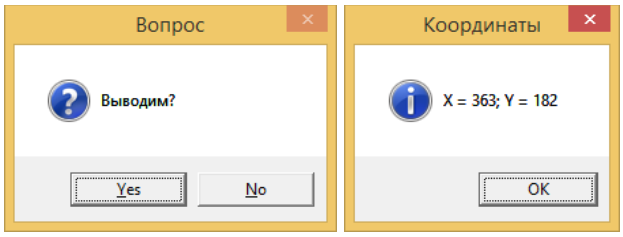
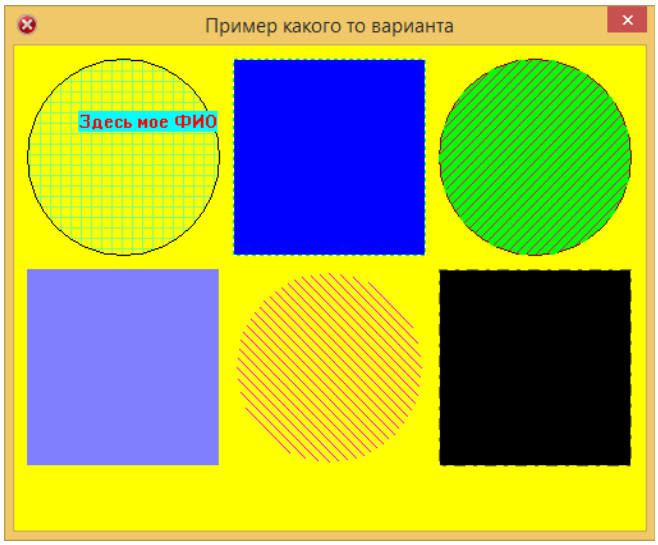
Лабораторная работа №4

1. Создал графическое приложение в CLion 2018.3.
2. Изменил класс окна на MyClassName
3. Изменил заголовок окна на «Пример какого то варианта»
4. Задал начальные координаты окна (10, 50) и размер окна 500x400
5. Изменил стиль окна так, чтобы нельзя было изменять размер мышкой и не было кнопок «свернуть» и «развернуть»
6. Сделал курсором изображение в виде пересечения вертикального и горизонтального отрезка, а иконкой белый крест на фоне красного круга

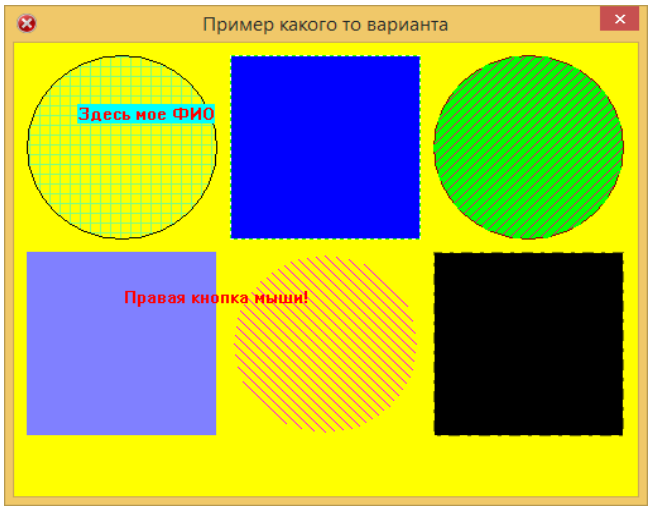
7. Изменил программу, чтобы при нажатии левой кнопки мыши выводился запрос на вывод координат, а при ответе «Да» - сами координаты.



8. Организовал вывод графических объектов при обработке сообщения WM_PAINT. Нарисовал 3 квадрата и 3 круга разными цветами и вывел текстовое сообщение в точке (50, 50).



9. Добавил обработку сообщения о нажатии правой кнопки мыши. В координатах нажатия выводится текст «Правая кнопка мыши!».



Код программы

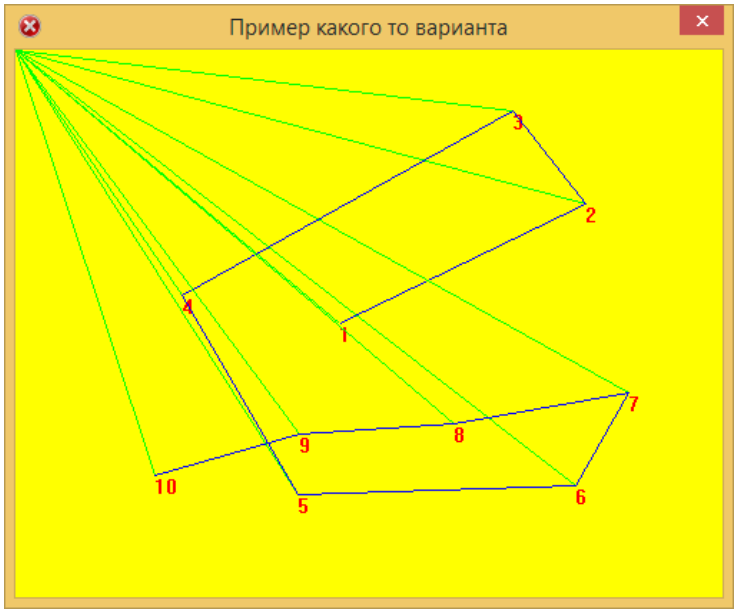
```
#include <windows.h>
#include <math.h>

/* Координаты последнего нажатия правой кнопки (-1 - нажатия не было) */
int lastRX = -1, lastRY = -1;

/**
 * Оконная функция
 * @param hWnd Окно
 * @param message сообщение
 * @param wParam w-параметр
 * @param lParam l-параметр
 * @return результат выполнения
 */
LRESULT CALLBACK WindowFunc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
    switch (message) {
        case WM_DESTROY: // уничтожение окна
            // отправляю сообщение WM_QUIT
            PostQuitMessage(0);
            break;

        case WM_LBUTTONDOWN: // левая кнпока мыши
            // Если пользователь отвечает "да"
            if (MessageBox(hWnd, "Выводим?", "Вопрос", MB_ICONQUESTION | MB_YESNO) ==
IDYES) {
                // Декодирую координаты
                WORD x = LOWORD(lParam);
                WORD y = HIWORD(lParam);
                // буфер достаточной длины для строки вывода координат
                char mem[400];
                // форматирую сообщение
                wsprintf(mem, "X = %d; Y = %d", x, y);
                // вывожу сообщение
                MessageBox(hWnd, mem, "Координаты", MB_ICONINFORMATION | MB_OK);
            }
            break;

        case WM_RBUTTONDOWN: // правая кнопка мыши
            // Декодирую координаты и сохраняю их глобально
            lastRX = LOWORD(lParam);
            lastRY = HIWORD(lParam);
            // Окно нуждается в перерисовке
            InvalidateRect(hWnd, NULL, true);
            break;

        case WM_PAINT: // отрисовка
```

```
        {
            // размеры перьев
            const int PEN_SIZES[] = {1, 1, 1, 3, 5, 1};
            // цвета перьев
            const COLORREF PEN_COLORS[] = {0, 0xFFFFFF, 0x000080, 0xFF00FF, 0x00FFFF,
0x008080};
            // стили перьев
            const int PEN_STYLES[] = {PS_SOLID, PS_DOT, PS_DASH, PS_NULL, PS_SOLID,
PS_DASHDOT};
            // цвета кистей
            const COLORREF BRUSH_COLOR[] = {0x80FF80, 0xFF0000, 0x8000FF, 0xFF8080,
0xFF00FF};
            // типы штрихов кистей (-1 - обычная кисть)
            const int BRUSH_HATCH[] = {HS_CROSS, -1, HS_BDIAGONAL, -1, HS_FDIAGONAL, -1};
            // цвета фона (-1 - прозрачный)
            const int BK_COLOR[] = {-1, 0x00FF00, 0x00FF00, 0x00FF00, -1, -1};

            // количество фигур
            const int FIGURES_COUNT = ARRAYSIZE(PEN_SIZES);

            // количество строк
            const int ROWS = 2;

            // количество фигур в строке
            const int FIGURES_PER_ROW = FIGURES_COUNT / ROWS;

            // отступ от края окна и между фигурами
            const int MARGIN = 10;

            // получаю информацию об окне
            WINDOWINFO wi;
            GetWindowInfo(hWnd, &wi);

            // размер фигуры включая отступ
            const int WIDTH = std::min(
                    (wi.rcClient.right - wi.rcClient.left - MARGIN) / FIGURES_PER_ROW, //
по ширине
