

Курс «Основы программирования»

Власенко Олег Федосович

SimbirSoft

Лекция 5

Функции

ЛР 8. Упаковываем в функции ранее написанный код

ЛР 9. Использование функций для рисования объектов

Что такое подпрограмма?

Процедуры и функции в
Pascal.

```
procedure ReadArray;  
begin  
end;
```

```
function Abs(x:single): single;  
begin  
end;
```

Функции в *Ci.*

```
void read_array () {  
}
```

```
float abs(float f) {  
}
```

Зачем нужны подпрограммы?

Зачем нужны подпрограммы?

- Писать меньше кода - Повторяющийся код реализовать один раз, а вызывать многократно (sin(), printf() ...)
- Сделать код проще для редактирования - Разделить длинный код на части (произвольно)
- Упростить код - Разбить сложный алгоритм на части
- Повысить уровень абстракции – уйти от низкоуровневых операций на уровень предметной области

- Создавать библиотеки для повторного использования – стандартная библиотека Си состоит из функций
- Писать большие программы (до десятков и сотен тысяч строк кода)

Знакомство с функциями в Си

Простейшие функции

```
#include <stdio.h>
```

```
void a() {  
    printf("Hello! It is a()!\n");  
}
```

```
void main() {  
    printf("Hello! It is main()!\n");  
    a();  
}
```

Простейшие функции (2)

The image shows a screenshot of a C++ IDE window titled "Source.cpp". The code defines two functions: `main()` and `a()`. The `main()` function calls `a()`. The IDE's error list at the bottom shows two errors: C4326 (warning) and C3861 (error).

```
1  #include <stdio.h>
2
3
4
5  void main() {
6      printf("Hello! It is main()!\n");
7      a();
8  }
9
10 void a() {
11     printf("Hello! It is a()!\n");
12 }
13
```

Список ошибок

Все решение 1 Ошибка 1 Предупреждение 0 Сообщения

Код	Описание
C4326	возвращаемый тип "main" должен быть "int", а не "void"
C3861	а: идентификатор не найден

Простейшие функции (3)

```
#include <stdio.h>
```

```
// это - ОБЪЯВЛЕНИЕ функции a()
```

```
void a();
```

```
// это - ОПРЕДЕЛЕНИЕ функции main()
```

```
void main() {
```

```
    // это ВЫЗОВ функции printf()
```

```
    printf("Hello! It is main()!\n");
```

```
    // это ВЫЗОВ функции a()
```

```
    a();
```

```
}
```

```
// это - ОПРЕДЕЛЕНИЕ функции a()
```

```
void a() {
```

```
    // это ВЫЗОВ функции printf()
```

```
    printf("Hello! It is a()!\n");
```

```
}
```


Функция может возвращать результат

```
#include <stdio.h>
```

```
// это - ОПРЕДЕЛЕНИЕ функции main() возвращающей значение типа int
int main() {
    // это ВЫЗОВ функции printf()
    printf("Hello! It is main()!\n");

    // из main() возвращаем 0 – говорим ОС что у нас всё ОК
    return 0;
}
```

<http://www.c-cpp.ru/books/znacheniya-vozvrashchaemye-funkciey-main>

Когда используется оператор `return` в `main()`, программа возвращает код завершения вызывавшему процессу (операционной системе). Возвращаемое значение должно быть целого типа. Большинство операционных систем, трактуют 0 как нормальное завершение программы. Остальные значения воспринимаются как ошибки.

Функция может получать аргументы (1)

```
#include <stdio.h>
```

```
// это - ОБЪЯВЛЕНИЕ функции ndfl(), получающей один аргумент типа float,
```

```
// и возвращающей результат типа float
```

```
float ndfl(float s);
```

```
int main() {
```

```
    // это ВЫЗОВ функции printf() с одним аргументом "s = "
```

```
    printf("s = ");
```

```
    // переменные
```

```
    // s - численная зарплата
```

```
    // nalog - НДФЛ
```

```
    float s;
```

```
    float nalog;
```

```
    // это ВЫЗОВ функции scanf_s() с двумя аргументами -
```

```
    // 1: "%f"
```

```
    // 2: &s
```

```
    scanf_s("%f", &s);
```

Функция может получать аргументы (2)

```
// это ВЫЗОВ функции ndf1() с одним аргументом s
// Значение, возвращенное из функции ndf1() заносится
// (присваивается) в переменную nalog
nalog = ndf1(s);
```

```
// это ВЫЗОВ функции printf() с тремя аргументами
// 1: "s = %.2f, nalog = %.2f"
// 2: s
// 3: nalog
printf("s = %.2f, nalog = %.2f", s, nalog);
```

```
// из main() возвращаем 0 - говорим ОС что у нас всё ОК
return 0;
```

```
}
```

```
// это - ОПРЕДЕЛЕНИЕ функции ndf1(), получающей один аргумент типа
float,
```

```
// и возвращающей результат типа float
```

```
float ndf1(float s) {
```

```
    // возвращается результат, составляющий 13% от s
```

```
    return s * 0.13;
```

```
}
```

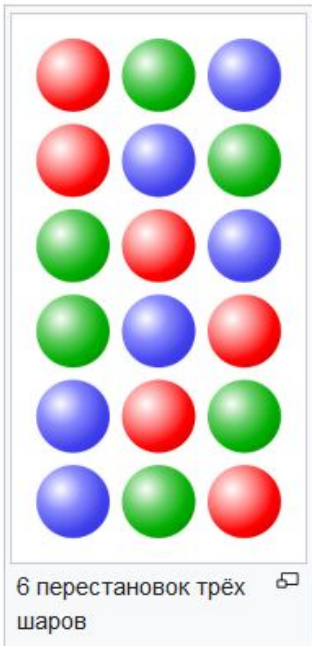
“Факториал” – что это такое

Факториал

[-https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B0%D0%BB](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B0%D0%BB)

Перестановка -

<https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0>



Факториал натурального числа n определяется как произведение всех натуральных чисел от 1 до n включительно:

$$n! = 1 \cdot 2 \cdot \dots \cdot n = \prod_{k=1}^n k.$$

Например,

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120.$$

Для $n = 0$ принимается в качестве соглашения, что

$$0! = 1.$$

Факториал активно используется в различных разделах математики: комбинаторике, математическом анализе, теории чисел, функциональном анализе и др.

Факториал является чрезвычайно быстро растущей функцией.

Функция может иметь свои (локальные) переменные (1)

```
#include <stdio.h>
```

```
// это - ОБЪЯВЛЕНИЕ функции fuct(), получающей один аргумент типа int,  
// и возвращающей результат типа long
```

```
long fuct(int n);
```

```
int main() {
```

```
    // локальная (в main()) переменная num, тип int
```

```
    int num;
```

```
    printf("num = ");
```

```
    scanf_s("%d", &num);
```

```
    // локальная (в main()) переменная f, тип long
```

```
    long f = fuct(num);
```

```
    printf("num = %d, num! = %ld", num, f);
```

```
    return 0;
```

```
}
```

Функция может иметь свои (локальные) переменные (2)

```
// это - ОПРЕДЕЛЕНИЕ функции fuct(), получающей один аргумент типа
int,
// и возвращающей результат типа long
long fuct(int n) {
    // локальная (в fuct()) переменная res, тип long
    long res = 1;
    // локальная (в fuct()) переменная i, тип int
    int i = 1;

    do {
        res = res * i;
        i = i + 1;
    } while (i <= n);

    return res;
}
```

Можно использовать глобальные переменные (1)

```
#include <stdio.h>
```

```
// Глобальная переменная - количество звезд
```

```
int numStars = 4;
```

```
// глобальная переменная - символ звездочки
```

```
char starSymbol = '*';
```

```
// это - ОБЪЯВЛЕНИЕ функции printStars(), не имеющей аргументов,
```

```
// и не возвращающей результата
```

```
void printStars();
```

```
// это - ОПРЕДЕЛЕНИЕ функции incStars(), не имеющей аргументов,
```

```
// и не возвращающей результата
```

```
void incStars() {
```

```
    numStars++;
```

```
}
```

```
// это - ОПРЕДЕЛЕНИЕ функции decStars(), не имеющей аргументов,
```

```
// и не возвращающей результата
```

```
void decStars() {
```

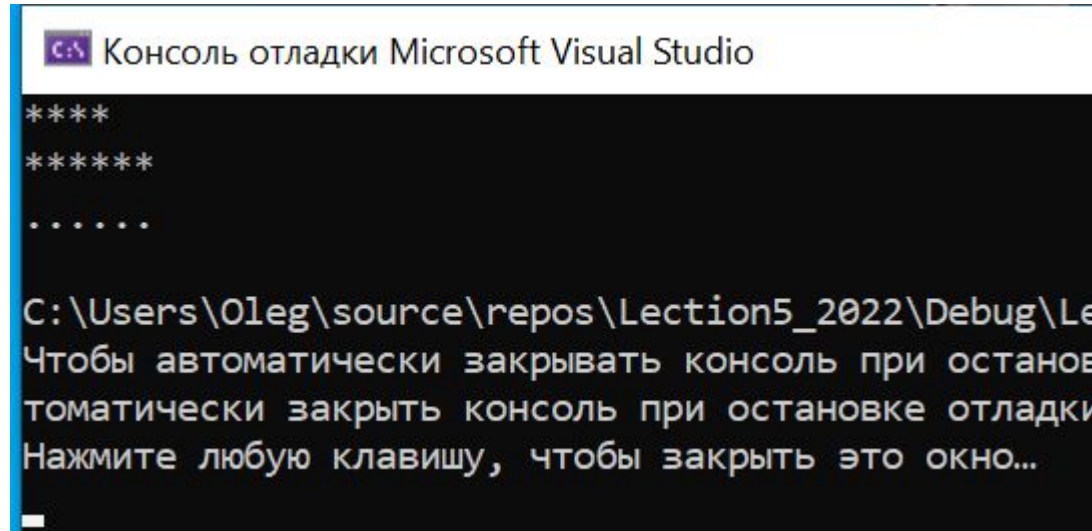
```
    numStars--;
```

```
}
```

Можно использовать глобальные переменные (2)

```
// это - ОПРЕДЕЛЕНИЕ функции main(), не имеющей аргументов,  
// и возвращающей результат типа int
```

```
int main() {  
    printStars();  
  
    incStars();  
    incStars();  
    printStars();  
  
    starSymbol = '.';  
    printStars();  
  
    return 0;  
}
```



```
C:\Users\Oleg\source\repos\Lectio5_2022\Debug\Le  
Чтобы автоматически закрывать консоль при остано  
томатически закрыть консоль при остановке отладки  
Нажмите любую клавишу, чтобы закрыть это окно...
```

```
// это - ОПРЕДЕЛЕНИЕ функции printStars(), не имеющей аргументов,  
// и не возвращающей результата
```

```
void printStars() {  
    int i = 1;  
    do {  
        printf("%c", starSymbol);  
        i++;  
    } while (i <= numStars);  
    printf("\n");  
}
```


