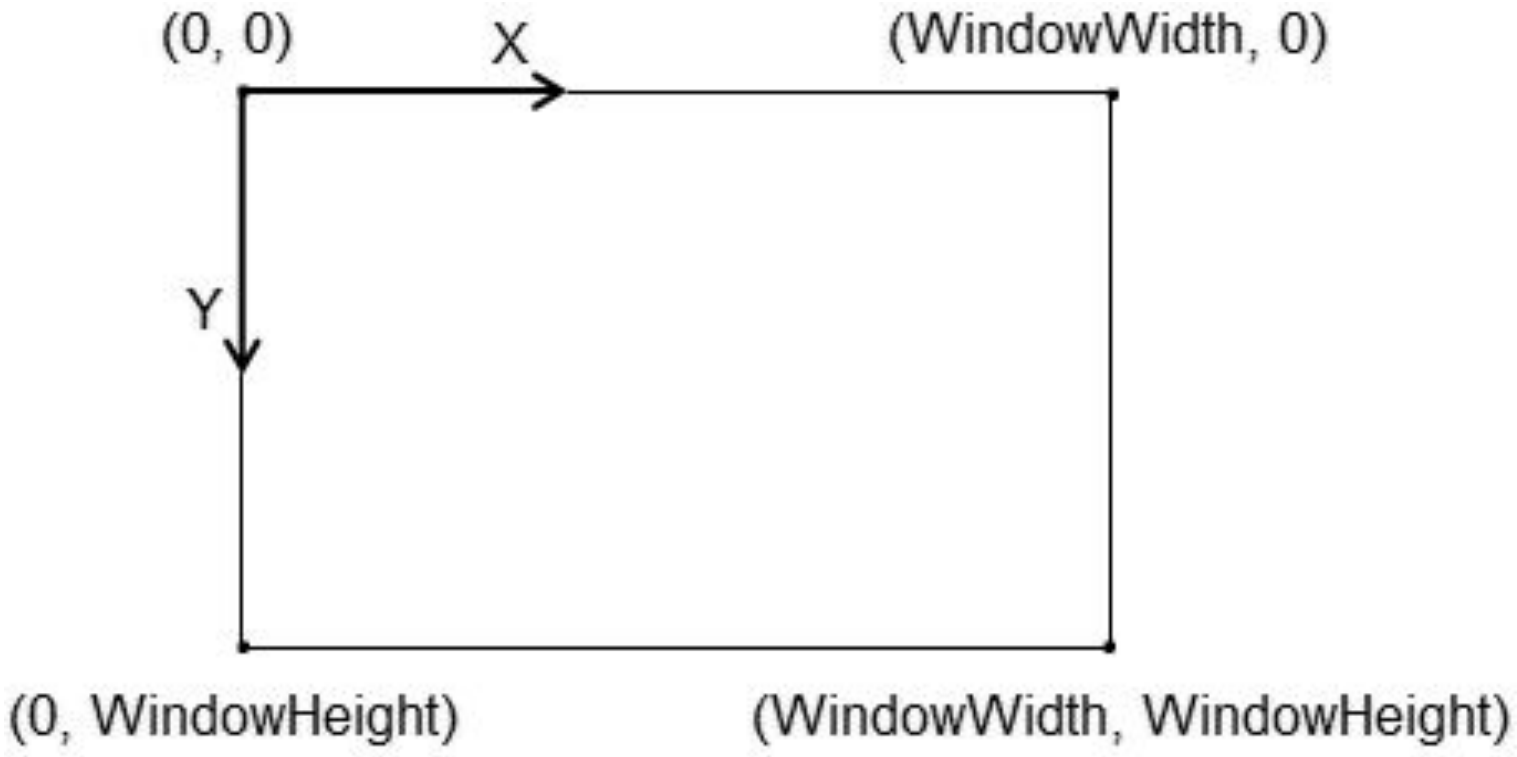


Рис. 2.11. Рисование прямой линии

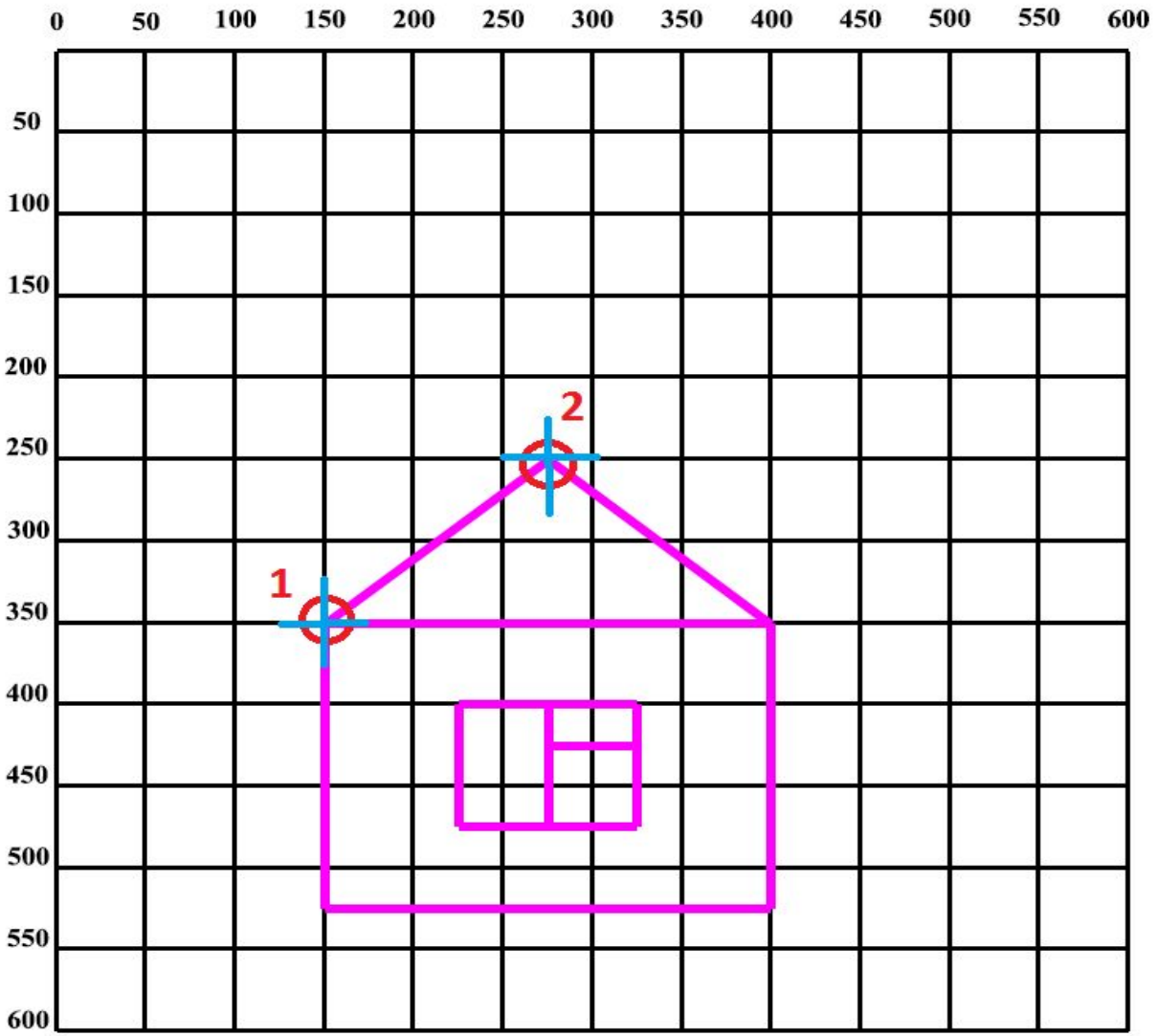
Точки в Декартовой системе координат



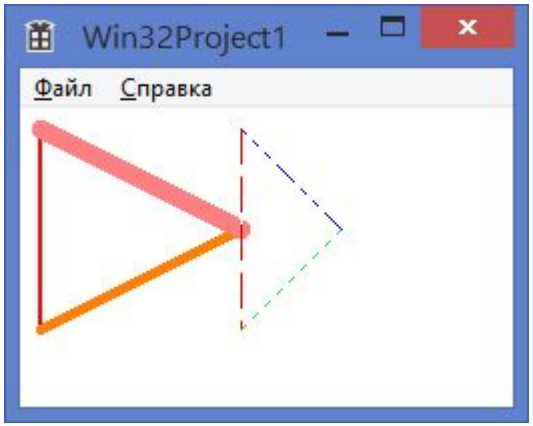
Экранная система координат



Оцифровка точек в координатной сетке



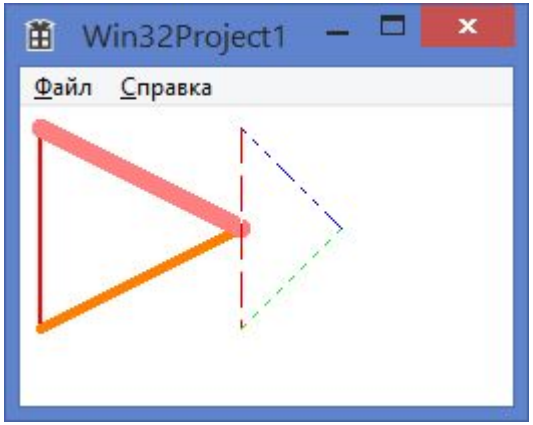
Такое разное перо



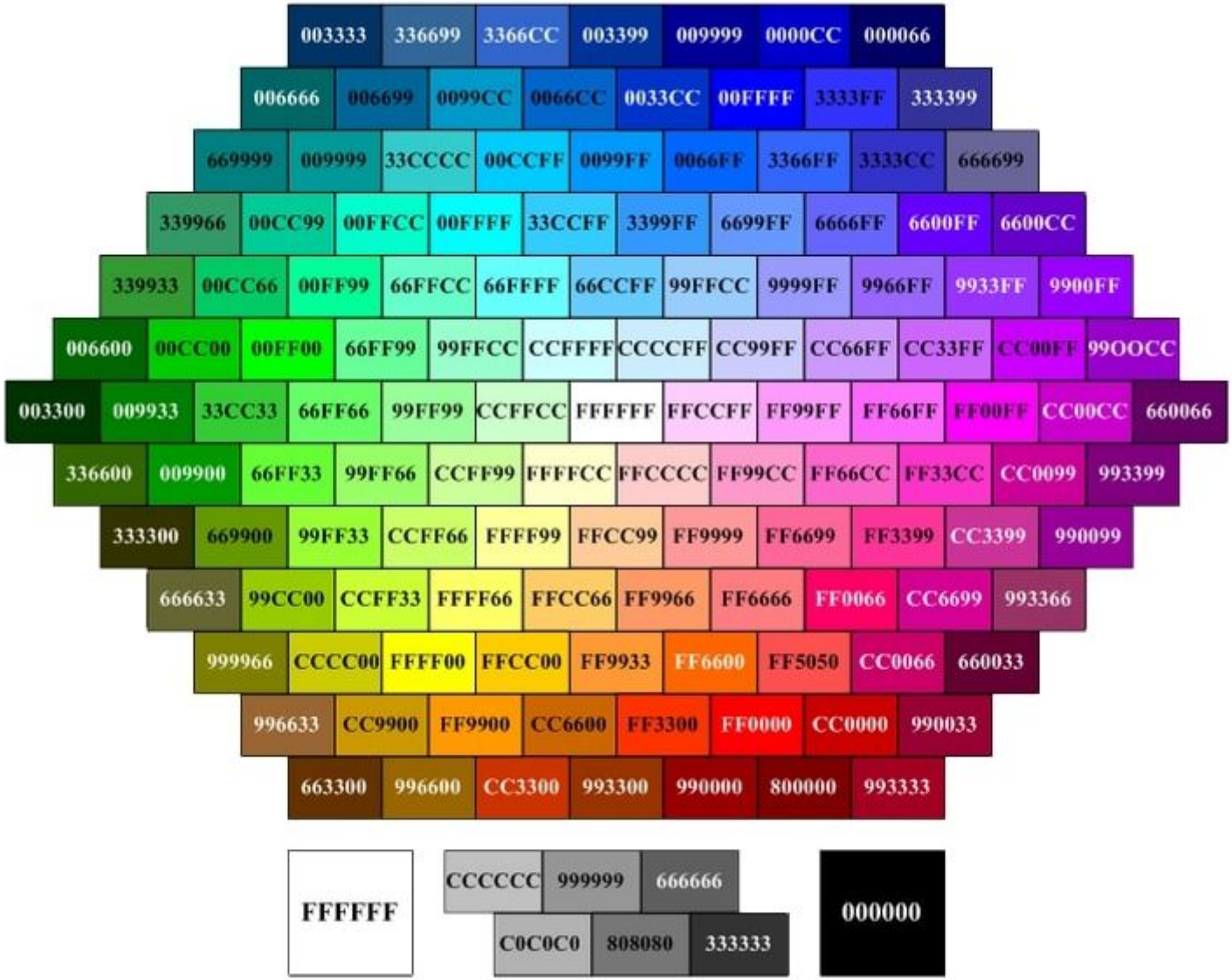

```
case WM_PAINT:   {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
    HPEN hPen;
    hPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
    SelectObject(hdc, hPen);
    MoveToEx(hdc, 10, 10, NULL);
    LineTo(hdc, 10, 110);

    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);
 }
```