                    (wi.rcClient.bottom - wi.rcClient.top - MARGIN) / ROWS // по высоте
            );

            // Начинаю отрисовку
            PAINTSTRUCT ps;
            HDC hDC = BeginPaint(hWnd, &ps);

            int i;
            for (i = 0; i < FIGURES_COUNT; i++) {
                // Определяю координаты фигуры
                int x = (i % FIGURES_PER_ROW) * WIDTH + MARGIN;
                int y = (i / FIGURES_PER_ROW) * WIDTH + MARGIN;
                // Создаю перо
                HGDIOBJ hPen = CreatePen(PEN_STYLES[i], PEN_SIZES[i], PEN_COLORS[i]);
                // Создаю кисть
                HGDIOBJ hBrush;
                if (BRUSH_HATCH[i] == -1) {
                    hBrush = CreateSolidBrush(BRUSH_COLOR[i]);
                } else {
                    hBrush = CreateHatchBrush(BRUSH_HATCH[i], BRUSH_COLOR[i]);
                }
                // Выбираю перо
                SelectObject(hDC, hPen);
                // Выбираю кисть
                SelectObject(hDC, hBrush);
                // Устанавливаю цвет фона
                if (BK_COLOR[i] >= 0) {
                    SetBkColor(hDC, BK_COLOR[i]);
                    SetBkMode(hDC, OPAQUE);
                } else {
                    SetBkMode(hDC, TRANSPARENT);
                }
                // Чередую круги и квадраты начиная с квадрата
```

```
                if (i & 1) {
                    Rectangle(hDC, x, y, x + WIDTH - MARGIN, y + WIDTH - MARGIN);
                } else {
                    Ellipse(hDC, x, y, x + WIDTH - MARGIN, y + WIDTH - MARGIN);
                }
                // Удаляю перо
                DeleteObject(hPen);
                // Удаляю кисть
                DeleteObject(hBrush);
                // Красный цвет текста
                SetTextColor(hDC, 0x0000FF);
                // Обратный цвет фона для текст
                SetBkColor(hDC, ~0x0000FF & 0xFFFFFF);
                // Видимый фон
                SetBkMode(hDC, OPAQUE);
                // Вывожу текст
                TextOut(hDC, 50, 50, "Здесь мое ФИО", 13);
                // Если когда-то нажималась правая кнопка
                if (lastRX >= 0) {
                    TCHAR text[] = "Правая кнопка мыши!";
                    // Прозрачный фон
                    SetBkMode(hDC, TRANSPARENT);
                    // вывожу в место последнего нажатия текст
                    TextOut(hDC, lastRX, lastRY, text, ARRAYSIZE(text));
                }
            }
            // Заканчиваю отрисовку
            EndPaint(hWnd, &ps);
            break;
        }

        default: // в противном случае вызов стандартной оконной функции
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}


int WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
    HWND hWnd;
    MSG msg;
    WNDCLASSEX wcl;
    char szWinName[] = "MyClassName";
    wcl.cbSize = sizeof(WNDCLASSEX); // размер структуры
    wcl.hInstance = hInstance; // дескриптор программы
    wcl.lpszClassName = (LPCSTR) szWinName; // класс окна
    wcl.lpfnWndProc = WindowFunc; // оконная функция
    wcl.style = 0; // стипль по умолчанию
    wcl.hIcon = LoadIcon(NULL, IDI_ERROR); // иконка
    wcl.hIconSm = LoadIcon(NULL, IDI_ERROR); // малая иконка
    wcl.hCursor = LoadCursor(NULL, IDC_CROSS); // курсор
    wcl.lpszMenuName = NULL; // меню
    wcl.cbClsExtra = 0; // нет дополнительной памяти для класса
    wcl.cbWndExtra = 0; // нет дополнительной памяти для экземпляра
    wcl.hbrBackground = CreateSolidBrush(0x00FFFF); // желтая кисть

    // Регистрирую класс
    ATOM wndClass = RegisterClassEx(&wcl);

    // Если зарегистрировал
    if (wndClass) {
        // Создаю окно
        hWnd = CreateWindowEx(
                0,
                (LPCSTR)wndClass,        // класс окна
                (LPCSTR) "Пример какого то варианта", // заголовок
                WS_OVERLAPPED | WS_SYSMENU, // только рамка и системное меню
                10, 50, // координаты
                500, 400, // размеры
```

```
                HWND_DESKTOP,
                NULL,
                GetModuleHandle(NULL),