НЕЛЬЗЯ вкладывать функции друг в друга

```
#include <stdio.h>
```

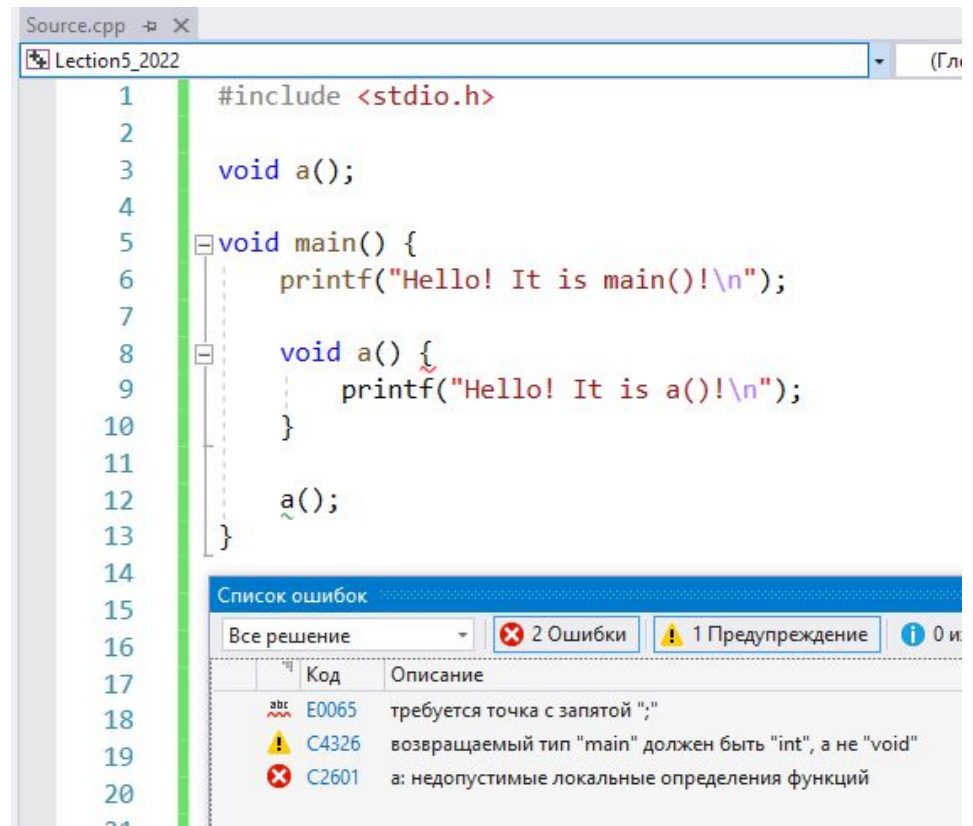
```
void a();
```

```
void main() {  
    printf("Hello! It is main()!\n");
```

```
    void a() {  
        printf("Hello! It is a()!\n");  
    }
```

```
    a();
```

```
}
```



The screenshot shows a C++ source file named 'Source.cpp' with the following code:

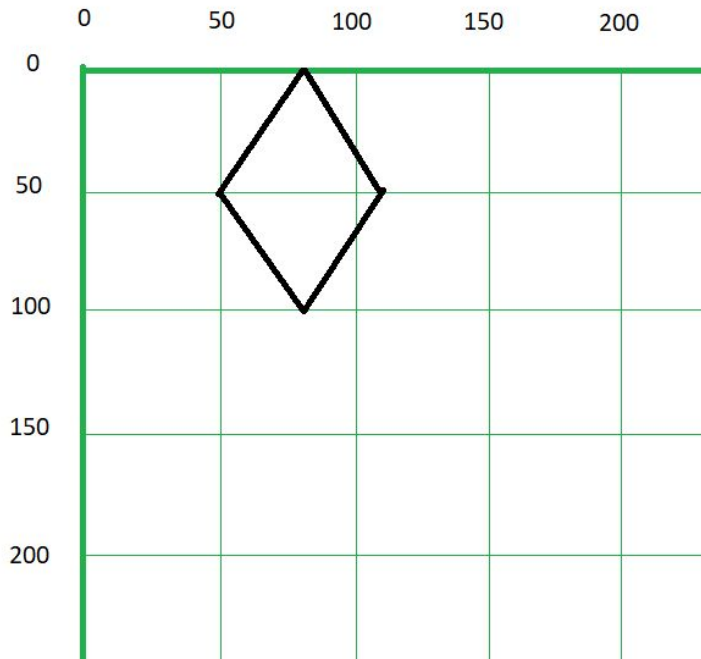
```
1 #include <stdio.h>  
2  
3 void a();  
4  
5 void main() {  
6     printf("Hello! It is main()!\n");  
7  
8     void a() {  
9         printf("Hello! It is a()!\n");  
10    }  
11  
12    a();  
13 }  
14
```

The IDE's error list window is open, showing the following errors:

Код	Описание
E0065	требуется точка с запятой ";"
C4326	возвращаемый тип "main" должен быть "int", а не "void"
C2601	а: недопустимые локальные определения функций

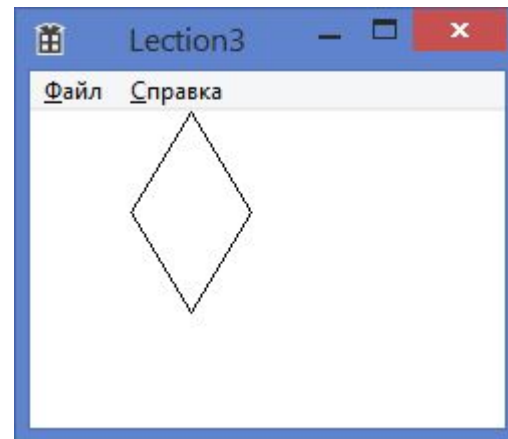
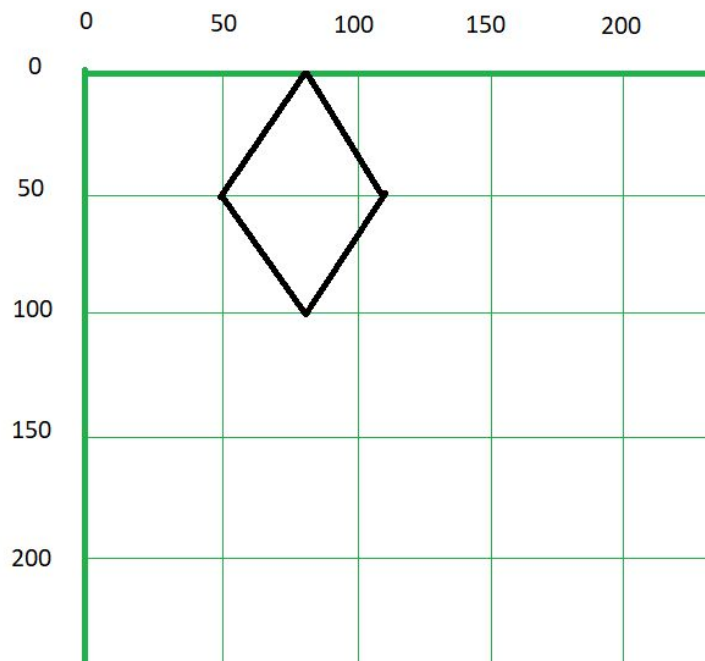
Используем функции в графике

Нарисуем ромб



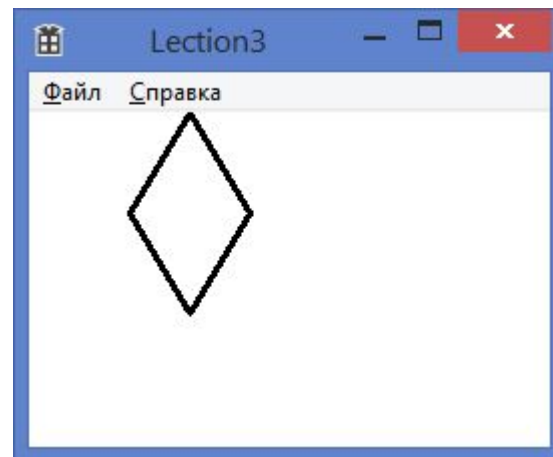
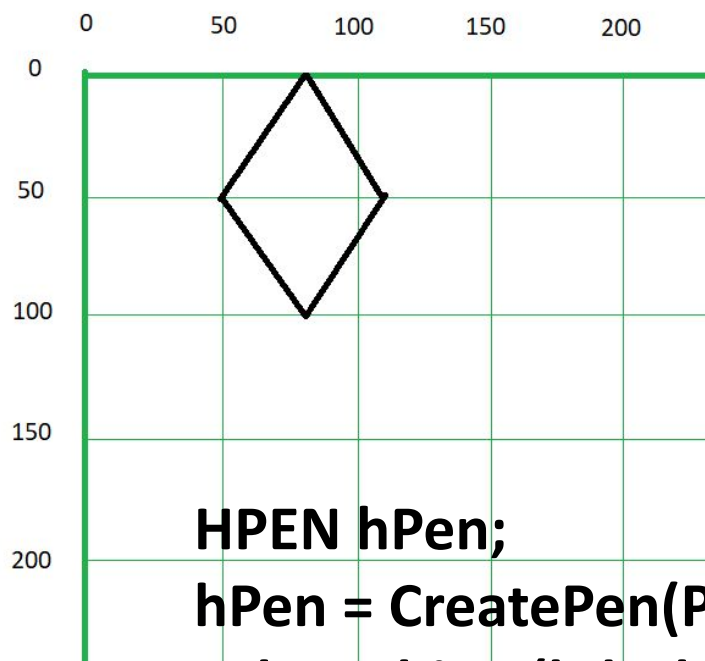
```
// Рисуем ромб  
MoveToEx(hdc, 80, 0, NULL);  
LineTo(hdc, 50, 50);  
LineTo(hdc, 80, 100);  
LineTo(hdc, 110, 50);  
LineTo(hdc, 80, 0);
```

Ромб



```
// Рисуем ромб  
MoveToEx(hdc, 80, 0, NULL);  
LineTo(hdc, 50, 50);  
LineTo(hdc, 80, 100);  
LineTo(hdc, 110, 50);  
LineTo(hdc, 80, 0);
```

Ромб



```
HPEN hPen;
```

```
hPen = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));
```

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

```
// Рисуем ромб
```

```
MoveToEx(hdc, 80, 0, NULL);
```

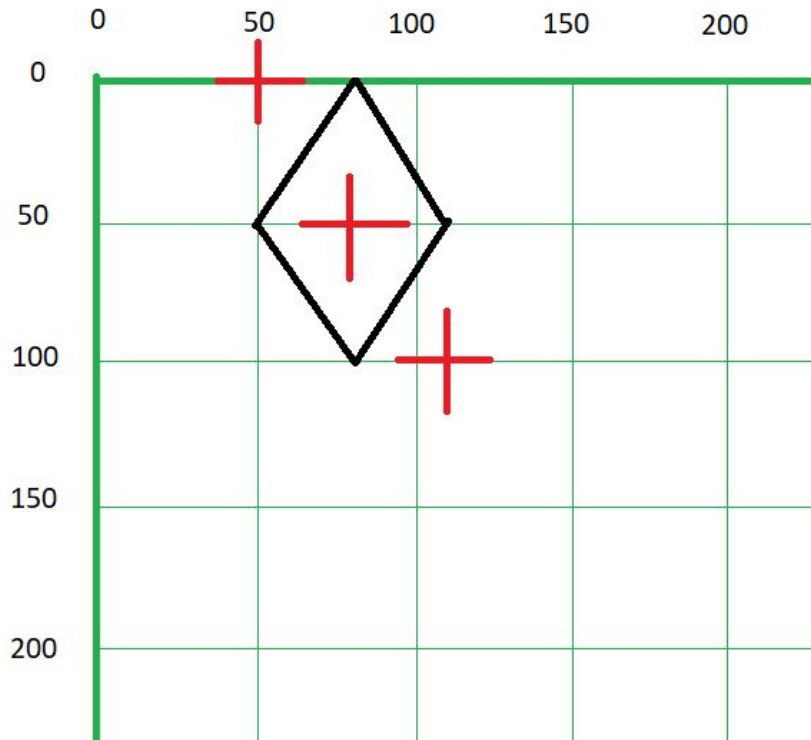
```
LineTo(hdc, 50, 50);
```

```
LineTo(hdc, 80, 100);
```

```
LineTo(hdc, 110, 50);
```