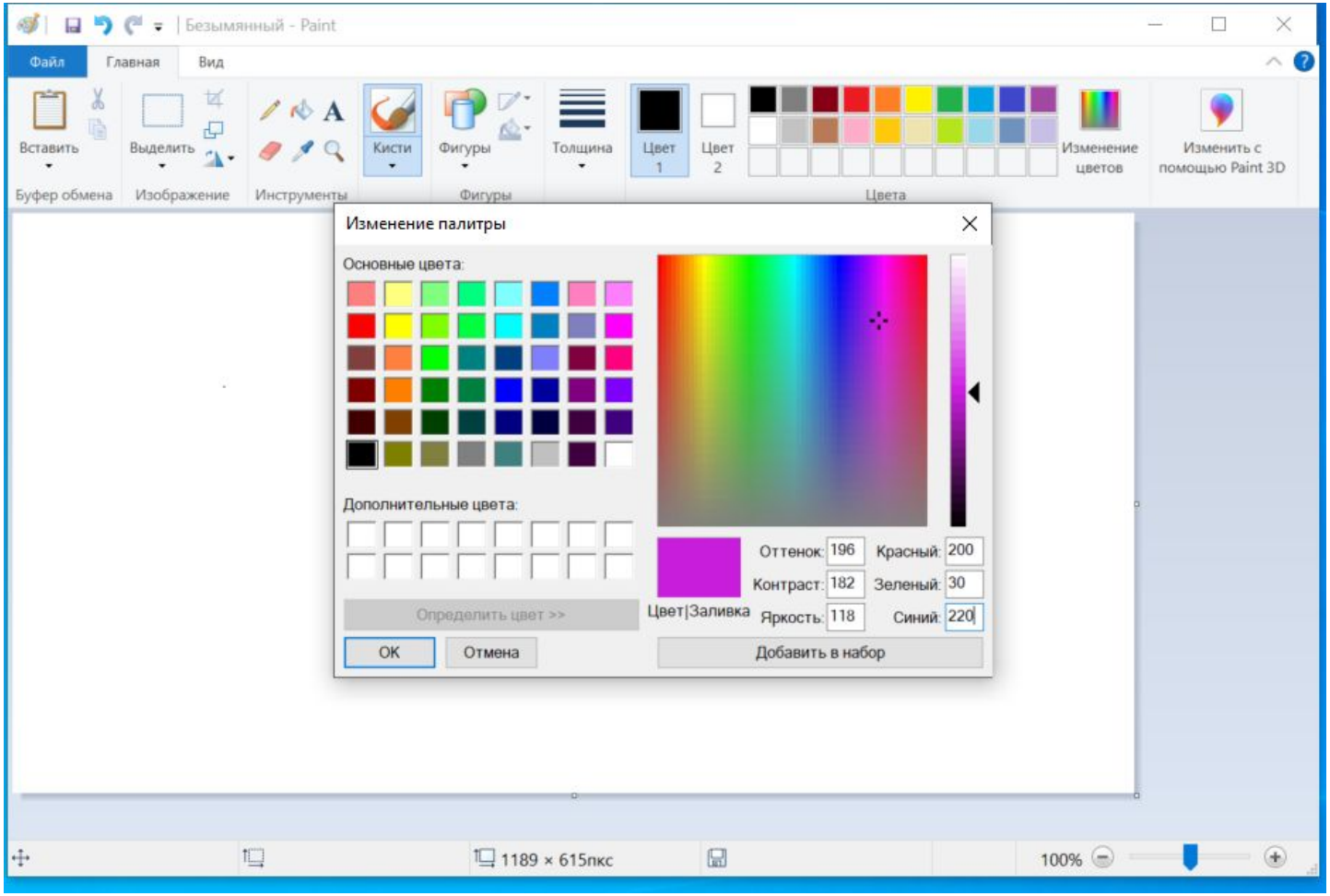


Некоторые цвета



https://www.mta-resource.ru/wiki/article/id/23

Как задать произвольный цвет

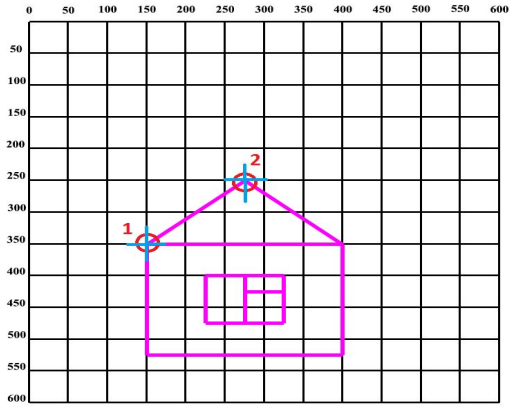


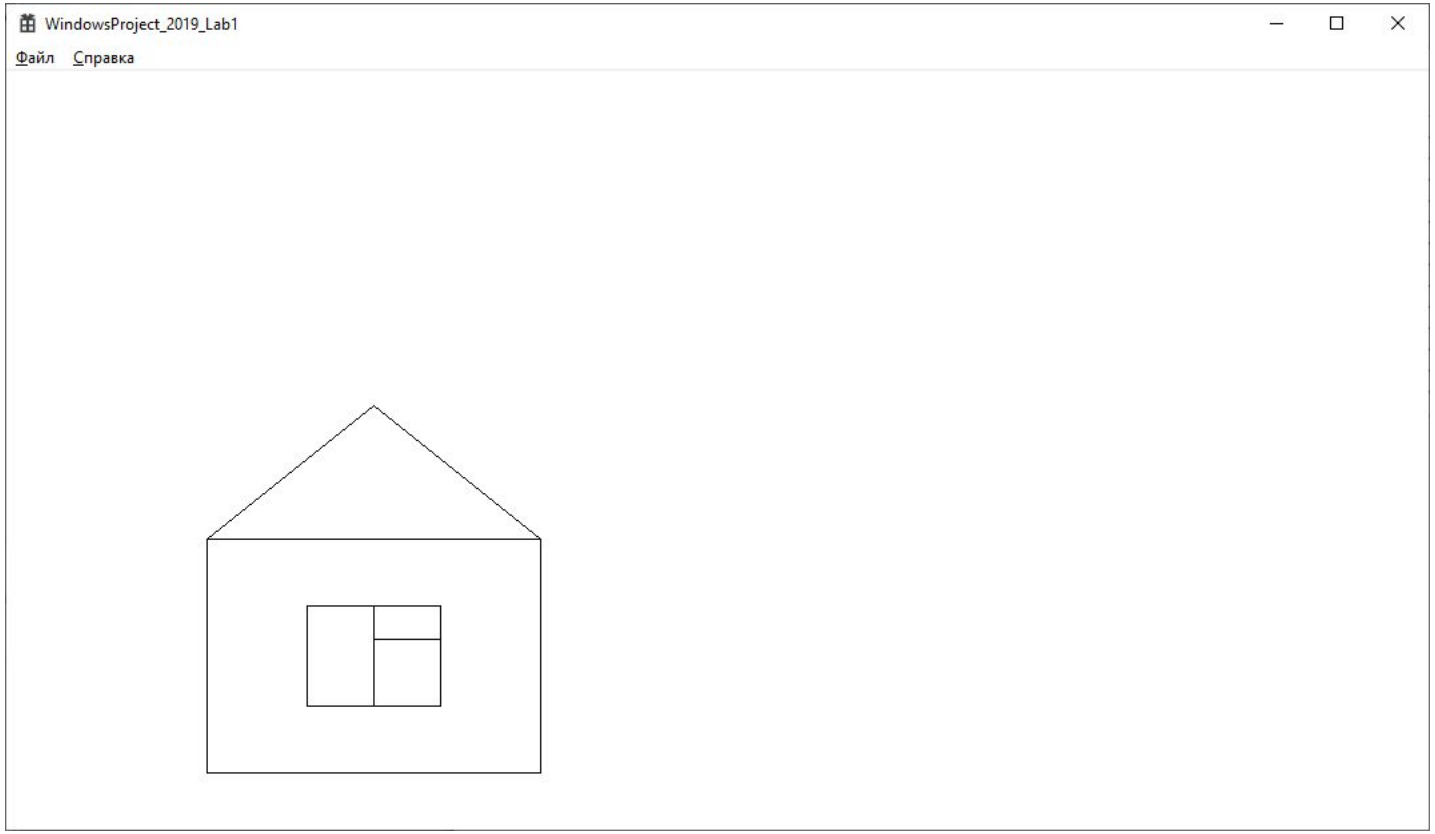
Лабораторная работа №4

Создание изображения из линий

Задача 1

В Windows приложение добавить рисунок домика.