                NULL
        );
        /*      Отображение окна        */
        ShowWindow(hWnd, SW_SHOW);
        /*      Цикл сообщений          */
        while (GetMessage(&msg, NULL, 0, 0)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        UnregisterClass((LPCSTR)wndClass, hInstance);
    }
    // Удаляю желтую кисть
    DeleteObject(wcl.hbrBackground);
}
```

10. Изменил реакцию на нажатие правой кнопки так, чтобы в позиции курсора выводился порядковый номер нажатия, рисовались отрезки зеленого цвета исходящие из верхнего левого угла окна и рисовались отрезки соединяющие текущую позицию курсора мыши с предыдущей. Рисование осуществляется в память.



Код программы

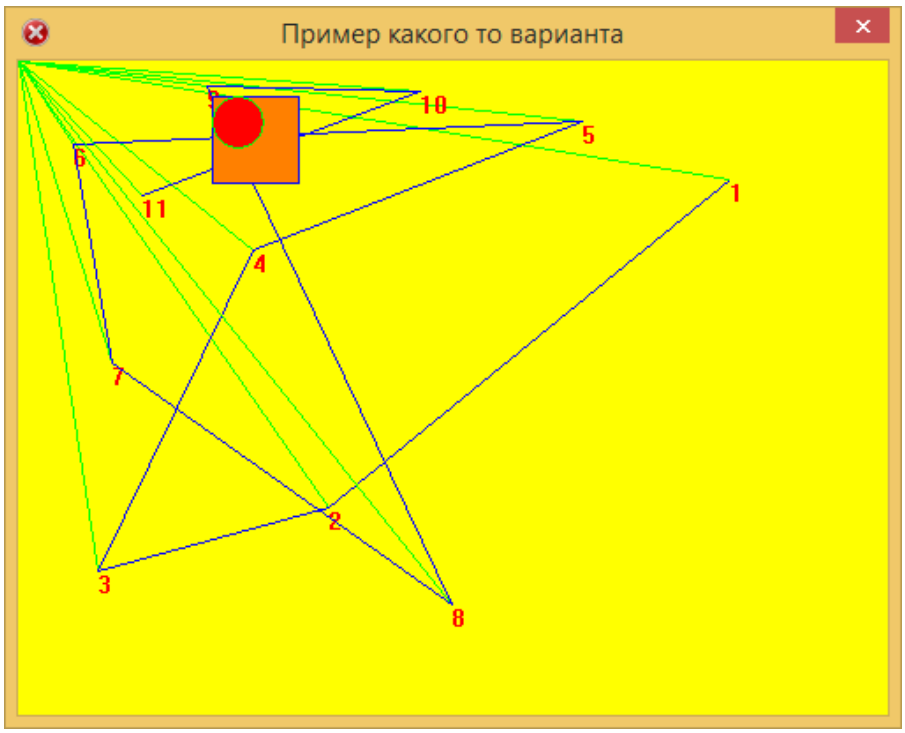
```
#include <windows.h>

/* Координаты последнего нажатия правой кнопки (-1 - нажатия не было) */
int lastRX = -1, lastRY = -1;

/* Порядковый номер нажатия на правую кнопку */
int rCounter = 0;

/* Контекст устройства памяти */
HDC hdcMemDC;

/* Растр */
HBITMAP hbmMemory;

/* Перья */
HPEN hGreenPen, hBluePen;

/**
 * Оконная функция
 * @param hWnd Окно
 * @param message сообщение
 * @param wParam w-параметр
```

```
 * @param lParam l-параметр
 * @return результат выполнения
 */
LRESULT CALLBACK WindowFunc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
    switch (message) {
        case WM_CREATE: // создание окна
        {
            /* Создаю зеленое перо */
            hGreenPen = CreatePen(PS_SOLID, 1, 0x00FF00);
            /* Создаю синее перо */
            hBluePen = CreatePen(PS_SOLID, 1, 0xFF0000);
            // Получаю контекст устройства окна
            HDC hDC = GetDC(hWnd);
            // Создаю совместимый контекст устройства памяти
            hdcMemDC = CreateCompatibleDC(hDC);
            // Получаю размеры экрана
            int width = GetSystemMetrics(SM_CXSCREEN);
            int height = GetSystemMetrics(SM_CXSCREEN);
            /** Создаю растр размером с экран совместимый с контекстом
                устройства окна. В нем будет хранится изображение окна для быстрой
                отрисовки. */
            hbmMemory = CreateCompatibleBitmap(hDC, width, height);
            // И выбираю его в контекст устройства памяти
            SelectObject(hdcMemDC, hbmMemory);
            // Получаю кисть, которая используется для закраски фона
            HGDIOBJ hBrush = (HGDIOBJ) (GetClassLongPtr(hWnd, GCLP_HBRBACKGROUND));
            // Выбираю кисть в контекст устройства памяти
            SelectObject(hdcMemDC, hBrush);
            // Закрашиваю растр кистью
            PatBlt(hdcMemDC, 0, 0, width, height, PATCOPY);
            // Освобождаю контекст устройства окна
            ReleaseDC(hWnd, hDC);
            break;
        }

        case WM_DESTROY: // уничтожение окна
            // удаляю зеленое перо
            DeleteObject(hGreenPen);
            // удаляю синее перо
            DeleteObject(hBluePen);
            // удаляю растр
            DeleteObject(hbmMemory);
            // удаляю контекст устройства с памятью
            DeleteDC(hdcMemDC);
            // отправляю сообщение WM_QUIT
            PostQuitMessage(0);
            break;

        case WM_LBUTTONDOWN: // левая кнпока мыши
            // Если пользователь отвечает "да"
            if (MessageBox(hWnd, "Выводим?", "Вопрос", MB_ICONQUESTION | MB_YESNO) ==
IDYES) {
                // Декодирую координаты
                WORD x = LOWORD(lParam);
                WORD y = HIWORD(lParam);
                // буфер достаточной длины для строки вывода координат
                char mem[400];
                // форматирую сообщение
                wsprintf(mem, "X = %d; Y = %d", x, y);
                // вывожу сообщение
                MessageBox(hWnd, mem, "Координаты", MB_ICONINFORMATION | MB_OK);
            }
            break;

        case WM_RBUTTONDOWN: // правая кнопка мыши
        {
            // Декодирую координаты
            int x = LOWORD(lParam);
            int y = HIWORD(lParam);
            // Красный цвет текста
```

```
            SetTextColor(hdcMemDC, 0x0000FF);