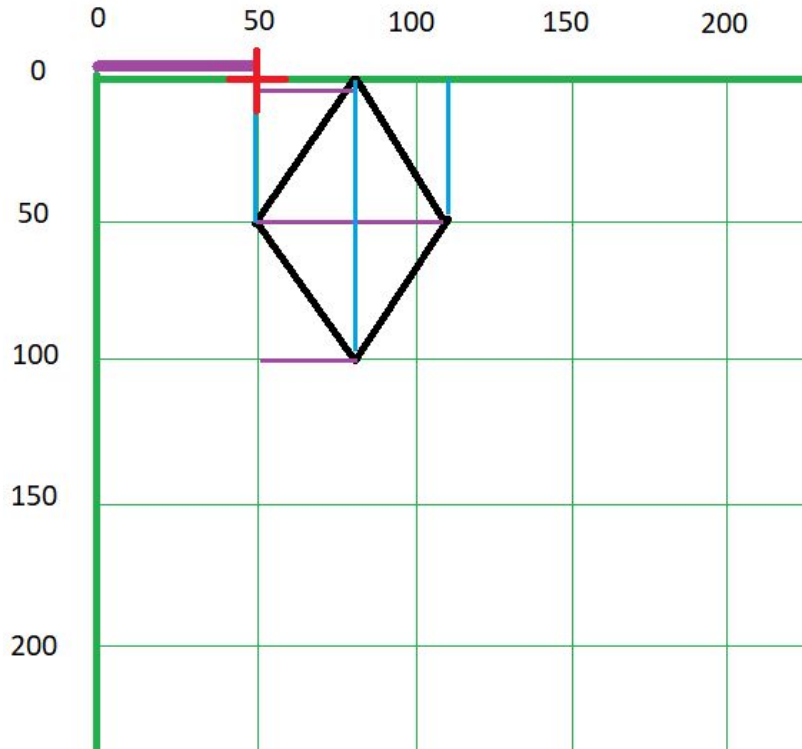
```
LineTo(hdc, 80, 0);
```

Относительные координаты



```
int x = 50;  
int y = 0;  
MoveToEx(hdc, x + 30, y, NULL);  
LineTo(hdc, x, y + 50);  
LineTo(hdc, x + 30, y + 100);  
LineTo(hdc, x + 60, y + 50);  
LineTo(hdc, x + 30, y);
```

Относительные координаты



```
int x = 50;
```

```
int y = 0;
```

```
MoveToEx(hdc, x + 30, y,  
NULL);
```

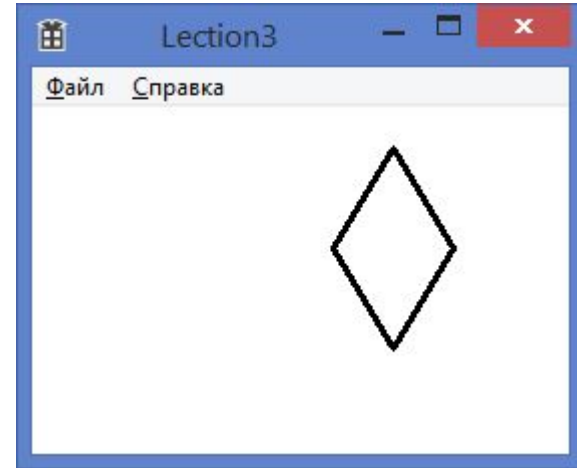
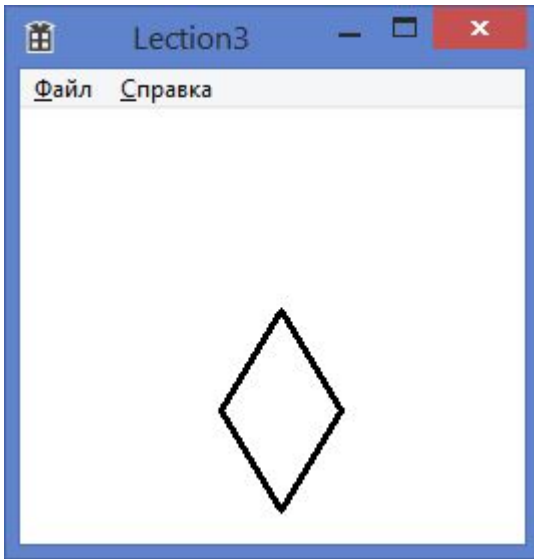
```
LineTo(hdc, x, y + 50);
```

```
LineTo(hdc, x + 30, y + 100);
```

```
LineTo(hdc, x + 60, y + 50);
```

```
LineTo(hdc, x + 30, y);
```

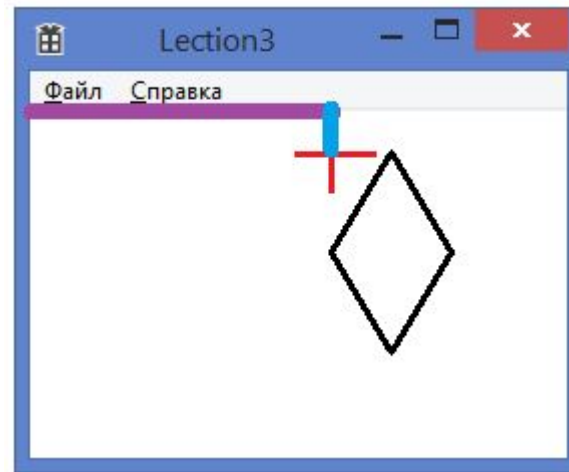
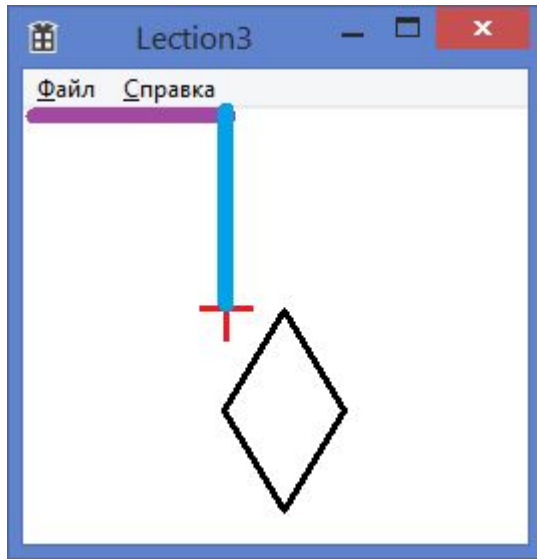

Относительные координаты



```
int x = 100;  
int y = 100;  
MoveToEx(hdc, x + 30, y, NULL);  
LineTo(hdc, x, y + 50);  
LineTo(hdc, x + 30, y + 100);  
LineTo(hdc, x + 60, y + 50);  
LineTo(hdc, x + 30, y);
```

```
int x = 150;  
int y = 20;  
MoveToEx(hdc, x + 30, y, NULL);  
LineTo(hdc, x, y + 50);  
LineTo(hdc, x + 30, y + 100);  
LineTo(hdc, x + 60, y + 50);  
LineTo(hdc, x + 30, y);
```

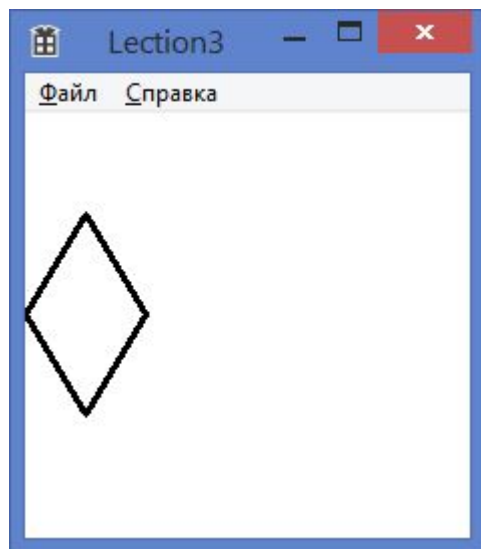
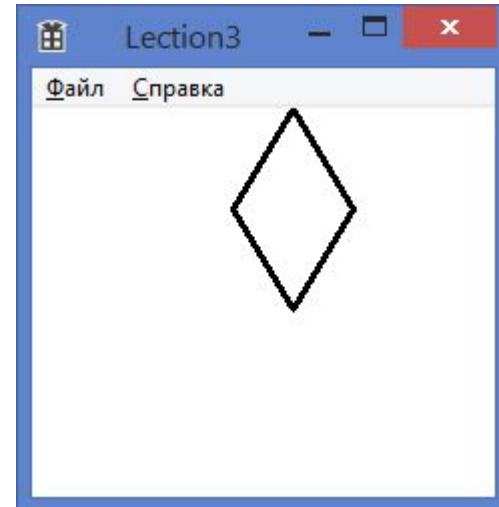
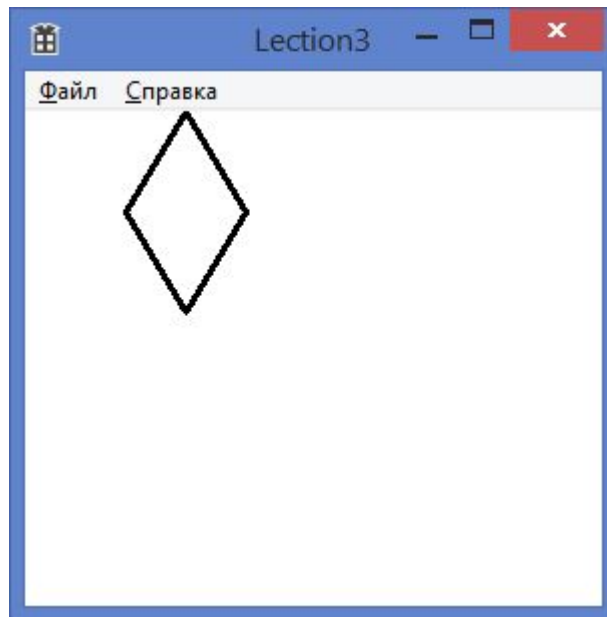
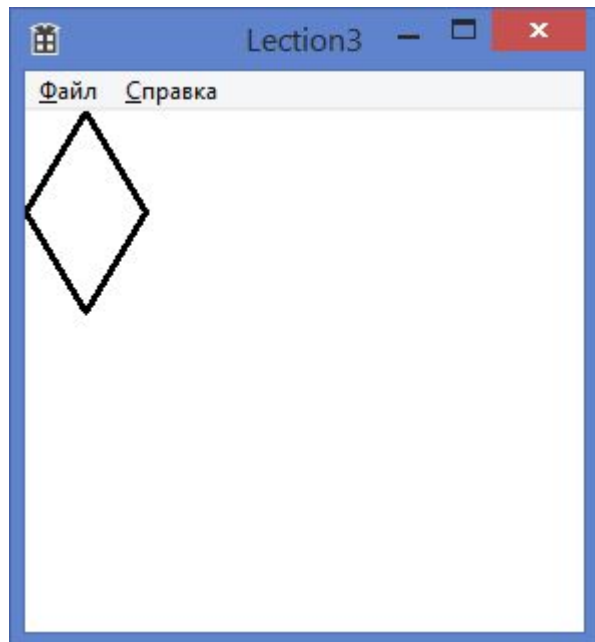
Относительные координаты



```
int x = 100;  
int y = 100;  
MoveToEx(hdc, x + 30, y, NULL);  
LineTo(hdc, x, y + 50);  
LineTo(hdc, x + 30, y + 100);  
LineTo(hdc, x + 60, y + 50);  
LineTo(hdc, x + 30, y);
```