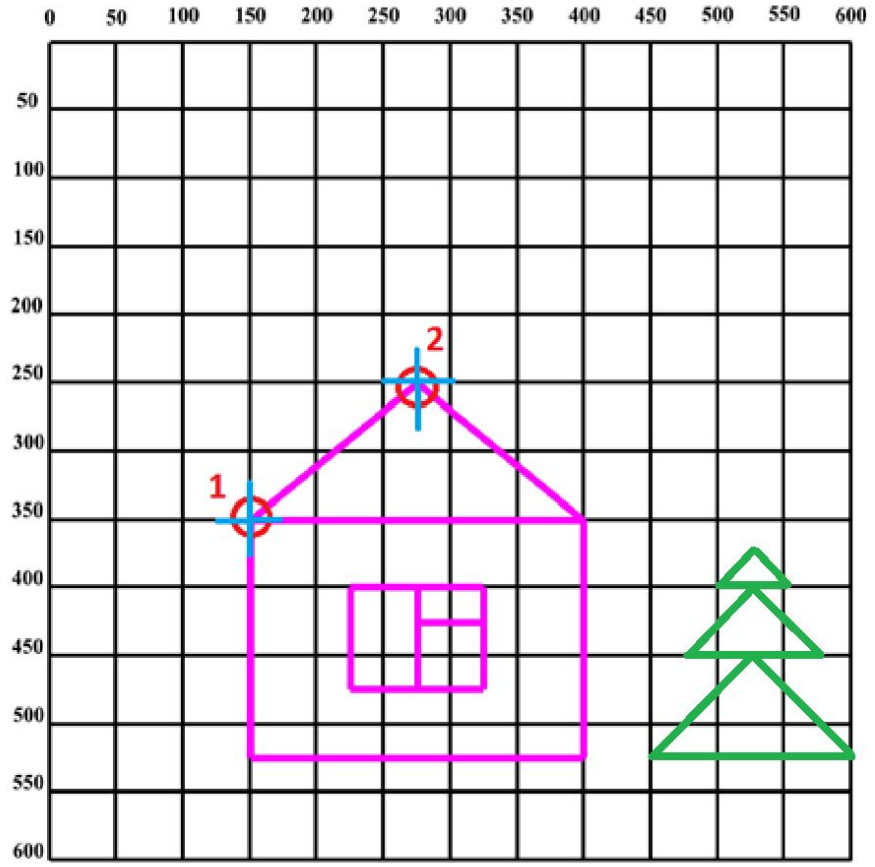
Задача 1: Решение

В Windows приложение добавить рисунок домика.

```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        // Крыша
        MoveToEx(hdc, 150, 350, NULL);
        LineTo(hdc, 275, 250);
        LineTo(hdc, 400, 350);
        // Дом
        LineTo(hdc, 400, 525);
        LineTo(hdc, 150, 525);
        LineTo(hdc, 150, 350);
        LineTo(hdc, 400, 350);
        // Окно
        MoveToEx(hdc, 225, 400, NULL);
        LineTo(hdc, 225, 475);
        LineTo(hdc, 325, 475);
        LineTo(hdc, 325, 400);
        LineTo(hdc, 225, 400);
        // Рама
        MoveToEx(hdc, 275, 400, NULL);
        LineTo(hdc, 275, 475);
        MoveToEx(hdc, 275, 425, NULL);
        LineTo(hdc, 325, 425);

        EndPaint(hWnd, &ps);
    }
    break;
case WM_DESTROY:
```