            // Прозрачный фон
            SetBkMode(hdcMemDC, TRANSPARENT);
            // Увеличиваю счетчик нажатия
            rCounter++;
            // Буфер достаточной длины для строки вывода счетчика
            char mem[10];
            // Форматирую сообщение
            int size = wsprintf(mem, "%d", rCounter);
            // Рисую счетчик в координатах курсора в контексте устройства памяти
            TextOut(hdcMemDC, x, y, (LPCSTR) mem, size);
            // Перемещаемся в позицию (0, 0)
            MoveToEx(hdcMemDC, 0, 0, NULL);
            // Выбираю зеленое перо
            SelectObject(hdcMemDC, hGreenPen);
            // Рисую в памяти отрезок от верхнего левого угла до позиции курсора
            LineTo(hdcMemDC, x, y);
            // Если это не первое нажатие
            if (rCounter > 1) {
                // Выбираю синее перо
                SelectObject(hdcMemDC, hBluePen);
                // Рисую отрезок от текущей позиции курсора к предыдузей
                LineTo(hdcMemDC, lastRX, lastRY);
            }
            // Окно нуждается в перерисовке
            InvalidateRect(hWnd, NULL, true);
            // Сохраняю координаты
            lastRX = x;
            lastRY = y;
            break;
        }

        case WM_PAINT: // отрисовка
        {

            // Начинаю отрисовку
            PAINTSTRUCT ps;
            HDC hDC = BeginPaint(hWnd, &ps);

            /* Копирую графику из контекста устройства памяти в конкекст устройства
               окна в области, которую нужно перерисовать */
            BitBlt(hDC, ps.rcPaint.left, ps.rcPaint.top,
                   ps.rcPaint.right - ps.rcPaint.left, ps.rcPaint.bottom -
ps.rcPaint.top,
                   hdcMemDC, ps.rcPaint.left, ps.rcPaint.top, SRCCOPY);

            // Заканчиваю отрисовку
            EndPaint(hWnd, &ps);
            break;
        }

        default: // в противном случае вызов стандартной оконной функции
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}


int WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
    HWND hWnd;
    MSG msg;
    WNDCLASSEX wcl;
    char szWinName[] = "MyClassName";
    wcl.cbSize = sizeof(WNDCLASSEX); // размер структуры
    wcl.hInstance = hInstance; // дескриптор программы
    wcl.lpszClassName = (LPCSTR) szWinName; // класс окна
    wcl.lpfnWndProc = WindowFunc; // оконная функция
    wcl.style = 0; // стипль по умолчанию
    wcl.hIcon = LoadIcon(NULL, IDI_ERROR); // иконка
```

```
    wcl.hIconSm = LoadIcon(NULL, IDI_ERROR); // малая иконка
    wcl.hCursor = LoadCursor(NULL, IDC_CROSS); // курсор
    wcl.lpszMenuName = NULL; // меню
    wcl.cbClsExtra = 0; // нет дополнительной памяти для класса
    wcl.cbWndExtra = 0; // нет дополнительной памяти для экземпляра
    wcl.hbrBackground = CreateSolidBrush(0x00FFFF); // желтая кисть

    // Регистрирую класс
    ATOM wndClass = RegisterClassEx(&wcl);

    // Если зарегистрировал
    if (wndClass) {
        // Создаю окно
        hWnd = CreateWindowEx(
                0,
                (LPCSTR) wndClass,         // класс окна
                (LPCSTR) "Пример какого то варианта", // заголовок
                WS_OVERLAPPED | WS_SYSMENU, // только рамка и системное меню
                10, 50, // координаты
                500, 400, // размеры
                HWND_DESKTOP,
                NULL,
                GetModuleHandle(NULL),
                NULL
        );
        /*      Отображение окна        */
        ShowWindow(hWnd, SW_SHOW);
        /*      Цикл сообщений          */
        while (GetMessage(&msg, NULL, 0, 0)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        UnregisterClass((LPCSTR) wndClass, hInstance);
    }
    // Удаляю желтую кисть
    DeleteObject(wcl.hbrBackground);
}
```

11. Изменил программу так, чтобы в позиции окна с координатами (20, 20) выводился рисунок, состоящий из оранжевого квадрата с синим контуром и поверх красного круга с зеленым контуром. При нажатии левой кнопки мыши рисунок сдвигается влево на 10 пикселей, а при нажатии правой — в право.