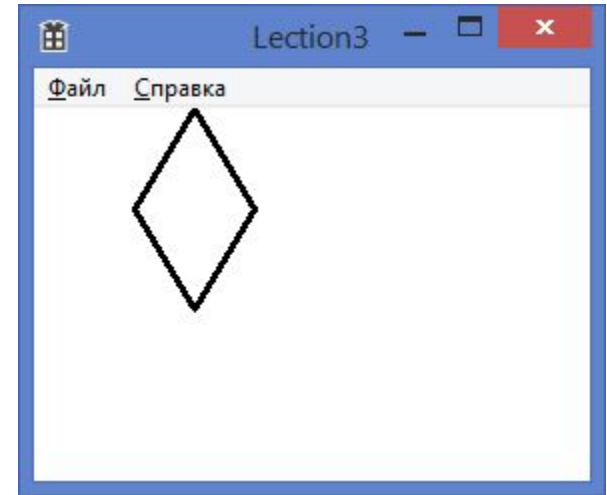
```
int x = 150;  
int y = 20;  
MoveToEx(hdc, x + 30, y, NULL);  
LineTo(hdc, x, y + 50);  
LineTo(hdc, x + 30, y + 100);  
LineTo(hdc, x + 60, y + 50);  
LineTo(hdc, x + 30, y);
```

Относительные координаты



Отдельная функция для отрисовки ромба с заданным положением

```
void Romb(HDC hdc, int x, int y) {  
    MoveToEx(hdc, x + 30, y, NULL);  
    LineTo(hdc, x, y + 50);  
    LineTo(hdc, x + 30, y + 100);  
    LineTo(hdc, x + 60, y + 50);  
    LineTo(hdc, x + 30, y);  
}
```



...

```
HDC hdc = BeginPaint(hWnd, &ps);  
HPEN hPen;  
hPen = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));  
SelectObject(hdc, hPen);  
Romb(hdc, 50, 0);
```

...

Рисуем при помощи нашей функции несколько ромбов в ряд

...

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

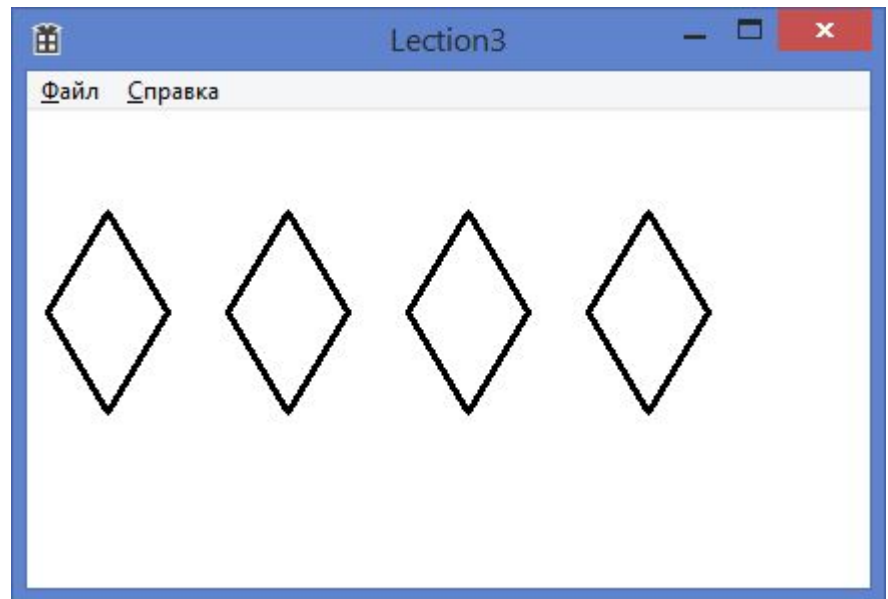
```
Romb(hdc, 10, 50);
```

```
Romb(hdc, 100, 50);
```

```
Romb(hdc, 190, 50);
```

```
Romb(hdc, 280, 50);
```

...



Рисуем при помощи нашей функции несколько ромбов – используем цикл

...

```
HPEN hPen;
```

```
hPen = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));
```

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

```
int x = 10;
```

```
int y = 50;
```

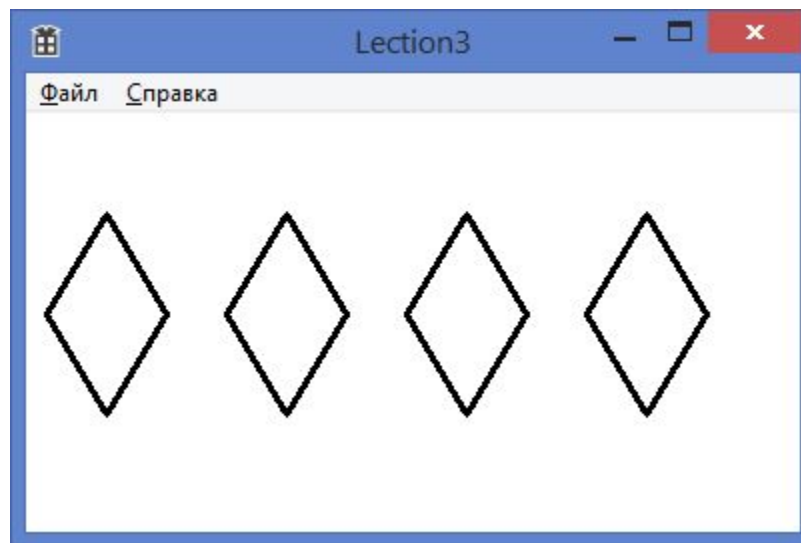
```
do {
```

```
    Romb(hdc, x, y);
```

```
    x += 90;
```

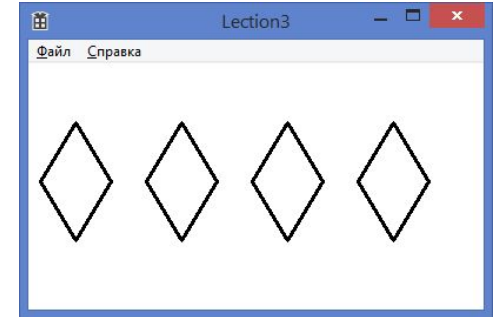
```
} while (x <= 280);
```

...



Создаем функцию, вызывающую нашу функцию

```
void drawRombLine(HDC hdc) {  
  
    HPEN hPen;  
    hPen = CreatePen(PS_SOLID, 3, RGB(0, 0, 0));  
    SelectObject(hdc, hPen);  
  
    int x = 10;  
    int y = 50;  
    do {  
        Romb(hdc, x, y);  
        x += 90;  
    } while (x <= 280);  
}
```



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


Лабораторная работа №8

Упаковываем в функции ранее
написанный код

Задача 1. Переделать задачу из ЛР1

Оформить информацию о себе в виде функции aboutMe.

Из main() вызвать эту функцию.

Нарисовать блок-схему

```
Source.cpp -> X
Lab8 (Глобаль
1  #include <stdio.h>
2  #include <Windows.h>
3
4  void aboutMe() {
5      printf("Петров\n");
6      printf("Петр\n");
7      printf("Петрович\n");
8      printf("\n");
9      printf("Группа: ИСдо-18\n");
10     printf("Дата: 14.02.2022\n");
11     printf("Предмет: Основы программирования\n");
12     printf("Лабораторная работа №8\n");
13 }
14
15 void main() {
16
17     SetConsoleCP(1251);
18     SetConsoleOutputCP(1251);
19
20     aboutMe();
21 }
22
```

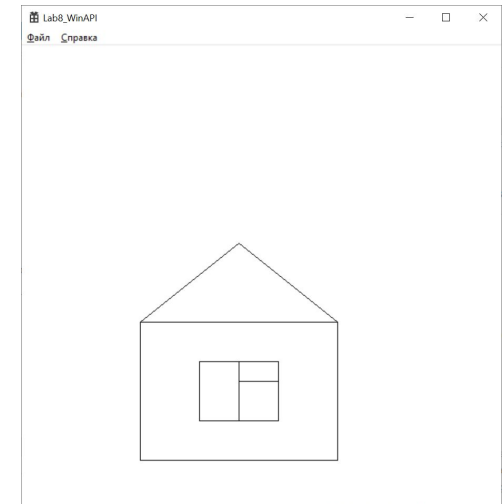
Задача 2. Переделать задачу из ЛР4

Код отрисовки дома перенести в функцию drawHome()

Из функции WndProc организовать вызов drawHome() при необходимости обновления изображения – обработка события WM_PAINT:

```
114
115 // Рисование дома из ЛР4
116 void drawHome(HDC hdc) {
117     // крыша
118     MoveToEx(hdc, 150, 350, NULL);
119     LineTo(hdc, 275, 250);
120     LineTo(hdc, 400, 350);
121
122     // Дом
123     LineTo(hdc, 400, 525);
124     LineTo(hdc, 150, 525);
125     LineTo(hdc, 150, 350);
126     LineTo(hdc, 400, 350);
127
128     // окно
129     MoveToEx(hdc, 225, 400, NULL);
130     LineTo(hdc, 225, 475);
131     LineTo(hdc, 325, 475);
132     LineTo(hdc, 325, 400);
133     LineTo(hdc, 225, 400);
134
135     // Рама
136     MoveToEx(hdc, 275, 400, NULL);
137     LineTo(hdc, 275, 475);
138     MoveToEx(hdc, 275, 425, NULL);
139     LineTo(hdc, 325, 425);
140 }
141
```

```
241
242
243 case WM_PAINT:
244     {
245         PAINTSTRUCT ps;
246         HDC hdc = BeginPaint(hWnd, &ps);
247         // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
248
249         // Рисование дома из ЛР4
250         drawHome(hdc);
251         EndPaint(hWnd, &ps);
252     }
253     break;
```