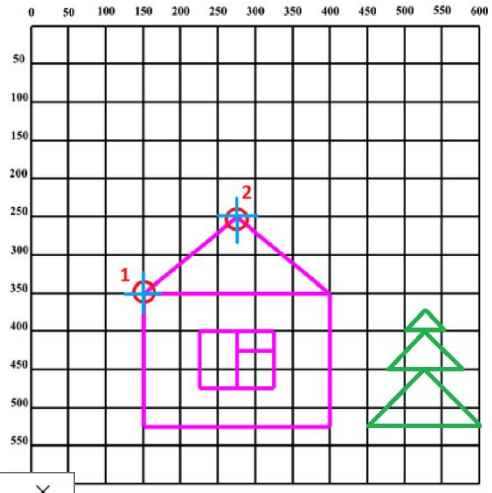
Задача 2

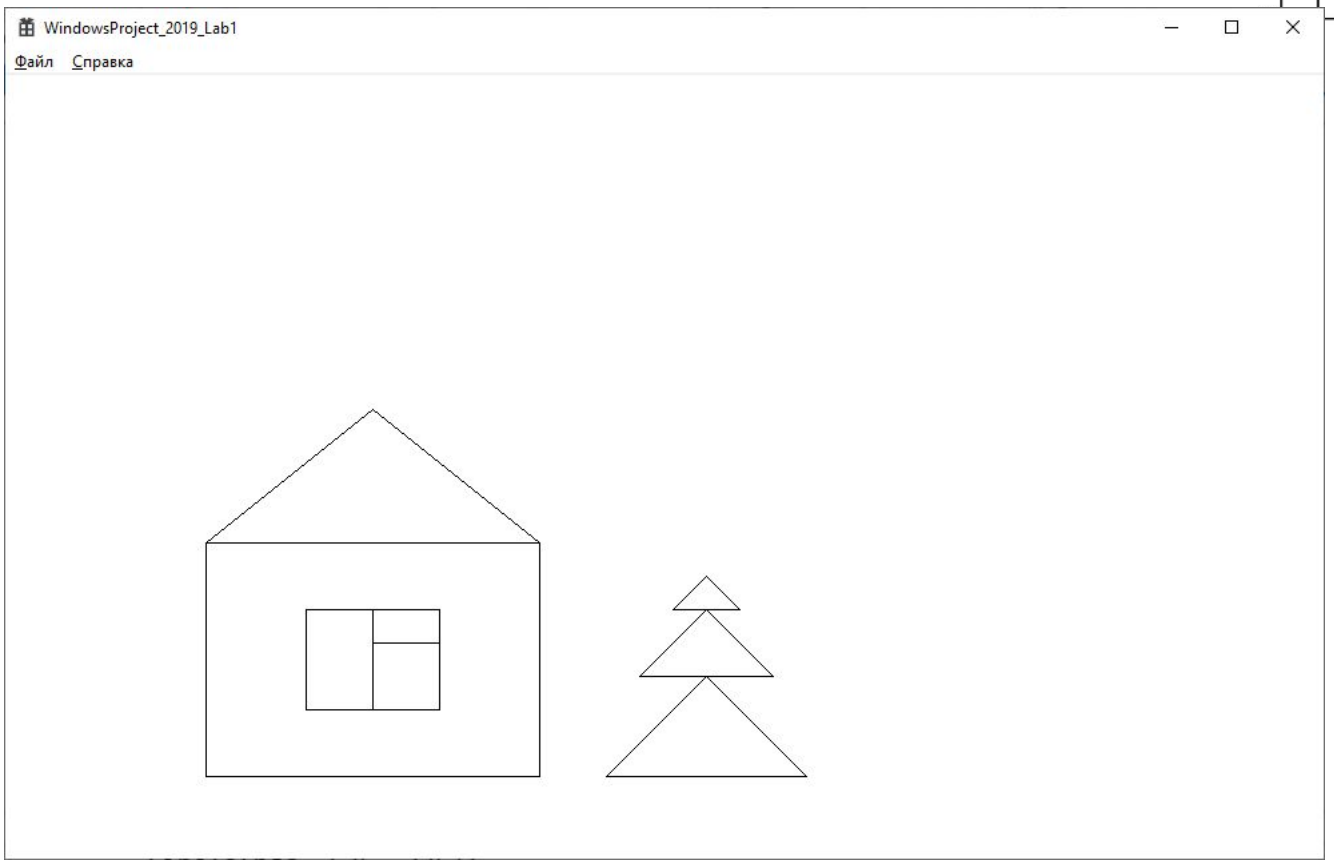
Рядом с домиком добавьте ёлочку



Задача 2

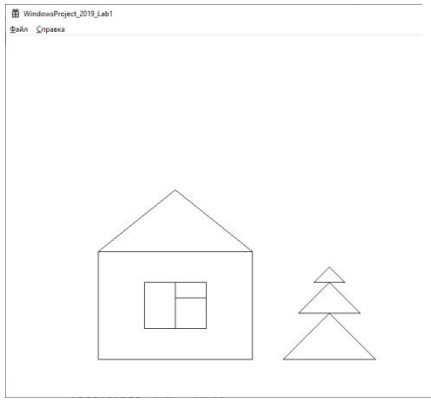
Рядом с домиком добавьте ёлочку

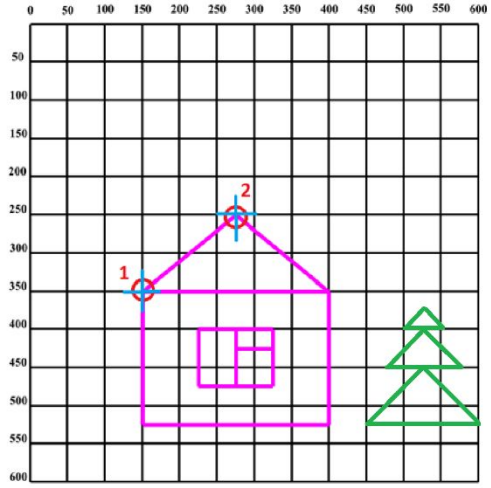




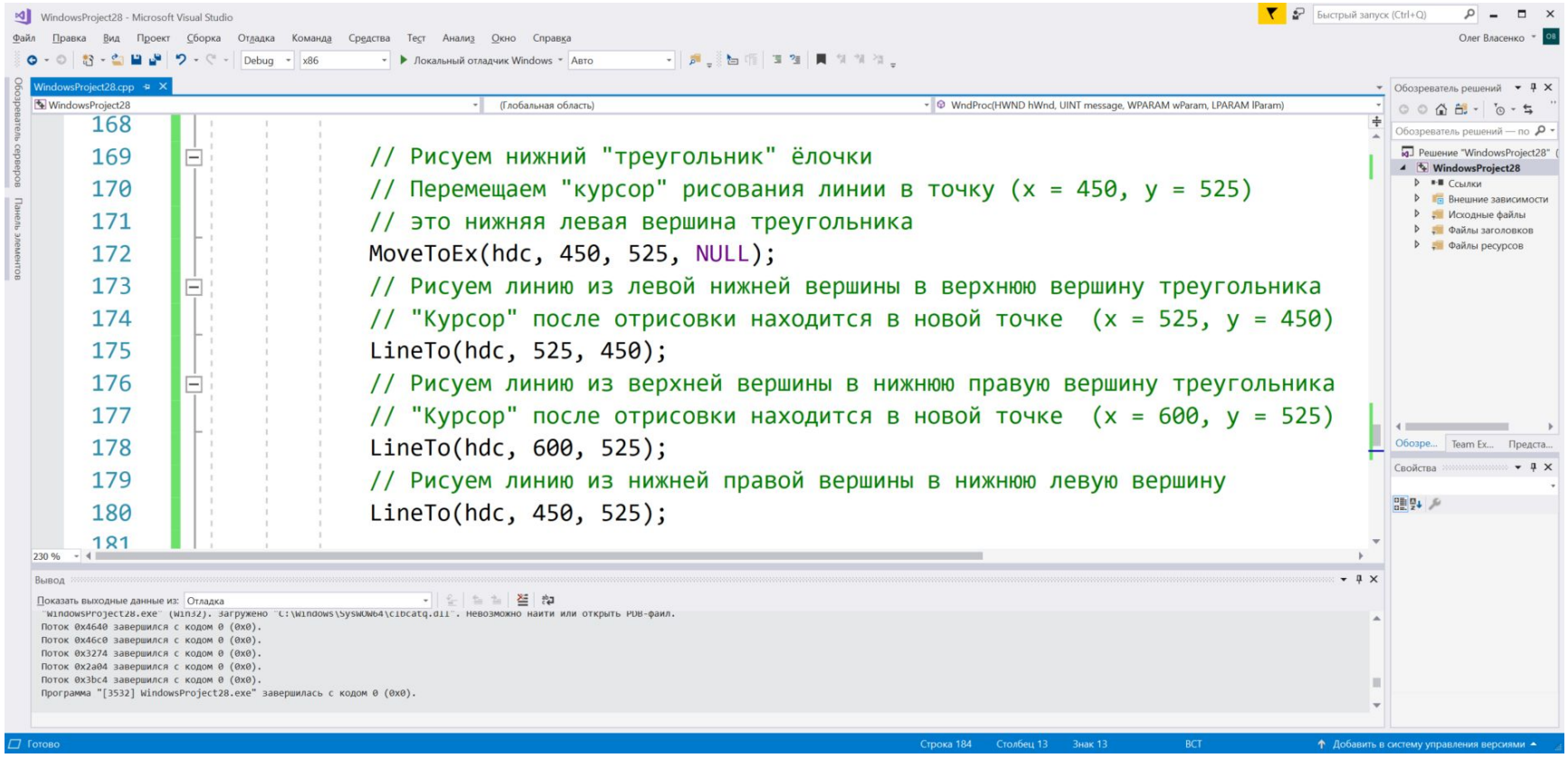
Задача 2

Рядом с домиком добавьте ёлочку



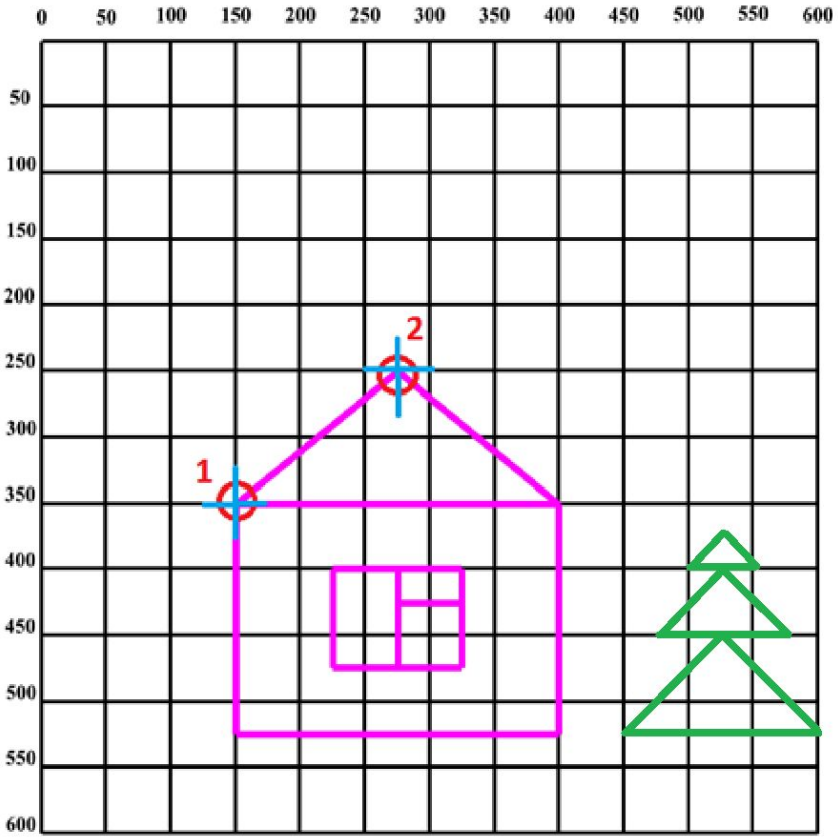


Код отрисовки нижнего треугольника.
Остальной код отрисовки
нужно самостоятельно написать



Задача 3

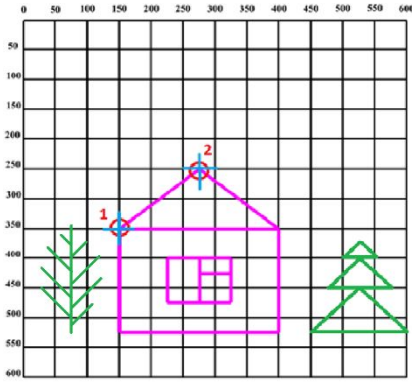
Дом и елочку нужно сделать цветными – творчество в подборе цвета приветствуется.



Домашнее задание к ЛР4

0. Доделать дома задачи 1-3 – если они не были доделаны в классе.

1. Нарисовать слева от дома дерево. Можно использовать за образец картинку ниже. Можно свое собственное. Нужно активно и творчески использовать цвет и толщину пера.



2. К получившейся картине добавить сооружение (сарай, мост, и т.п.) нарисованное не менее чем 15 линиями. Нужно активно и творчески использовать цвет и толщину пера.

3. * Добавить что-то от себя, состоящее из линий (скамейка, забор, сугробы, и т.п.). Нужно активно и творчески использовать цвет и толщину пера.

Задания со звездочками - необязательные

Домашнее задание – оформление

Для сдачи домашней работы нужно иметь:

1) Доказательство того, что вы рассчитывали координаты концов линий. Если расчеты были на бумаге – нужна эта бумага.

Рисуем разные фигуры

https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Рисование прямоугольника

Простейшая функция, с помощью которой можно нарисовать прямоугольник, называется Rectangle :

```
BOOL WINAPI Rectangle(
  HDC hdc,    // идентификатор контекста отображения
  int nxTL,   // координата x верхнего левого угла
  int nyTL,   // координата y верхнего левого угла
  int nxBR,   // координата x правого нижнего угла
  int nyBR); // координата y правого нижнего угла
```

Функция Rectangle рисует прямоугольник для контекста отображения hdc, возвращая значение TRUE в случае успеха или FALSE при ошибке.

