**Министерство науки и высшего образования РФ**

**ФГБОУ ВО  
«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**Кафедра «Информатика и программное обеспечение»**

**Расчетно-графическая работа**

**по дисциплине «Программирование приложений Windows»**

направление подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем»

на тему:

**Работа с окнами, пользовательским интерфейсом. Работа с меню**

Вариант №67

Выполнил студ. гр. 17-МОА:

**Нагибин В. А.**

Проверил:

к.т.н., доц. **Белов Е. А.**

Брянск 2020

Оглавление

[1. Задания 3](#_Toc42443299)

[2. Теоретический материал 4](#_Toc42443300)

[2.1. Вывод графики в окно 4](#_Toc42443303)

[2.2. Стили окон 6](#_Toc42443307)

[3. Описание программ 7](#_Toc42443310)

[3.1. Задание №1 7](#_Toc42443312)

[3.1. Задание №2 8](#_Toc42443316)

[4. Контрольные примеры работы программ 10](#_Toc42443317)

[4.1. Задание №1 10](#_Toc42443322)

[4.2. Задание №2 10](#_Toc42443328)

[Список литературы 12](#_Toc42443332)

1. Задания

* Задание №1. Создать макет калькулятора, содержащего поле ввода чисел, кнопки цифр от 0 до 9, десятичной точки, изменения знака числа, арифметических действий и "=".
* Задание №2. Нарисовать светофор, в котором цвет "зажигается" при нажатии на эту лампу левой клавишей мыши. Одновременно может гореть только один цвет.

1. Теоретический материал
3. 1. Вывод графики в окно

Вывод графики в окна обладает рядом особенностей.

* Нельзя пользоваться функциями вывода библиотеки компилятора, так как они приспособлены для вывода в одно и единственное окно. В операционной системе Windows приложения выводят одновременно в различные окна. Система сама решает все проблемы, связанные с возможным перекрытием или перемещением этих окон. С этим обстоятельством связано то, что в окно стремятся выводить в одном месте приложения – при обработке сообщения WM\_PAINT. Приложение описывают таким образом, чтобы при поступлении сообщения WM\_PAINT функция окна могла перерисовать все окно или любую его заданную часть.
* Интерфейс графических устройств (Graphics Device Interface – GDI) системы открывает доступ к большому числу функций вывода. Приложения, обращаясь к функциям GDI, работают не с физическими устройствами вывода, а с логическими, т. е. описание вызова функций не зависит от физического способа отображения. Функция GDI передает указания о выводе драйверу устройства вывода. Драйвер работает непосредственно с физическим устройством и при управлении выводом учитывает его ограниченные возможности и аппаратные особенности. Благодаря этому приложения способны работать с любым устройством вывода, драйвер которого установлен в системе. Однако не все физические устройства вывода способны поддерживать те режимы, в которых работает видеомонитор, поэтому, описывая последовательность операций вывода, например на принтер, следует учитывать ограниченные возможности установленного принтера.
* Параметры вывода устанавливают в контексте отображения с помощью функций GDI. Контекст отображения – это структура данных, которая содержит характеристики устройства вывода и указатели на выбранные инструменты рисования. Функции GDI используют только выбранные в контекст отображения параметры и инструменты рисования, например для рисования линии некоторой толщины в контекст отображения, приложение должно выбрать перо этой толщины.
* Дескриптор контекста отображения служит первым аргументом вызова всех функций, связанных с выводом в окно.

GDI, с точки зрения программиста – это контекст отображения. Контекст отображения можно сравнить с листом бумаги, на котором приложение рисует, а также инструментом для рисования. Инструменты для рисования – это перья, кисти, а также шрифты и целые графические изображения, с помощью которых приложение создаст необходимый рисунок. Кроме контекста отображения и инструмента для рисования, приложению доступны десятки функций программного интерфейса GDI.