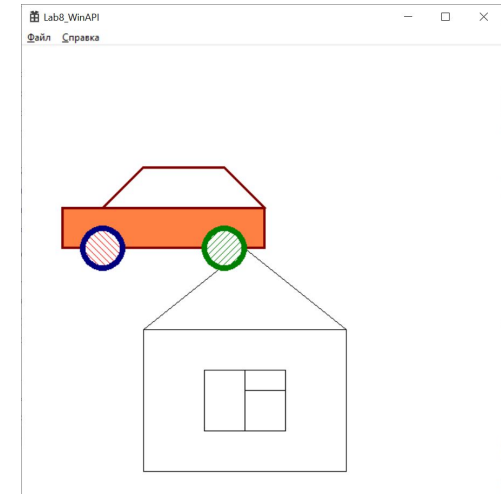


Задача 3. Переделать задачу из ЛР5

Код отрисовки автомобиля перенести в функцию drawCar()
Из функции WndProc организовать вызов drawCar() при необходимости обновления изображения – обработка события WM_PAINT:

```
142 // Рисование автомобиля из ЛР5
143
144 void drawCar(HDC hdc) {
145
146     HPEN hPen = CreatePen(PS_SOLID, 3, RGB(128, 0, 0));
147     SelectObject(hdc, hPen);
148
149     HBRUSH hBrush;
150     hBrush = CreateSolidBrush(RGB(255, 128, 67));
151     SelectObject(hdc, hBrush);
152
153     Rectangle(hdc, 50, 200, 300, 250);
154
155     MoveToEx(hdc, 100, 200, NULL);
156     LineTo(hdc, 150, 150);
157     LineTo(hdc, 250, 150);
158     LineTo(hdc, 300, 200);
159
160
161     hPen = CreatePen(PS_SOLID, 7, RGB(0, 0, 128));
162     SelectObject(hdc, hPen);
163
164     hBrush = CreateHatchBrush(HS_FDIAGONAL, RGB(255, 0, 0));
165     SelectObject(hdc, hBrush);
166
167     Ellipse(hdc, 75, 225, 125, 275);
168
169
170
171     hPen = CreatePen(PS_SOLID, 7, RGB(0, 128, 0));
172     SelectObject(hdc, hPen);
173
174     hBrush = CreateHatchBrush(HS_BDIAGONAL, RGB(0, 128, 0));
175     SelectObject(hdc, hBrush);
176
177     Ellipse(hdc, 225, 225, 275, 275);
178 }
179
```

```
241
242
243 case WM_PAINT:
244 {
245     PAINTSTRUCT ps;
246     HDC hdc = BeginPaint(hWnd, &ps);
247     // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
248
249     // Рисование дома из ЛР4
250     drawHome(hdc);
251     // Рисование автомобиля из ЛР5
252     drawCar(hdc);
253
254     EndPaint(hWnd, &ps);
255 }
256 break;
```



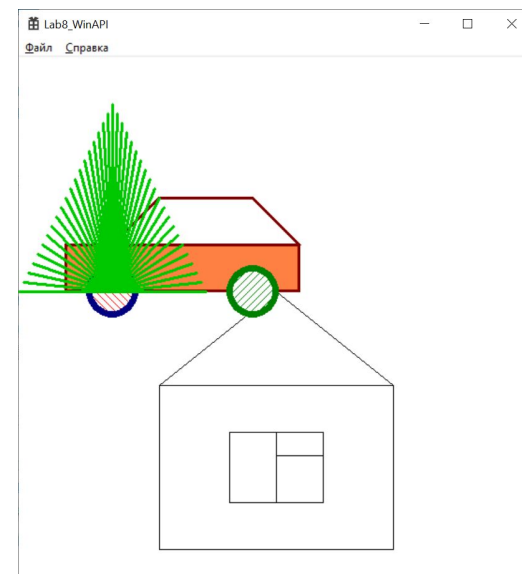
Задача 4*. Переделать задачу из ЛР7

Код отрисовки автомобиля перенести в функцию drawTree()

Из функции WndProc организовать вызов drawTree() при необходимости обновления изображения – обработка события WM_PAINT:

```
179
180 // Рисование дерева из ЛР7
181 void drawTree(HDC hdc) {
182
183     HPEN hPen = CreatePen(PS_SOLID, 3, RGB(0, 200, 0));
184     SelectObject(hdc, hPen);
185
186     int x1 = 0, y1 = 250;
187     int x2 = 100, y2 = 250;
188
189     do {
190         MoveToEx(hdc, x1, y1, NULL);
191         LineTo(hdc, x2, y2);
192         x1 += 5;
193         y1 -= 10;
194     } while (x1 <= 100);
195
196     x1 = 100;
197     y1 = 50;
198     do {
199         MoveToEx(hdc, x1, y1, NULL);
200         LineTo(hdc, x2, y2);
201         x1 += 5;
202         y1 += 10;
203     } while (x1 <= 200);
204
205 }
206
207
```

```
240
241 case WM_PAINT:
242     {
243         PAINTSTRUCT ps;
244         HDC hdc = BeginPaint(hWnd, &ps);
245         // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
246
247         // Рисование дома из ЛР4
248         drawHome(hdc);
249         // Рисование автомобиля из ЛР5
250         drawCar(hdc);
251         // Рисование дерева из ЛР7
252         drawTree(hdc);
253
254         EndPaint(hWnd, &ps);
255     }
256     break;
```



Домашнее задание по ЛР8

- 1) Доделать задачи 1-3.
- 2) Все задания по отрисовке рисунков из лабораторных работ 4, 5, 7 ранее сделанные вами, нужно оформить в виде функций. Для каждого изображения (мост, дом, машина, снеговик и т.п.) нужно создать отдельную функцию. Из имени функции должно быть понятно, что именно эта функция отрисовывает. Созданные функции вызываются из функции `WndProc` при необходимости обновления изображения – при обработке события `WM_PAINT`, как реализовано в задачах этой лабораторной работы.
- 3) Обязательно! Принести получившийся код на занятие. Его будем использовать и переделывать на следующих лабораторных работах.

Лабораторная работа №9

функций для рисования объектов по
координатам

Автомобиль деда Мороза - логотип

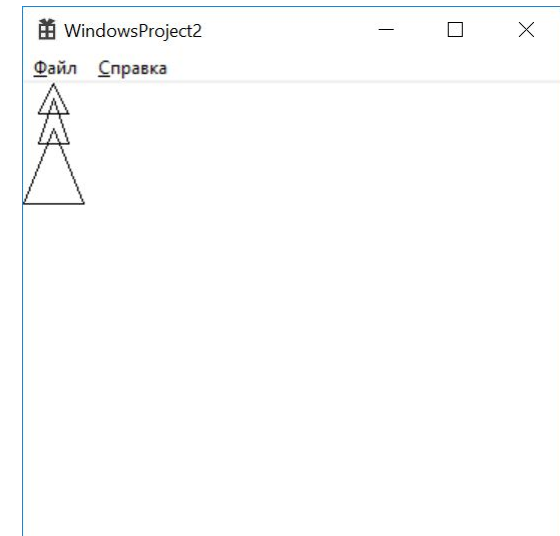
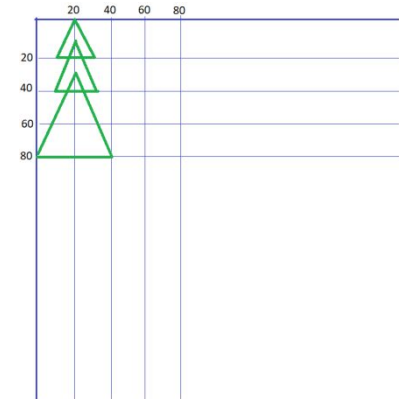
```
case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hWnd, &ps);
    // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

    // верхний треугольник
    MoveToEx(hdc, 20, 0, NULL);
    LineTo(hdc, 30, 20);
    LineTo(hdc, 10, 20);
    LineTo(hdc, 20, 0);

    // средний треугольник
    MoveToEx(hdc, 20, 10, NULL);
    LineTo(hdc, 30, 40);
    LineTo(hdc, 10, 40);
    LineTo(hdc, 20, 10);

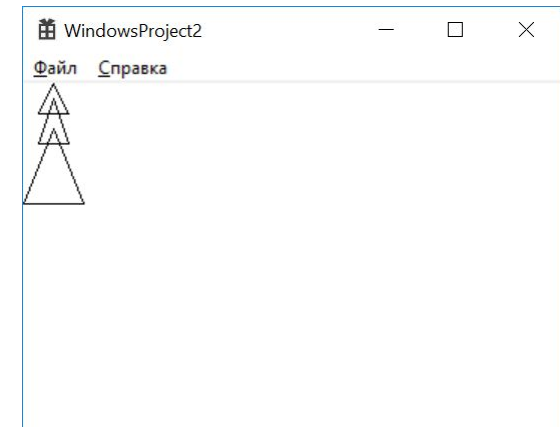
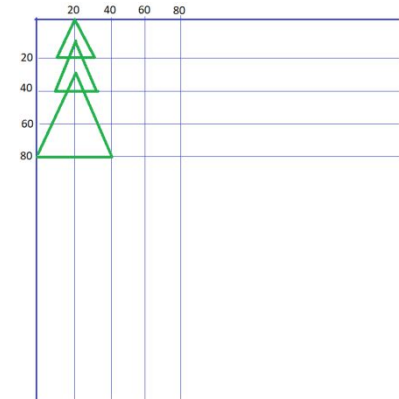
    // нижний треугольник
    MoveToEx(hdc, 20, 30, NULL);
    LineTo(hdc, 40, 80);
    LineTo(hdc, 0, 80);
    LineTo(hdc, 20, 30);

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



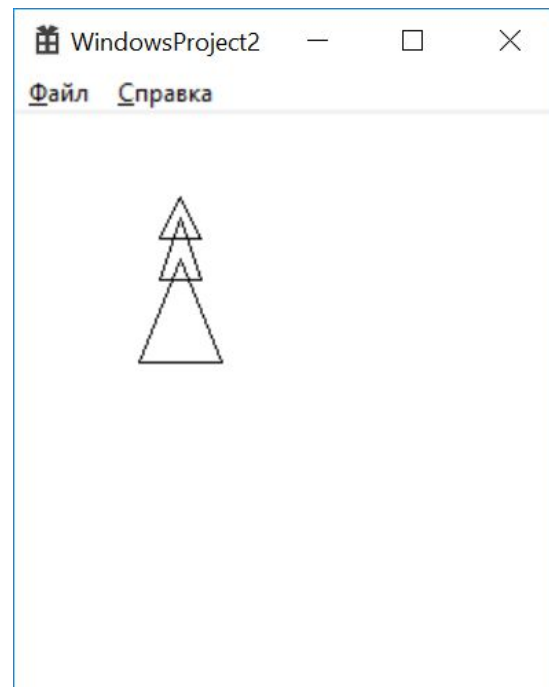
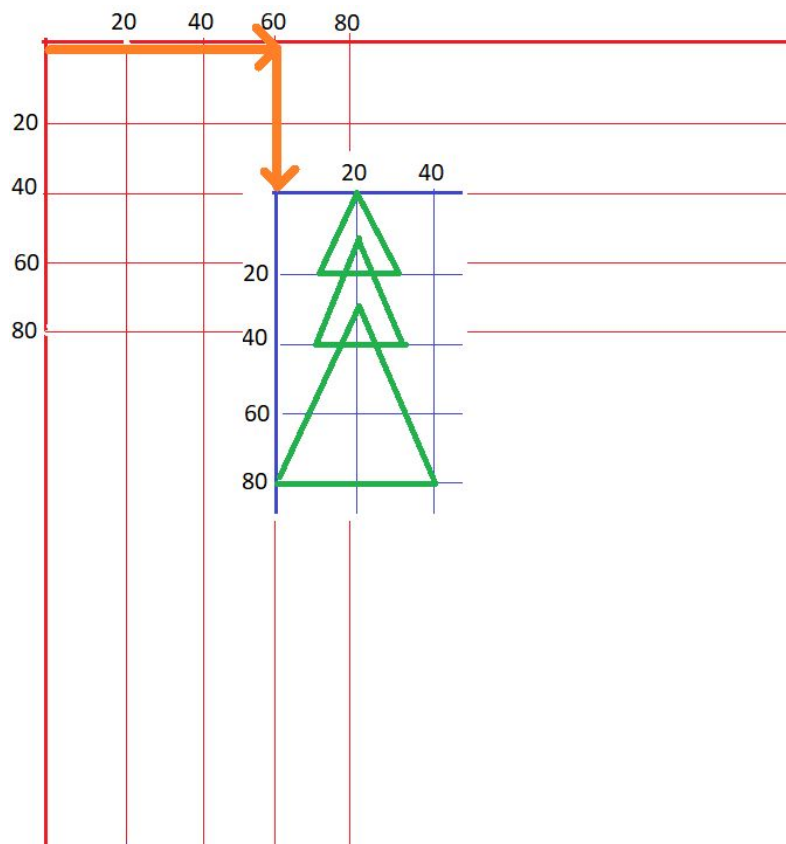
Вынесли код в отдельную функцию