Код программы

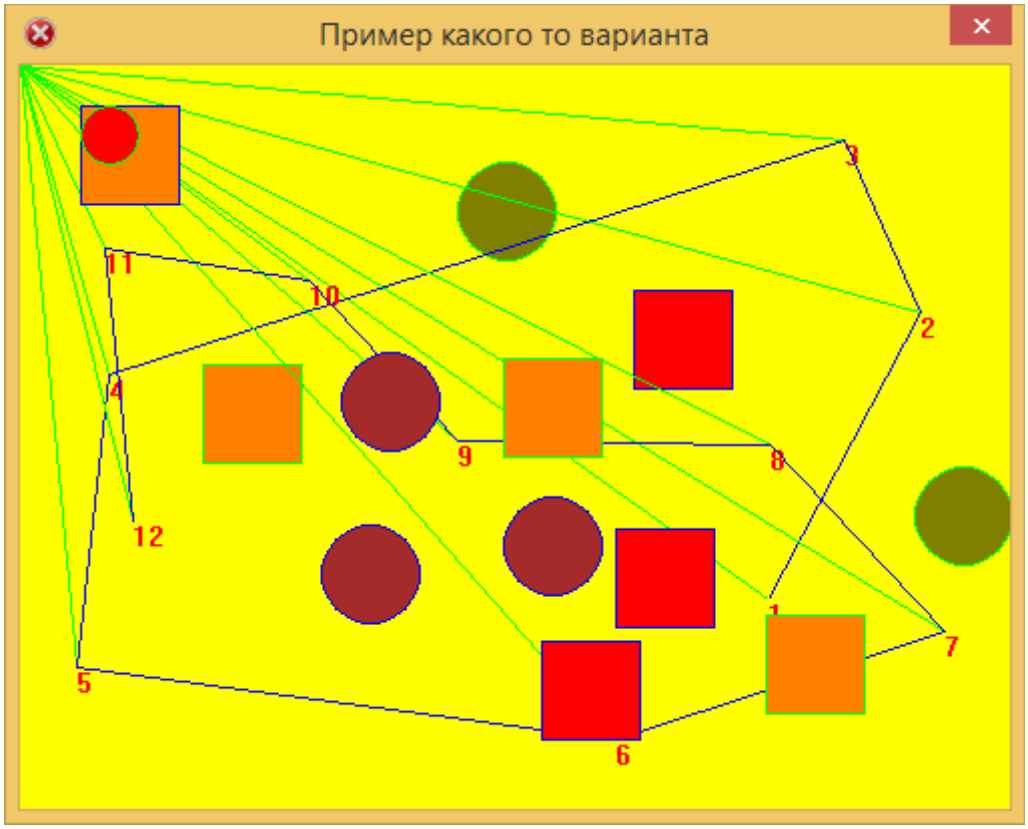
```
#include <windows.h>

/* Координаты последнего нажатия правой кнопки (-1 - нажатия не было) */
int lastRX = -1, lastRY = -1;

/* Порядковый номер нажатия на правую кнопку */
int rCounter = 0;

/* Контекст устройства памяти */
HDC hdcMemDC;

/* Растр */
HBITMAP hbmMemory;

/* Перья */
HPEN hGreenPen, hBluePen;

/* Левая координата рисунка */
int imageX = 20;

/**
 * Оконная функция
 * @param hWnd Окно
 * @param message сообщение
 * @param wParam w-параметр
 * @param lParam l-параметр
 * @return результат выполнения
 */
LRESULT CALLBACK WindowFunc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
    switch (message) {
        case WM_CREATE: // создание окна
        {
            /* Создаю зеленое перо */
            hGreenPen = CreatePen(PS_SOLID, 1, 0x00FF00);
            /* Создаю синее перо */
            hBluePen = CreatePen(PS_SOLID, 1, 0xFF0000);
            // Получаю контекст устройства окна
            HDC hDC = GetDC(hWnd);
            // Создаю совместимый контекст устройства памяти
            hdcMemDC = CreateCompatibleDC(hDC);
            // Получаю размеры экрана
            int width = GetSystemMetrics(SM_CXSCREEN);
            int height = GetSystemMetrics(SM_CXSCREEN);
            /** Создаю растр размером с экран совместимый с контекстом
                устройства окна. В нем будет хранится изображение окна для быстрой
                отрисовки. */
            hbmMemory = CreateCompatibleBitmap(hDC, width, height);
            // И выбираю его в контекст устройства памяти
            SelectObject(hdcMemDC, hbmMemory);
            // Получаю кисть, которая используется для закраски фона
            HGDIOBJ hBrush = (HGDIOBJ) (GetClassLongPtr(hWnd, GCLP_HBRBACKGROUND));
            // Выбираю кисть в контекст устройства памяти
            SelectObject(hdcMemDC, hBrush);
            // Закрашиваю растр кистью
            PatBlt(hdcMemDC, 0, 0, width, height, PATCOPY);
            // Освобождаю контекст устройства окна
            ReleaseDC(hWnd, hDC);
            break;
        }

        case WM_DESTROY: // уничтожение окна
            // удаляю зеленое перо
            DeleteObject(hGreenPen);
            // удаляю синее перо
            DeleteObject(hBluePen);
            // удаляю растр
            DeleteObject(hbmMemory);
            // удаляю контекст устройства с памятью
            DeleteDC(hdcMemDC);
```

```
            // отправляю сообщение WM_QUIT
            PostQuitMessage(0);
            break;

        case WM_LBUTTONDOWN: // левая кнпока мыши
            // Сдвиг рисунка влево на 10 пикселей
            imageX -= 10;
            // Окно нуждается в перерисовке
            InvalidateRect(hWnd, NULL, true);
            break;

        case WM_RBUTTONDOWN: // правая кнопка мыши
        {
            // Сдвиг рисунка вправо на 10 пикселей
            imageX += 10;
            // Декодирую координаты
            int x = LOWORD(lParam);
            int y = HIWORD(lParam);
            // Красный цвет текста
            SetTextColor(hdcMemDC, 0x0000FF);
            // Прозрачный фон
            SetBkMode(hdcMemDC, TRANSPARENT);
            // Увеличиваю счетчик нажатия
            rCounter++;
            // Буфер достаточной длины для строки вывода счетчика
            char mem[10];
            // Форматирую сообщение
            int size = wsprintf(mem, "%d", rCounter);
            // Рисую счетчик в координатах курсора в контексте устройства памяти
            TextOut(hdcMemDC, x, y, (LPCSTR) mem, size);
            // Перемещаемся в позицию (0, 0)
            MoveToEx(hdcMemDC, 0, 0, NULL);
            // Выбираю зеленое перо
            SelectObject(hdcMemDC, hGreenPen);
            // Рисую в памяти отрезок от верхнего левого угла до позиции курсора
            LineTo(hdcMemDC, x, y);
            // Если это не первое нажатие
            if (rCounter > 1) {
                // Выбираю синее перо
                SelectObject(hdcMemDC, hBluePen);