Назначение остальных параметров иллюстрируется рис. 2.17.

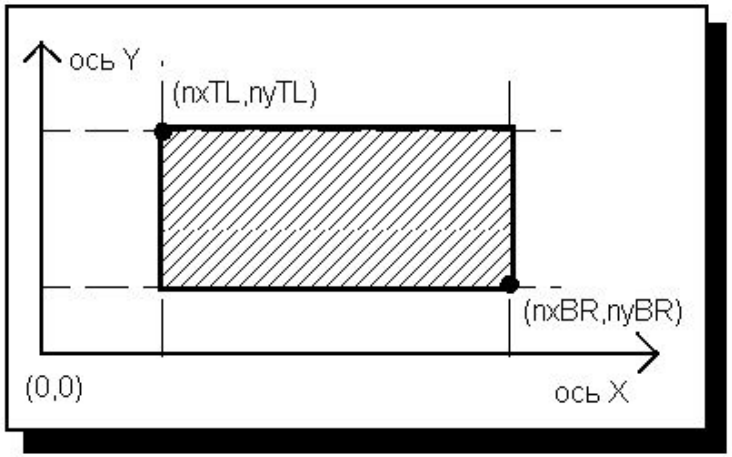


Рис. 2.17. Рисование прямоугольника

https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Функция FillRect закрашивает прямоугольную область окна заданной кистью:

```
int WINAPI FillRect(
  HDC hdc,   // идентификатор контекста отображения
  const RECT FAR* lprc, // указатель на структуру RECT
  HBRUSH hbrush); // идентификатор кисти для закрашивания
```

Параметр lprc должен указывать на структуру типа RECT, в которую следует записать координаты закрашиваемой прямоугольной области. Правая и нижняя граница указанной области не закрашивается.

Независимо от того, какая кисть выбрана в контекст отображения, функция FillRect будет использовать для закраски кисть, указанную параметром hbrush.

Учтите, что правильная работа функции FillRect гарантируется только в том случае, когда значение поля bottom структуры RECT больше значения поля top, а значение поля right больше значения поля left.

Для закрашивания границы прямоугольной области (т. е. для рисования прямоугольной рамки) можно использовать функцию FrameRect :

```
int WINAPI FrameRect(
  HDC hdc,   // идентификатор контекста отображения
  const RECT FAR* lprc, // указатель на структуру RECT
  HBRUSH hbrush); // идентификатор кисти для закрашивания
```

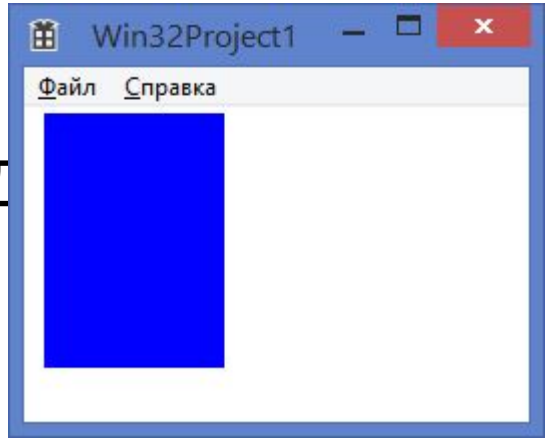
Параметры этой функции аналогичны параметрам функции FillRect.

Ширина пера, используемого для рисования рамки, всегда равна одной логической единице. Структура RECT должна быть подготовлена таким же образом, что и для функции FillRect, т. е. значение поля bottom структуры RECT должно быть больше значения поля top, а значение поля right - больше значения поля left.

Значение, возвращаемое функциями FillRect и FrameRect не используется, приложения должны его игнорировать.

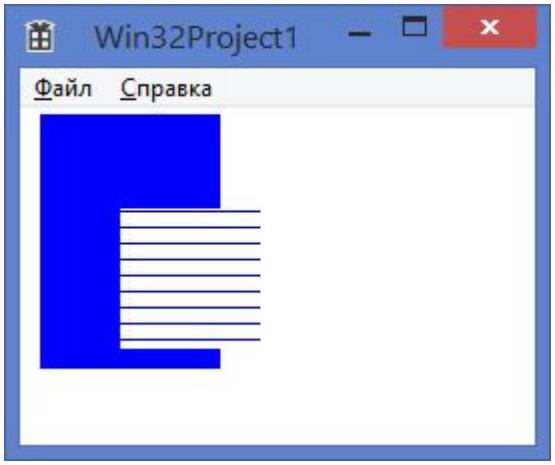
Кисть

```
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 СЮД

    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);
```



А если нужно что-то круглое?

Чем будем рисовать

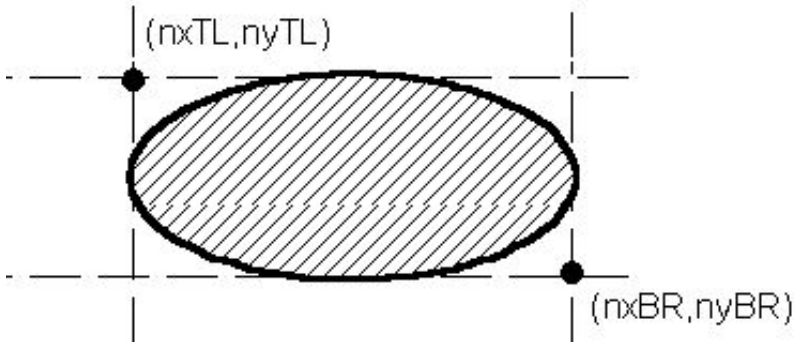
https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Рисование эллипса

Для рисования эллипса вы можете использовать функцию Ellipse :

```
BOOL WINAPI Ellipse(
    HDC hdc, // идентификатор контекста отображения
    int nxTL, // координата x верхнего левого угла
    int nyTL, // координата y верхнего левого угла
    int nxBR, // координата x правого нижнего угла
    int nyBR); // координата y правого нижнего угла
```

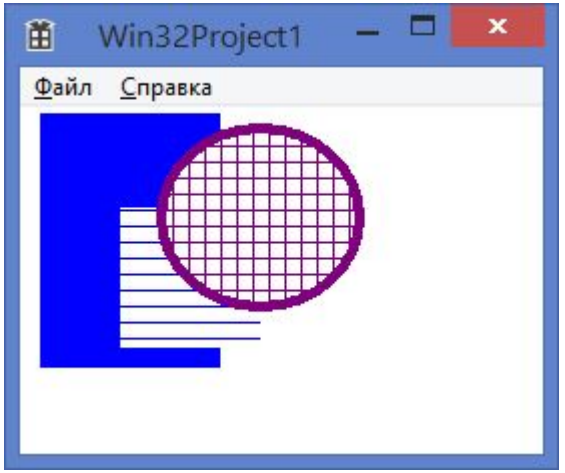
Первый параметр этой функции указывает идентификатор контекста отображения, остальные - координаты верхнего левого и правого нижнего углов прямоугольника, в который должен быть вписан эллипс



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);