```
void StClausAuto(HDC hdc) {  
    // верхний треугольник  
    MoveToEx(hdc, 20, 0, NULL);  
    LineTo(hdc, 30, 20);  
    LineTo(hdc, 10, 20);  
    LineTo(hdc, 20, 0);  
  
    // средний треугольник  
    MoveToEx(hdc, 20, 10, NULL);  
    LineTo(hdc, 30, 40);  
    LineTo(hdc, 10, 40);  
    LineTo(hdc, 20, 10);  
  
    // нижний треугольник  
    MoveToEx(hdc, 20, 30, NULL);  
    LineTo(hdc, 40, 80);  
    LineTo(hdc, 0, 80);  
    LineTo(hdc, 20, 30);  
}  
...
```



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

Относительные координаты

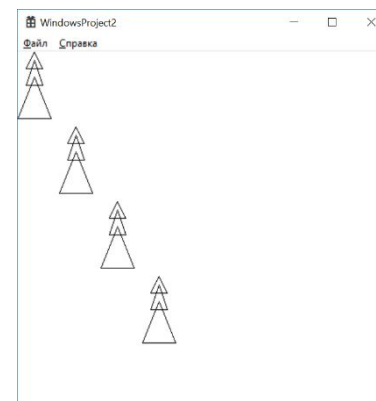
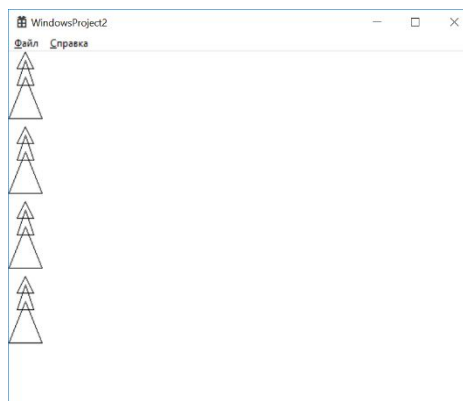
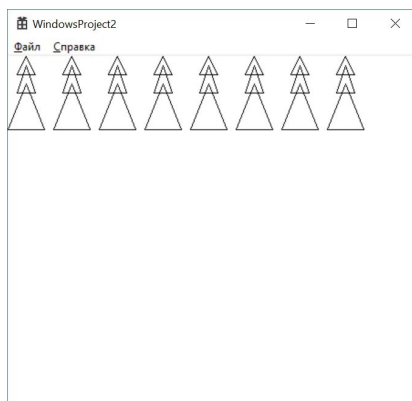
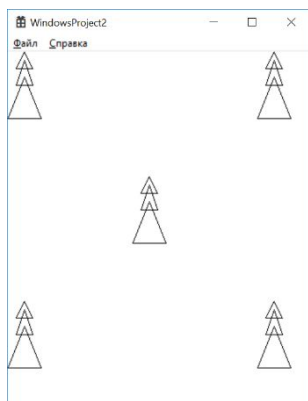


Задача 1 Елочка (Логотип авто Деда Мороза) в виде функции с параметрами x , y

Сделать функцию

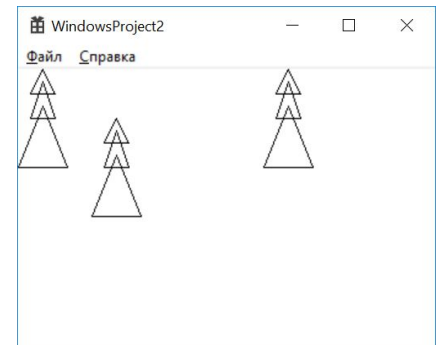
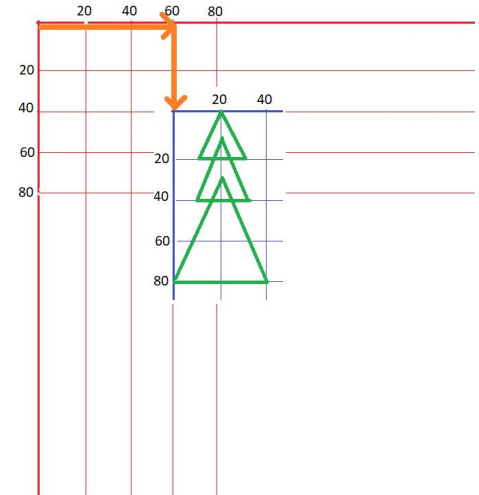
```
void StClausAuto(HDC hdc, int x, int y) { ... }
```

и используя её нарисовать из елочек узоры по следующим схемам



Относительные координаты

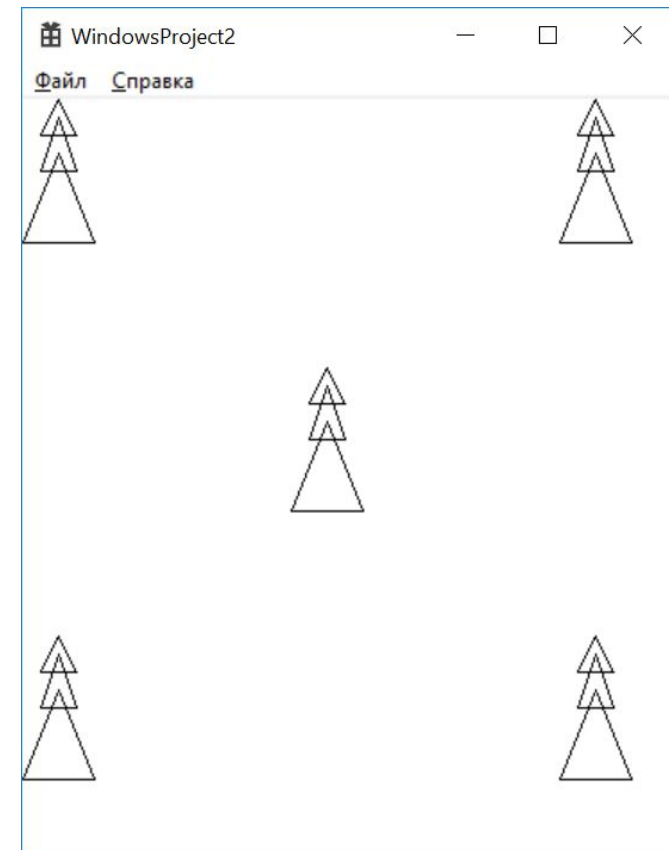
```
void StClausAuto(HDC hdc, int x, int y) {  
    // верхний треугольник  
    MoveToEx(hdc, 20 + x, 0 + y, NULL);  
    LineTo(hdc, 30 + x, 20 + y);  
    LineTo(hdc, 10 + x, 20 + y);  
    LineTo(hdc, 20 + x, 0 + y);  
  
    // средний треугольник  
    MoveToEx(hdc, 20 + x, 10 + y, NULL);  
    LineTo(hdc, 30 + x, 40 + y);  
    LineTo(hdc, 10 + x, 40 + y);  
    LineTo(hdc, 20 + x, 10 + y);  
  
    // нижний треугольник  
    MoveToEx(hdc, 20 + x, 30 + y, NULL);  
    LineTo(hdc, 40 + x, 80 + y);  
    LineTo(hdc, 0 + x, 80 + y);  
    LineTo(hdc, 20 + x, 30 + y);  
}  
...
```



```
case WM_PAINT:  
{  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);  
    // TODO: Добавьте сюда любой код прорисовки, использующий HDC...  
  
    StClausAuto(hdc, 60, 40);  
    StClausAuto(hdc, 0, 0);  
    StClausAuto(hdc, 200, 0);  
  
    EndPaint(hWnd, &ps);  
}  
break;
```

Задача 1.1: 5 логотипов по углам и в центре

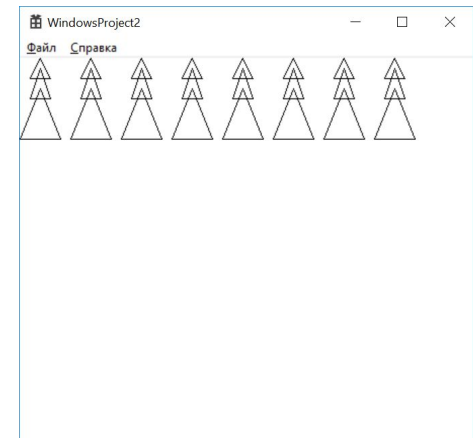
```
247 case WM_PAINT:
248 {
249     PAINTSTRUCT ps;
250     HDC hdc = BeginPaint(hWnd, &ps);
251     // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
252
253     StClausAuto(hdc, 0, 0);
254     StClausAuto(hdc, 200, 200);
255     StClausAuto(hdc, 100, 100);
256     StClausAuto(hdc, 0, 200);
257     StClausAuto(hdc, 200, 0);
258
259     EndPaint(hWnd, &ps);
260 }
261 break;
```



Задача 1.2: 8 логотипов в горизонтальную линию

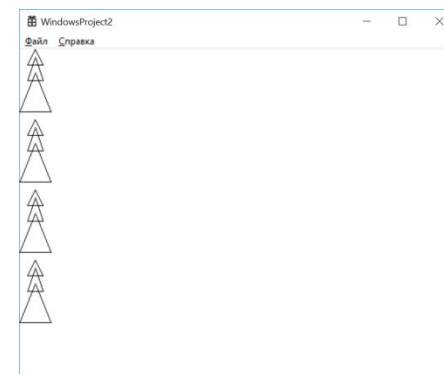
Решение

```
case WM_PAINT:  
{  
    PAINTSTRUCT ps;  
    HDC hdc = BeginPaint(hWnd, &ps);  
    // TODO: Добавьте сюда любой код прорисовки, использующий HDC...  
  
    int x = 0;  
    do {  
        StClausAuto(hdc, x, 0);  
        x += 50;  
    } while (x < 400);  
  