                // Рисую отрезок от текущей позиции курсора к предыдузей
                LineTo(hdcMemDC, lastRX, lastRY);
            }
            // Окно нуждается в перерисовке
            InvalidateRect(hWnd, NULL, true);
            // Сохраняю координаты
            lastRX = x;
            lastRY = y;
            break;
        }

        case WM_PAINT: // отрисовка
        {

            // Начинаю отрисовку
            PAINTSTRUCT ps;
            HDC hDC = BeginPaint(hWnd, &ps);

            /* Копирую графику из контекста устройства памяти в конкекст устройства
               окна в области, которую нужно перерисовать */
            BitBlt(hDC, ps.rcPaint.left, ps.rcPaint.top,
                   ps.rcPaint.right - ps.rcPaint.left, ps.rcPaint.bottom -
ps.rcPaint.top,
                   hdcMemDC, ps.rcPaint.left, ps.rcPaint.top, SRCCOPY);

            // Создаю красную кисть
            HGDIOBJ hRedBrush = CreateSolidBrush(0x0000FF);
            // Создаю оранжевую кисть
            HGDIOBJ hOrangeBrush = CreateSolidBrush(0x0080FF);
            // Выбираю синее перо
```

```
            SelectObject(hDC, hBluePen);
            // Выбираю оранжевую кисть
            SelectObject(hDC, hOrangeBrush);
            // Рисую квадрат
            Rectangle(hDC, imageX, 20, imageX + 50, 70);
            // Выбираю зеленое перо
            SelectObject(hDC, hGreenPen);
            // Выбираю красную кисть
            SelectObject(hDC, hRedBrush);
            // Рисую круг
            Ellipse(hDC, imageX, 20, imageX + 30, 50);
            // Удаляю красную кисть
            DeleteObject(hRedBrush);
            // Удаляю оранжевую кисть
            DeleteObject(hOrangeBrush);
            // Заканчиваю отрисовку
            EndPaint(hWnd, &ps);
            break;
        }

        default: // в противном случае вызов стандартной оконной функции
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}


int WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
    HWND hWnd;
    MSG msg;
    WNDCLASSEX wcl;
    char szWinName[] = "MyClassName";
    wcl.cbSize = sizeof(WNDCLASSEX); // размер структуры
    wcl.hInstance = hInstance; // дескриптор программы
    wcl.lpszClassName = (LPCSTR) szWinName; // класс окна
    wcl.lpfnWndProc = WindowFunc; // оконная функция
    wcl.style = 0; // стипль по умолчанию
    wcl.hIcon = LoadIcon(NULL, IDI_ERROR); // иконка
    wcl.hIconSm = LoadIcon(NULL, IDI_ERROR); // малая иконка
    wcl.hCursor = LoadCursor(NULL, IDC_CROSS); // курсор
    wcl.lpszMenuName = NULL; // меню
    wcl.cbClsExtra = 0; // нет дополнительной памяти для класса
    wcl.cbWndExtra = 0; // нет дополнительной памяти для экземпляра
    wcl.hbrBackground = CreateSolidBrush(0x00FFFF); // желтая кисть

    // Регистрирую класс
    ATOM wndClass = RegisterClassEx(&wcl);

    // Если зарегистрировал
    if (wndClass) {
        // Создаю окно
        hWnd = CreateWindowEx(
                0,
                (LPCSTR) wndClass,         // класс окна
                (LPCSTR) "Пример какого то варианта", // заголовок
                WS_OVERLAPPED | WS_SYSMENU, // только рамка и системное меню
                10, 50, // координаты
                500, 400, // размеры
                HWND_DESKTOP,
                NULL,
                GetModuleHandle(NULL),
                NULL
        );
        /*     Отображение окна       */
        ShowWindow(hWnd, SW_SHOW);
        /*     Цикл сообщений        */
        while (GetMessage(&msg, NULL, 0, 0)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
```

```
        }
        UnregisterClass((LPCSTR) wndClass, hInstance);
        return msg.wParam;
    }
    // Удаляю желтую кисть
    DeleteObject(wcl.hbrBackground);
}
```

12. Изменил программу так, чтобы при каждом нажатии левой кнопки мыши в координатах нажатия поочередно выводились квадрат и круг. Квадраты поочередно выводятся синим контуром и красным фоном и зеленым контуром и оранжевым фоном. Круги поочередно выводятся с синим контуром и коричневым фоном и зеленым контуром и фоном оливкового цвета.