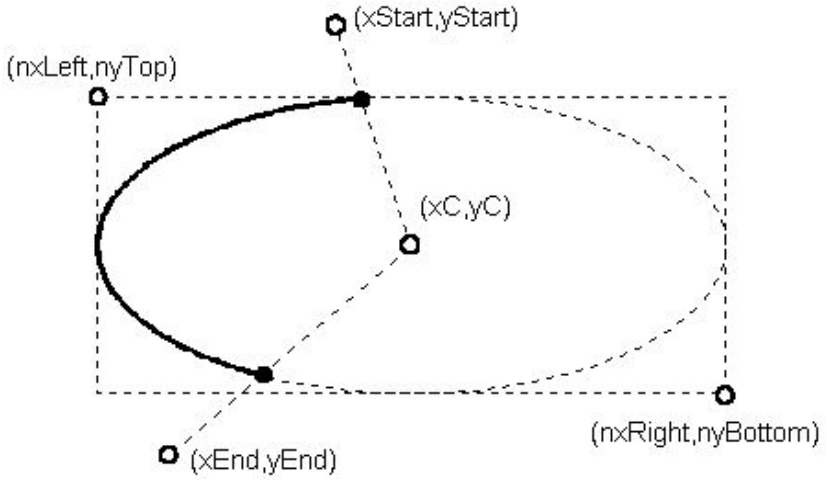
Чем еще можно рисовать

https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Рисование дуги эллипса

функция Arc позволяет нарисовать дугу эллипса или окружности:

```
BOOL WINAPI Arc(
    HDC hdc, // идентификатор контекста отображения
    int nxLeft, int nyTop, // верхний левый угол
    int nxRight, int nyBottom, // правый нижний угол
    int nxStart, int nyStart, // начало дуги
    int nxEnd, int nyEnd); // конец дуги
```



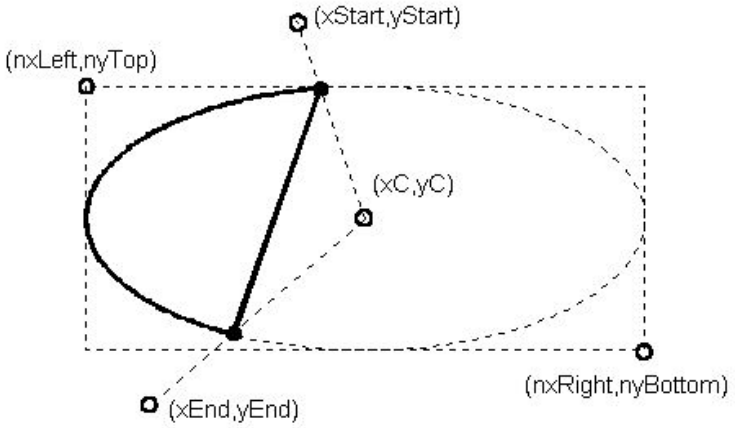
Чем еще можно рисовать

https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Рисование сегмента эллипса

Сегмент эллипса можно нарисовать при помощи функции Chord :

```
BOOL WINAPI Chord(
    HDC hdc, // идентификатор контекста отображения
    int nxLeft, int nyTop, // верхий левый угол
    int nxRight, int nyBottom, // правый нижний угол
    int nxStart, int nyStart, // начало дуги
    int nxEnd, int nyEnd); // конец дуги
```



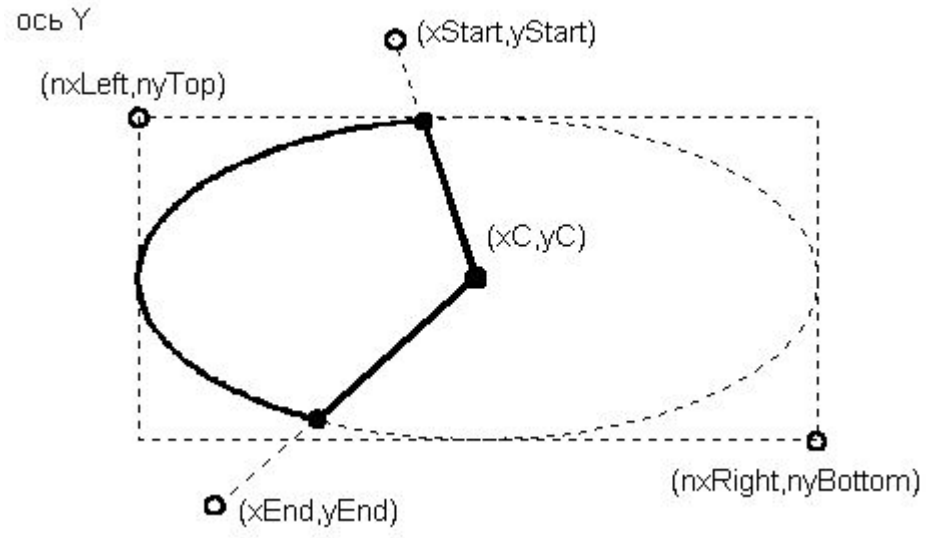
Чем еще можно рисовать

https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

Рисование сектора эллипса

Для рисования сектора эллипса следует использовать функцию Pie , аналогичную по своим параметрам функциям Arc и Chord:

```
BOOL WINAPI Pie(
    HDC hdc, // идентификатор контекста отображения
    int nxLeft, int nyTop, // верхний левый угол
    int nxRight, int nyBottom, // правый нижний угол
    int nxStart, int nyStart, // начало дуги
    int nxEnd, int nyEnd); // конец дуги
```



Источники информации

- Рисование геометрических фигур - https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

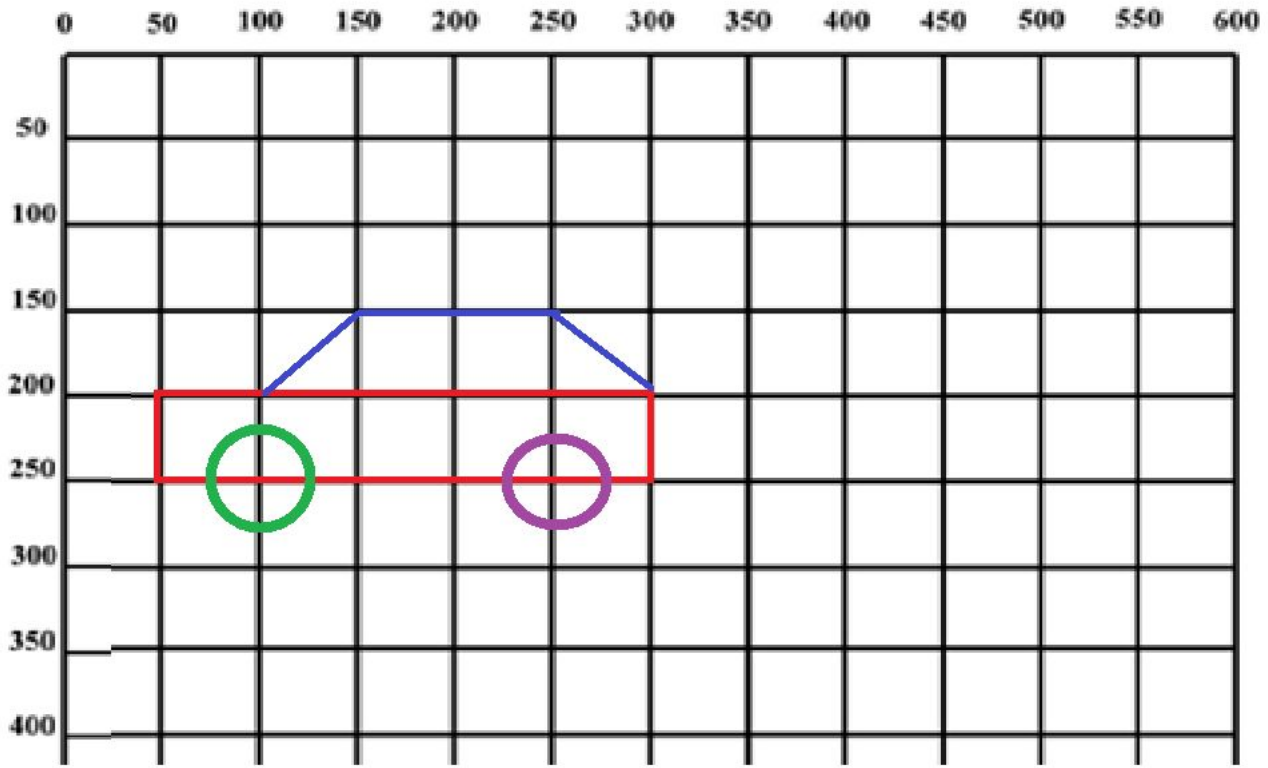
- КАК рисовать в Win32 API? -http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html

Лабораторная работа №5

Создание изображения из прямоугольников, эллипсов и др.