    EndPaint(hWnd, &ps);  
}  
break;
```

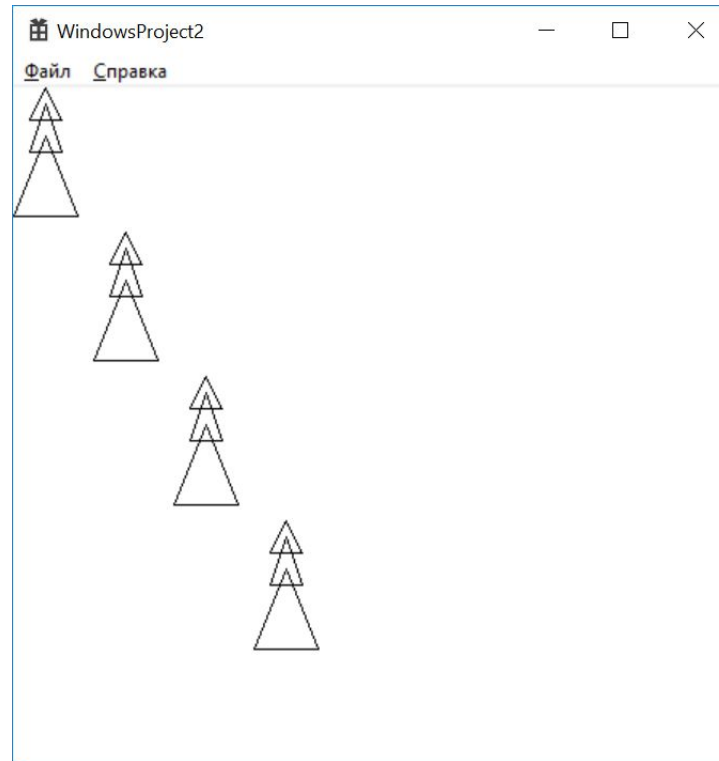


Задача 1.3 - РЕШЕНИЕ: 4 логотипа в вертикальную линию

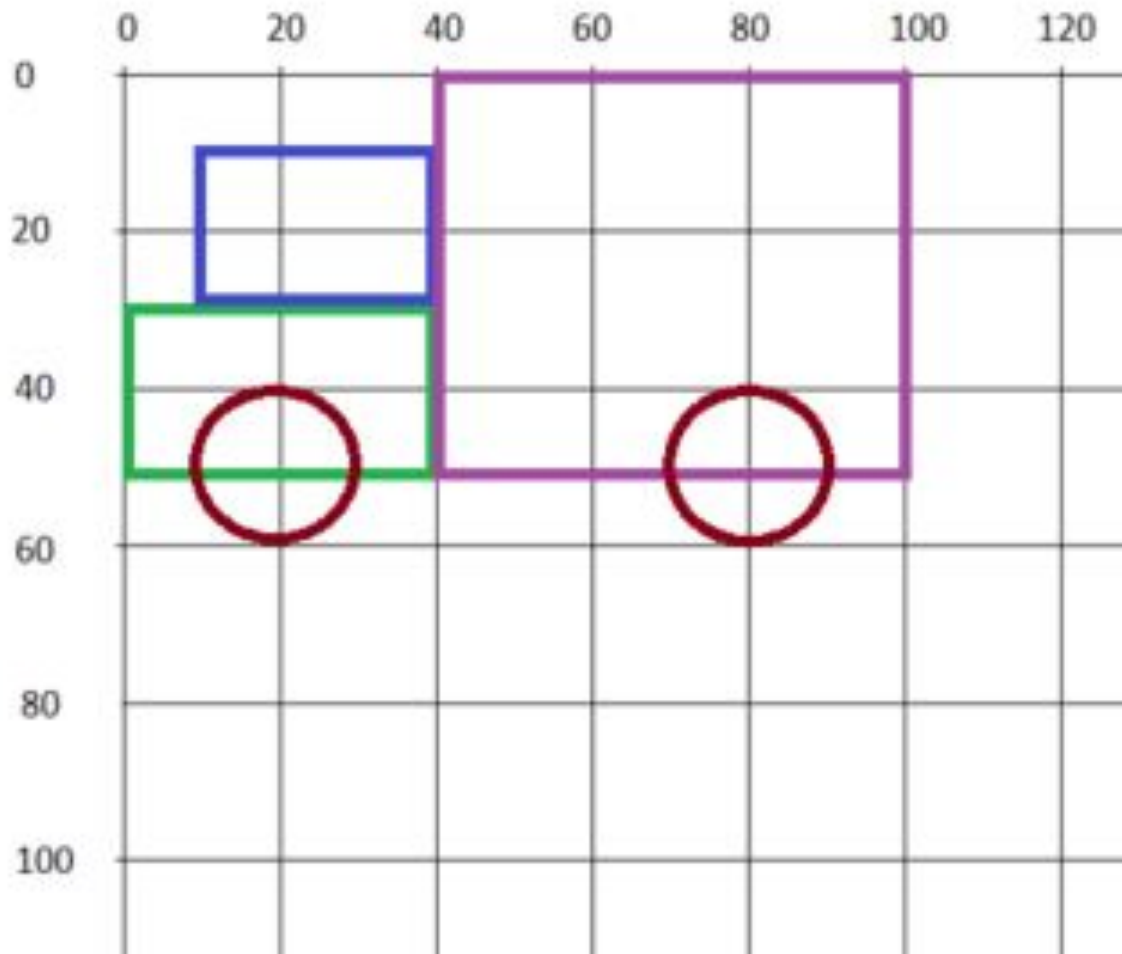
```
int y = 0;  
do {  
    StClausAuto(hdc, 0, y);  
    y += 90;  
} while (y < 300);
```



Задача 1.4: 4 логотипа в диагональную ЛИНИЮ



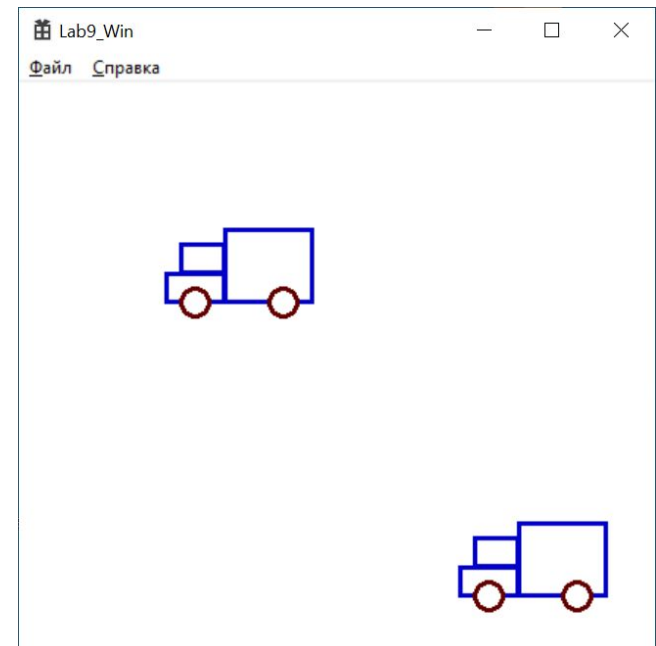
Грузовой автомобиль - расчеты



Задача 2.1 Создать функцию drawTruck для рисования грузового автомобиля

Сделать функцию drawTruck(HDC hdc, int x, int y) { ... } и используя ее нарисовать несколько грузовиков

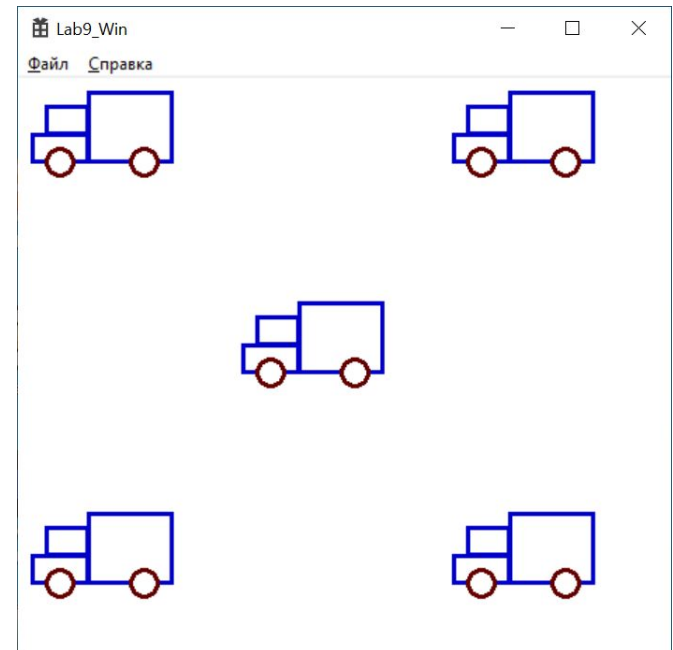
```
117
118 // Рисование грузовика
119 void drawTruck(HDC hdc, int x, int y) {
120     HPEN hPen = CreatePen(PS_SOLID, 3, RGB(0, 0, 200));
121     SelectObject(hdc, hPen);
122
123     Rectangle(hdc, 0 + x, 30 + y, 40 + x, 50 + y);
124     Rectangle(hdc, 10 + x, 10 + y, 40 + x, 30 + y);
125     Rectangle(hdc, 40 + x, 0 + y, 100 + x, 50 + y);
126
127     hPen = CreatePen(PS_SOLID, 3, RGB(100, 0, 0));
128     SelectObject(hdc, hPen);
129     Ellipse(hdc, 10 + x, 40 + y, 30 + x, 60 + y);
130     Ellipse(hdc, 70 + x, 40 + y, 90 + x, 60 + y);
131 }
132
```



Задача 2.2: 5 автомобилей по углам и в центре

Сделать функцию drawTrucks1(HDC hdc) которая рисует грузовики по следующей схеме:

```
132
133 void drawTrucks1(HDC hdc) {
134     drawTruck(hdc, 10, 10);
135     drawTruck(hdc, 10, 310);
136     drawTruck(hdc, 310, 10);
137     drawTruck(hdc, 310, 310);
138     drawTruck(hdc, 160, 160);
139 }
140
```