Контекст отображения является структурой данных, описывающих устройство отображения. В этой структуре хранятся различные характеристики устройства и набор инструментов для рисования, выбранные по умолчанию. Приложение может выбрать в контекст отображения различные инструменты (например, перья различной толщины и цвета). Поэтому, если необходимо нарисовать линию красного цвета, то перед выполнением операции рисования, необходимо выбрать в контекст отображения соответствующее перо.

Функции рисования не имеют параметров, указывающих цвет и толщину линий. Такие параметры хранятся в контексте отображения.

Приложение может создать контекст отображения не только для экрана монитора, но и для любого другою графического устройства, например, для принтера. В этом случае оно может рисовать на принтере различные изображения, используя те же функции, что и при рисовании на экране. Можно создать контекст отображения для метафайла. Метафайл – это обычный файл на диске или в памяти, в котором хранятся последовательности команд интерфейса GDI. Приложение может выполнить вывод графической информации в метафайл, как в обычное устройство вывода, а затем «проиграть» метафайл на реальном устройстве вывода.

2. 2. Стили окон

По совокупности свойств различают перекрывающиеся (overlapped), временные (pop-up) и дочерние (child) окна.

Перекрывающиеся окна чаще используют в качестве окон приложения. Они всегда имеют заголовок, рамку и рабочую область окна, могут иметь системное меню, кнопки восстановления размеров, закрытия и сворачивания окна в пиктограмму, горизонтальную и вертикальную полосы просмотра, меню, панель инструментов и строку состояния. Базовый стиль таких окон описан константой WS\_OVERLAPPED. Чаще используемый стиль окон WS\_OVERLAPPEDWINDOW в дополнение к базовому указывает, что окно имеет системное меню, кнопки восстановления размеров, закрытия и сворачивания окна.

Перекрывающееся окно может принадлежать другому окну. Если окно-владелец сворачивается в пиктограмму, то подчиненные ему окна становятся невидимыми. При уничтожении окна автоматически уничтожаются подчиненные ему окна. Подчиненные окна всегда располагаются над поверхностью окна владельца, загораживая его.

Временные окна обычно используют для вывода сообщений пользователю и остаются на экране непродолжительное время. Базовый стиль временного окна описан константой WS\_POPUP. Такое окно по умолчанию не имеет заголовка. Чаще временное окно описывают константой WS\_POPUPWINDOW. Для добавления к временному окну системного меню и заголовка стиль WS\_POPUPWINDOW комбинируют со стилем WS\_CAPTION. Во всем остальном временные окна – это специальный вид перекрывающихся окон.

Дочерние окна используют для создания элементов управления. Определяемые системой классы элементов управления представляют собой дочерние окна. Базовый стиль дочерних окон описан константой WS\_CHILD. Этот стиль полностью совпадает со стилем WS\_CHILDWINDOW. Дочерние окна не имеют кнопок минимизации и максимального увеличения размера, но всегда имеют окно-родителя. Они «прилипают» к поверхности родителя, перемещаются с ним и не могут выйти за пределы родительского окна.

1. Описание программ
2. 1. Задание №1

Области для создания окна и полоски для вывода результата задаются кодом:

|  |
| --- |
| hwnd = CreateWindowEx(WS\_EX\_CLIENTEDGE, "WindowClass", "Калькулятор", WS\_VISIBLE | WS\_OVERLAPPEDWINDOW & ~(WS\_MAXIMIZEBOX | WS\_MINIMIZEBOX | WS\_THICKFRAME),  CW\_USEDEFAULT, /\* x \*/  CW\_USEDEFAULT, /\* y \*/  170, /\* width \*/  260, /\* height \*/  NULL, NULL, hInstance, NULL);  Edit = CreateWindowEx(WS\_EX\_CLIENTEDGE, "Edit", "", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | ES\_NUMBER,  10, /\* x \*/  10, /\* y \*/  139, /\* width \*/  30, /\* height \*/  hwnd,  NULL,  hInstance,  NULL); |

Листинг 1. Окно и полоска вывода

Переменная hwnd создает основное окно, а переменная Edit создает полосу вывода чисел и является дочернее к основному окну, аналогично для функциональных кнопок.