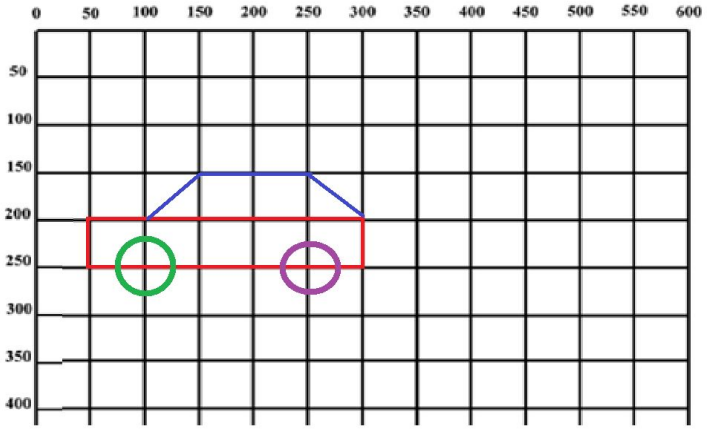
Задача 1

Нарисовать автомобиль
Использовать 3 разных кисти.
В рисунке есть 2 круга.



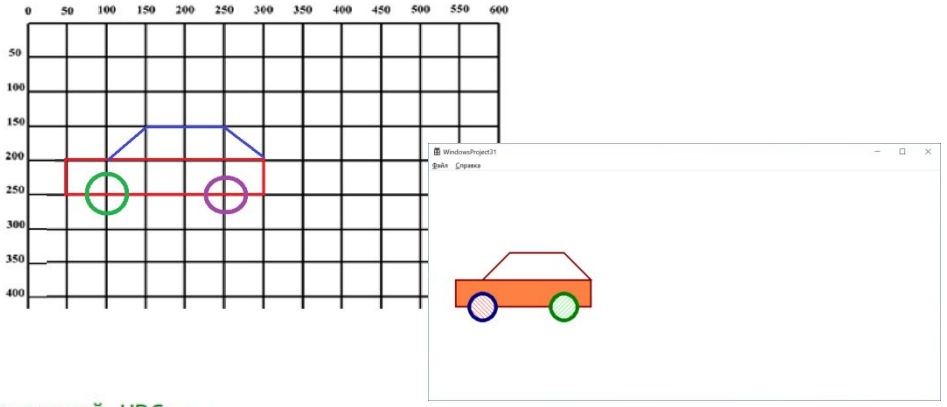
Задача 1

Нарисовать автомобиль
Использовать 3 разных кисти.
В рисунке есть 2 круга.



Задача 1

Нарисовать автомобиль
Использовать 3 разных кисти.
В рисунке есть 2 круга.

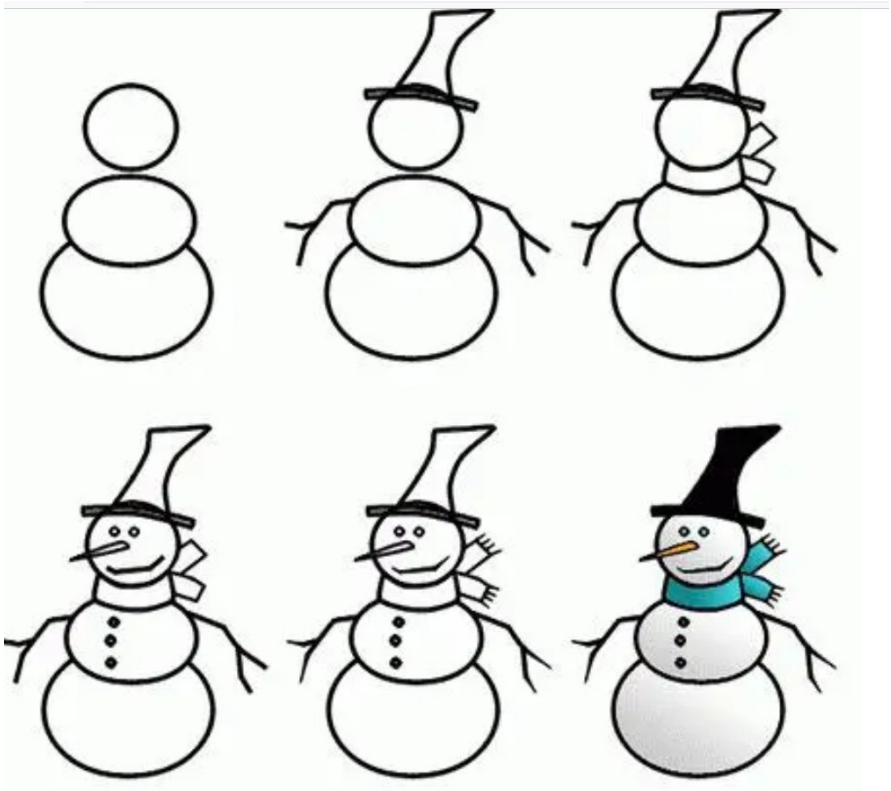



```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        HPEN hPen = CreatePen(PS_SOLID, 3, RGB(128, 0, 0));
        SelectObject(hdc, hPen);

        HBRUSH hBrush; //создаём объект-кисть
        hBrush = CreateSolidBrush(RGB(255, 128, 67)); //задаём сплошную кисть, закрашенную цветом RGB
        SelectObject(hdc, hBrush);

        Rectangle(hdc, 50, 200, 300, 250);

        MoveToEx(hdc, 100, 200, NULL);
        LineTo(hdc, 150, 150);
        LineTo(hdc, 250, 150);
        LineTo(hdc, 300, 200);

        hPen = CreatePen(PS_SOLID, 7, RGB(0, 0, 128));
        SelectObject(hdc, hPen);

        hBrush = CreateHatchBrush(HS_FDIAGONAL, RGB(255, 0, 0)); //задаём диагональную кисть, закрашенную цветом RGB
        SelectObject(hdc, hBrush);

        Ellipse(hdc, 75, 225, 125, 275);
```

Задача 2

Добавить к картинке снежную бабу.
В рисунке должно быть 3 закрашенных эллипса. Руки, нос и другие элементы можно изобразить линиями.

(рисунок ниже для вдохновения)



Домашнее задание

0. Доделать дома задачи 1-2 – если они не были доделаны в классе.

1. К картине с машиной и снежной бабой, добавить рисунок транспортного средства с колесами – грузовой автомобиль, автобус, и др..

2. Добавить сооружение, растение, или что-угодно иное, содержащее закрашенные прямоугольники. Нужно использовать максимальное количество разных кистей и перьев.

3. * Добавить облако, дерево, или что угодно другое, сделанное из множества эллипсов.

4. ** Добавить часы, мангал и т.п. сделанное из дуг, и других элементов.

Задания со звездочками - необязательные

Домашнее задание – оформление

Для сдачи домашней работы нужно иметь:

1) Доказательство того, что вы рассчитывали координаты концов линий. Если расчеты были на бумаге – нужна эта бумага.

ИТОГО по лекции

1. Обсудили что такое ОС, интерфейс, GUI, API, Windows API (WinAPI)
2. Узнали как создать WinAPI приложение в VS.
3. Узнали/повторили декартову систему координат.
4. Узнали про экранную систему координат.
5. Узнали как нарисовать рисунок из линий.
6. Узнали как задать стиль, цвет и толщину у пера.
7. Узнали как нарисовать прямоугольник и эллипс.
8. Узнали как создать кисть для рисования фигур.
9. Узнали что нужно делать в ЛР4 и ЛР5.

Источники информации

- Рисование геометрических фигур - https://www.frolov-lib.ru/books/bsp/v14/ch2_3.htm

- КАК рисовать в Win32 API? -http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html