```
173 case WM_PAINT:
174     {
175         PAINTSTRUCT ps;
176         HDC hdc = BeginPaint(hWnd, &ps);
177         // TODO: Добавьте сюда любой код прорисовки, использующий HDC...
178
179         drawTrucks1(hdc);
180
181         EndPaint(hWnd, &ps);
182     }
183     break;
```

Задача 2.3 – 2.5– Создать 3 рисунка из грузовиков

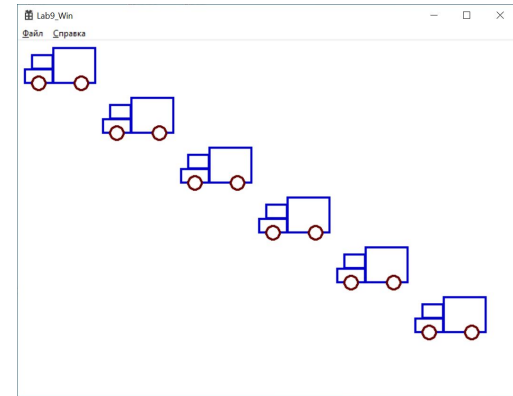
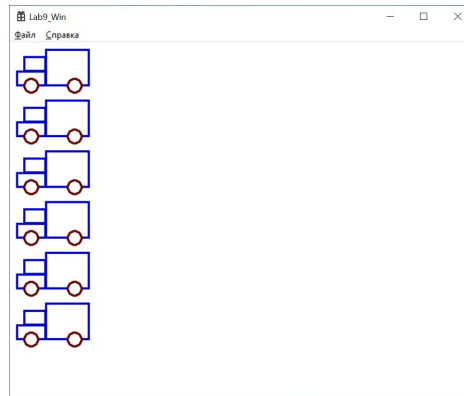
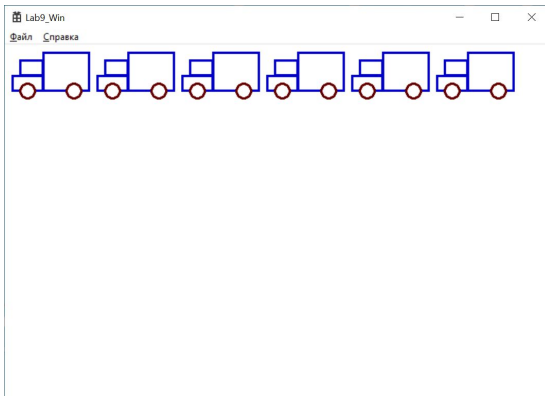
Сделать функции

`drawTrucks2(HDC hdc)`

`drawTrucks3(HDC hdc)`

`drawTrucks4(HDC hdc)`

которые создают рисунки из грузовиков по следующим схемам:

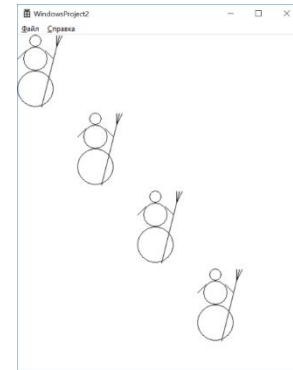
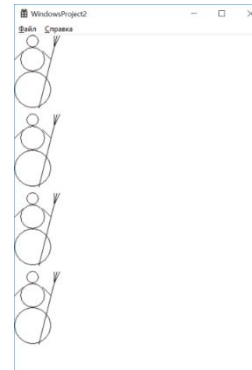
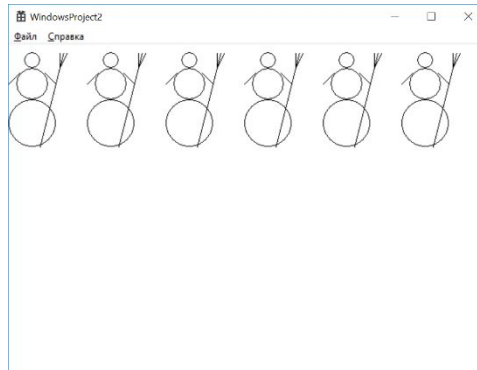
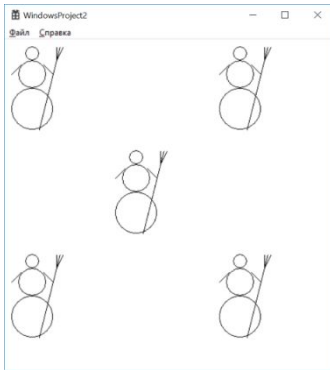


Задача 3*. Снежная баба в виде функции с параметрами x, y

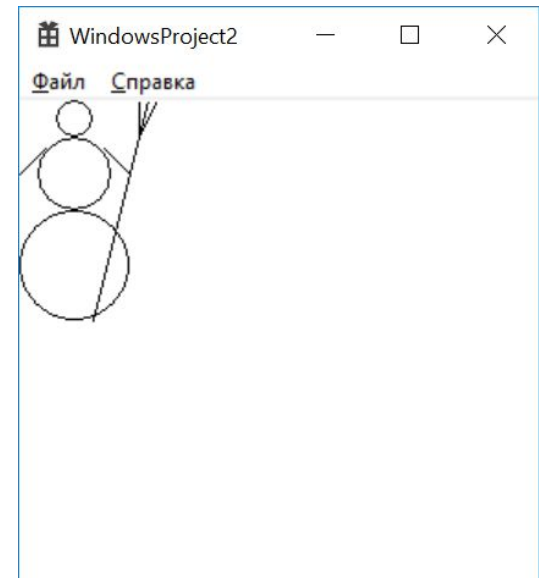
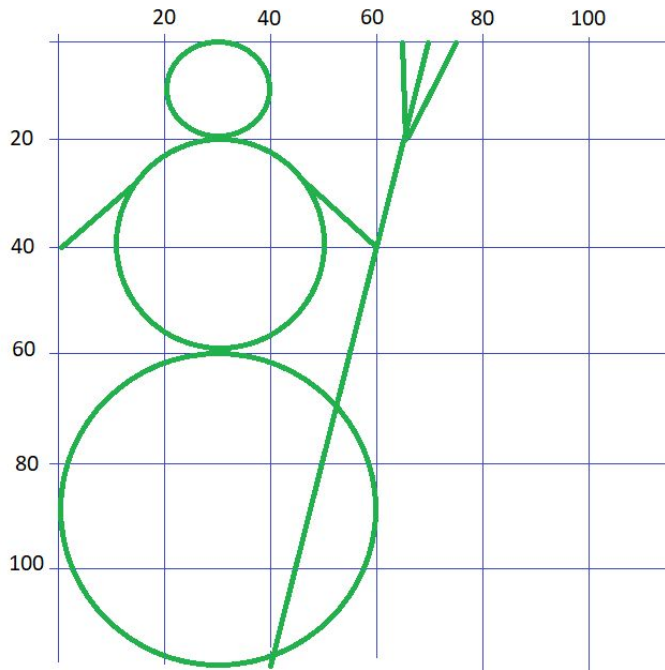
Сделать функцию

```
void SnowWoman(HDC hdc, int x, int y) { ... }
```

и используя её нарисовать узоры по следующим схемам



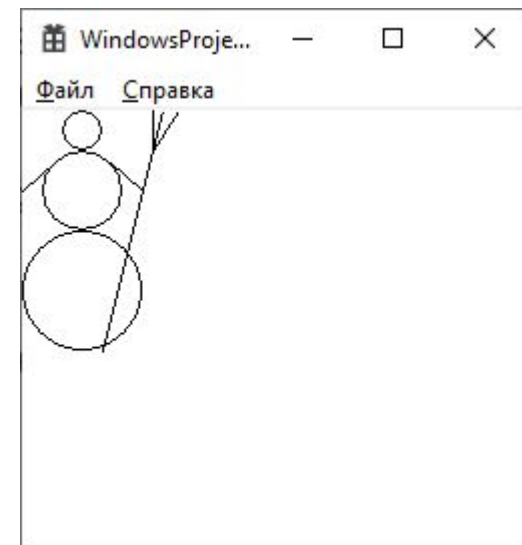
Задача 3.1* – Создать картинку по образцу



Задача 3.2* – Создать функцию SnowWoman

```
209 void SnowWoman(HDC hdc, int x, int y) {  
210     Ellipse(hdc, 20 + x, 0 + y, 40 + x, 20 + y);  
211     Ellipse(hdc, 10 + x, 20 + y, 50 + x, 60 + y);
```

```
261     case WM_PAINT:  
262     {  
263         PAINTSTRUCT ps;  
264         HDC hdc = BeginPaint(hWnd, &ps);  
265         // TODO: Добавьте сюда любой код прорисовки, использующий HDC...  
266  
267         SnowWoman(hdc, 0, 0);  
268  
269         EndPaint(hWnd, &ps);  
270     }  
271     break;
```



Задача 3.3* – 3.6* – Создать 4 рисунков из снежных баб

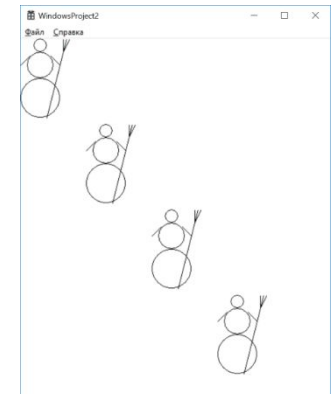
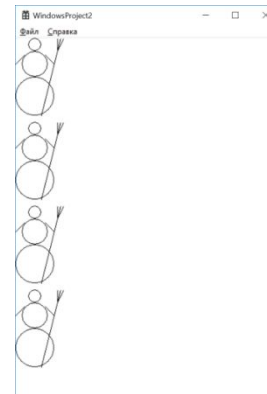
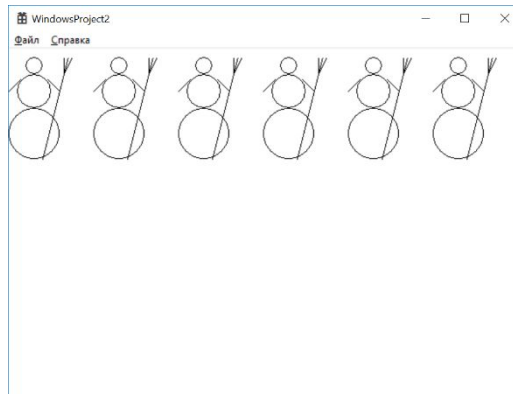
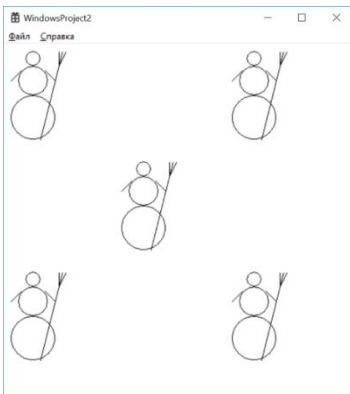
Рекомендуется для отрисовки каждого из рисунков создать отдельную функцию. Рекомендованные имена:

`drawSnowWomen1(HDC hdc)`

`drawSnowWomen2(HDC hdc)`

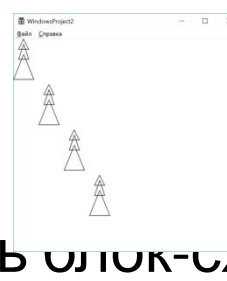
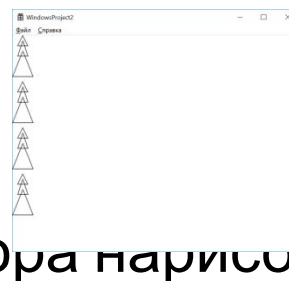
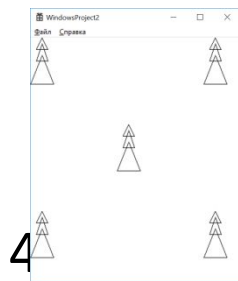
`drawSnowWomen3(HDC hdc)`

`drawSnowWomen4(HDC hdc)`



Домашнее задание ЛР9

- 1) Доделать рисунки из Задач 1 и 2, которые не успели сделать на занятии в классе.
- 2) Создать в виде отдельной функции логотип любого автомобиля. Созданная функция должна иметь вид `Logo(HDC hdc, int x, int y)`. При этом сделать логотип такого размера, чтобы он вмещался по высоте окна не меньше 4 раз, по ширине не меньше 6 раз
- 3) Используя эту функцию создать узоры из логотипов по следующим 4 схемам:



4) Последнему узору нарисовать блок-схему алгоритма.

ЛР9 – оформление

Для сдачи работы нужно иметь:

- 1) код программы (с собой)
- 2) Расчет картинка – на бумаге (на отдельном листе или в тетради) или в файле с видимыми признаками расчетов
- 4) Блоксхема для одного из рисунков, где используется циклический алгоритм

Срок выполнения – до следующей встречи на лабораторной работе

Если болел/не мог – это становится «долгом»

ИТОГО по лекции 5

1. Узнали как объявляются и определяются функции
2. Узнали как вернуть значение из функции
3. Узнали как передать значение в функцию
4. Узнали как создать функцию для отрисовки изображений
5. Узнали как создать функцию для отрисовки изображений по координатам