Код программы

```
#include <windows.h>

/* Координаты последнего нажатия правой кнопки (-1 - нажатия не было) */
int lastRX = -1, lastRY = -1;

/* Порядковый номер нажатия на правую кнопку */
int rCounter = 0;

/* Порядковый номер нажатия на левую кнопку */
int lCounter = 0;

/* Контекст устройства памяти */
HDC hdcMemDC;

/* Растр */
HBITMAP hbmMemory;

/* Перья */
HPEN hGreenPen, hBluePen;

/* Кисти */
```

```
HBRUSH hRedBrush, hOrangeBrush, hBrownBrush, hOliveBrush;

/* Левая координата рисунка */
int imageX = 20;

/**
 * Оконная функция
 * @param hWnd Окно
 * @param message сообщение
 * @param wParam w-параметр
 * @param lParam l-параметр
 * @return результат выполнения
 */
LRESULT CALLBACK WindowFunc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
    switch (message) {
        case WM_CREATE: // создание окна
        {
            /* Создаю зеленое перо */
            hGreenPen = CreatePen(PS_SOLID, 1, 0x00FF00);
            /* Создаю синее перо */
            hBluePen = CreatePen(PS_SOLID, 1, 0xFF0000);
            // Создаю красную кисть
            hRedBrush = CreateSolidBrush(0x0000FF);
            // Создаю оранжевую кисть
            hOrangeBrush = CreateSolidBrush(0x0080FF);
            // Создаю коричневую кисть
            hBrownBrush = CreateSolidBrush(0x2A2AA5);
            // Создаю кисть оливкового цвета
            hOliveBrush = CreateSolidBrush(0x008080);
            // Получаю контекст устройства окна
            HDC hDC = GetDC(hWnd);
            // Создаю совместимый контекст устройства памяти
            hdcMemDC = CreateCompatibleDC(hDC);
            // Получаю размеры экрана
            int width = GetSystemMetrics(SM_CXSCREEN);
            int height = GetSystemMetrics(SM_CXSCREEN);
            /** Создаю растр размером с экран совместимый с контекстом
                устройства окна. В нем будет хранится изображение окна для быстрой
                отрисовки. */
            hbmMemory = CreateCompatibleBitmap(hDC, width, height);
            // И выбираю его в контекст устройства памяти
            SelectObject(hdcMemDC, hbmMemory);
            // Получаю кисть, которая используется для закраски фона
            HGDIOBJ hBrush = (HGDIOBJ) (GetClassLongPtr(hWnd, GCLP_HBRBACKGROUND));
            // Выбираю кисть в контекст устройства памяти
            SelectObject(hdcMemDC, hBrush);
            // Закрашиваю растр кистью
            PatBlt(hdcMemDC, 0, 0, width, height, PATCOPY);
            // Освобождаю контекст устройства окна
            ReleaseDC(hWnd, hDC);
            break;
        }

        case WM_DESTROY: // уничтожение окна
            // удаляю зеленое перо
            DeleteObject(hGreenPen);
            // удаляю синее перо
            DeleteObject(hBluePen);
            // удаляю красную кисть
            DeleteObject(hRedBrush);
            // удаляю оранжевую кисть
            DeleteObject(hOrangeBrush);
            // удаляю коричневую кисть
            DeleteObject(hBrownBrush);
            // удаляю кисть оливкового цвета
            DeleteObject(hOliveBrush);
            // удаляю растр
            DeleteObject(hbmMemory);
            // удаляю контекст устройства с памятью
            DeleteDC(hdcMemDC);
```

```
        // отправляю сообщение WM_QUIT
        PostQuitMessage(0);
        break;

    case WM_LBUTTONDOWN: // левая кнпока мыши
    {
        // Сдвиг рисунка влево на 10 пикселей
        imageX -= 10;
        // Декодирую координаты
        int x = LOWORD(lParam);
        int y = HIWORD(lParam);
        // Увеличиваю счетчик нажатия
        lCounter++;
        switch (lCounter & 3) {
            case 0: // круг оливкового цвета с зеленым контуром контуром
                // Выбираю зеленое перо
                SelectObject(hdcMemDC, hGreenPen);
                // Выбираю кисть оливкового цвета
                SelectObject(hdcMemDC, hOliveBrush);
                // Рисую круг
                Ellipse(hdcMemDC, x, y, x + 50, y + 50);
                break;
            case 1: // красный квадрат с синим контуром
                // Выбираю синее перо
                SelectObject(hdcMemDC, hBluePen);
                // Выбираю красную кисть
                SelectObject(hdcMemDC, hRedBrush);
                // Рисую квадрат
                Rectangle(hdcMemDC, x, y, x + 50, y + 50);
                break;
            case 2: // коричневый круг с синим контуром
                // Выбираю синее перо
                SelectObject(hdcMemDC, hBluePen);
                // Выбираю коричневую кисть
                SelectObject(hdcMemDC, hBrownBrush);
                // Рисую круг
                Ellipse(hdcMemDC, x, y, x + 50, y + 50);
                break;
            case 3: // оранжевый квадрат с зеленым контуром
                // Выбираю зеленое перо
                SelectObject(hdcMemDC, hGreenPen);
                // Выбираю оранжевую кисть
                SelectObject(hdcMemDC, hOrangeBrush);
                // Рисую квадрат
                Rectangle(hdcMemDC, x, y, x + 50, y + 50);
                break;
        }
        // Окно нуждается в перерисовке
        InvalidateRect(hWnd, NULL, true);
        break;
    }

    case WM_RBUTTONDOWN: // правая кнопка мыши
    {
        // Сдвиг рисунка вправо на 10 пикселей
        imageX += 10;
        // Декодирую координаты
        int x = LOWORD(lParam);
        int y = HIWORD(lParam);
        // Красный цвет текста
        SetTextColor(hdcMemDC, 0x0000FF);
        // Прозрачный фон
        SetBkMode(hdcMemDC, TRANSPARENT);
        // Увеличиваю счетчик нажатия
        rCounter++;
        // Буфер достаточной длины для строки вывода счетчика
        char mem[10];
        // Форматирую сообщение
        int size = wsprintf(mem, "%d", rCounter);
        // Рисую счетчик в координатах курсора в контексте устройства памяти
```

```
            TextOut(hdcMemDC, x, y, (LPCSTR) mem, size);
            // Перемещаемся в позицию (0, 0)
            MoveToEx(hdcMemDC, 0, 0, NULL);
            // Выбираю зеленое перо
            SelectObject(hdcMemDC, hGreenPen);
            // Рисую в памяти отрезок от верхнего левого угла до позиции курсора
            LineTo(hdcMemDC, x, y);
            // Если это не первое нажатие
            if (rCounter > 1) {
                // Выбираю синее перо
                SelectObject(hdcMemDC, hBluePen);
                // Рисую отрезок от текущей позиции курсора к предыдузей
                LineTo(hdcMemDC, lastRX, lastRY);
            }
            // Окно нуждается в перерисовке
            InvalidateRect(hWnd, NULL, true);
            // Сохраняю координаты
            lastRX = x;
            lastRY = y;
            break;
        }

        case WM_PAINT: // отрисовка
        {

            // Начинаю отрисовку
            PAINTSTRUCT ps;
            HDC hDC = BeginPaint(hWnd, &ps);

            /* Копирую графику из контекста устройства памяти в конкекст устройства
               окна в области, которую нужно перерисовать */
            BitBlt(hDC, ps.rcPaint.left, ps.rcPaint.top,
                   ps.rcPaint.right - ps.rcPaint.left, ps.rcPaint.bottom -
ps.rcPaint.top,
                   hdcMemDC, ps.rcPaint.left, ps.rcPaint.top, SRCCOPY);

            // Выбираю синее перо
            SelectObject(hDC, hBluePen);
            // Выбираю оранжевую кисть
            SelectObject(hDC, hOrangeBrush);
            // Рисую квадрат
            Rectangle(hDC, imageX, 20, imageX + 50, 70);
            // Выбираю зеленое перо
            SelectObject(hDC, hGreenPen);
            // Выбираю красную кисть
            SelectObject(hDC, hRedBrush);
            // Рисую круг
            Ellipse(hDC, imageX, 20, imageX + 30, 50);
            // Заканчиваю отрисовку
            EndPaint(hWnd, &ps);
            break;
        }

        default: // в противном случае вызов стандартной оконной функции
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}


int WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
    HWND hWnd;
    MSG msg;
    WNDCLASSEX wcl;
    char szWinName[] = "MyClassName";
    wcl.cbSize = sizeof(WNDCLASSEX); // размер структуры
    wcl.hInstance = hInstance; // дескриптор программы
    wcl.lpszClassName = (LPCSTR) szWinName; // класс окна
    wcl.lpfnWndProc = WindowFunc; // оконная функция
```

```
    wcl.style = 0; // стипль по умолчанию
    wcl.hIcon = LoadIcon(NULL, IDI_ERROR); // иконка
    wcl.hIconSm = LoadIcon(NULL, IDI_ERROR); // малая иконка
    wcl.hCursor = LoadCursor(NULL, IDC_CROSS); // курсор
    wcl.lpszMenuName = NULL; // меню
    wcl.cbClsExtra = 0; // нет дополнительной памяти для класса
    wcl.cbWndExtra = 0; // нет дополнительной памяти для экземпляра
    wcl.hbrBackground = CreateSolidBrush(0x00FFFF); // желтая кисть

    // Регистрирую класс
    ATOM wndClass = RegisterClassEx(&wcl);

    // Если зарегистрировал
    if (wndClass) {
        // Создаю окно
        hWnd = CreateWindowEx(
                0,
                (LPCSTR) wndClass,         // класс окна
                (LPCSTR) "Пример какого то варианта", // заголовок
                WS_OVERLAPPED | WS_SYSMENU, // только рамка и системное меню
                10, 50, // координаты
                500, 400, // размеры
                HWND_DESKTOP,
                NULL,
                GetModuleHandle(NULL),
                NULL
        );
        /*      Отображение окна        */
        ShowWindow(hWnd, SW_SHOW);
        /*      Цикл сообщений         */
        while (GetMessage(&msg, NULL, 0, 0)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        UnregisterClass((LPCSTR) wndClass, hInstance);
        return msg.wParam;
    }
    // Удаляю желтую кисть
    DeleteObject(wcl.hbrBackground);
}
```