Расчёт значений производится, посредством функции switch.

|  |
| --- |
| case Multiply: {  Sign = Multiply;  for (int i = 0; i <= strlen(FirstOperand); i++)  SecondOperand[i] = FirstOperand[i];  FirstOperand[0] = 0;  SetWindowText(Edit, "");  break;  }  case Divided: {  Sign = Divided;  for (int i = 0; i <= strlen(FirstOperand); i++)  SecondOperand[i] = FirstOperand[i];  FirstOperand[0] = 0;  SetWindowText(Edit, "");  break;  } |

Листинг 2. Пример расчёта функций

Здесь значение переменных складывается между собой, а при вводе числа механизм прост, число введенное первым перезаписывается в элемент SecondOperand, для того, чтобы второе число могло занять его место.

Отрисовка результата осуществляется посредством функции

SetWindowText(Edit, FirstOperand), где Edit, это дочернее окно, в которое передаем число, а FirstOperand строка, которую передаем.

3. 1. Задание №2

Регистрация окна.

|  |
| --- |
| int RegClass(void)  {  WNDCLASS wc;  memset(&wc, 0, sizeof(WNDCLASS));  wc.style = CS\_VREDRAW | CS\_HREDRAW;  wc.lpfnWndProc = WndProc;  wc.hInstance = hInstance;  wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wc.hCursor = LoadCursor(NULL, IDC\_ARROW);  wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);  wc.lpszMenuName = NULL;  wc.lpszClassName = "MyWindowClass";  return RegisterClass(&wc);  } |

Листинг 3. Регистрация окна

Создание класса окна, и задание ему свойств.

Само создание окна, в main коде

|  |
| --- |
| hwnd\_main = CreateWindow("MyWindowClass", "Светофор",  WS\_OVERLAPPEDWINDOW,  0, 0, 250, 450, NULL, NULL, hInstance, 0);  if (!hwnd\_main) return 0;  ShowWindow(hwnd\_main, SW\_SHOW);  UpdateWindow(hwnd\_main); |

Листинг 4. Создание окна

Обработка нажатия на нужный круг.

|  |
| --- |
| case WM\_LBUTTONDOWN: {  xPos = LOWORD(lParam);  yPos = HIWORD(lParam);  if (xPos > 70 && xPos < 140 && yPos>280 && yPos < 350) {  HDC hdc4 = GetDC(hwnd\_main);  HBRUSH brush = CreateSolidBrush(RGB(100, 255, 0));  SelectObject(hdc4, brush);  Ellipse(hdc4, 70, 280, 140, 350);  DeleteObject(brush);  //-------------------------  HDC hdc = GetDC(hwnd\_main);  HBRUSH brush2 = (HBRUSH)GetStockObject(GRAY\_BRUSH);  SetROP2(hdc, R2\_COPYPEN);  HGDIOBJ obj = SelectObject(hdc, brush2);  SelectObject(hdc, obj);  Ellipse(hdc, 70, 70, 140, 140);  ReleaseDC(hwnd, hdc);  //----------------------------  HDC hdc2 = GetDC(hwnd\_main);  SetROP2(hdc2, R2\_COPYPEN);  HGDIOBJ obj2 = SelectObject(hdc2, brush2);  SelectObject(hdc2, obj2);  Ellipse(hdc2, 70, 175, 140, 245);  DeleteObject(brush2);  ReleaseDC(hwnd, hdc2);  } |

Листинг 5. Обработка нажатий на круг

При поступлении сообщения WM\_LBUTTONDOWN, формируются координаты x и y точки нажатия, затем через условие они передаются в нужный процесс и задают необходимые параметры цвета и отрисовка объекта этим цветом на окне.

1. Контрольные примеры работы программ


5. 1. Задание №1

Запустим приложение.



Рис. 1. Окно после запуска программы

Вычислим значение 15+35.

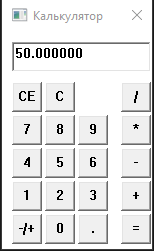


Рис. 2. Окно с выведенным результатом

Как видно из снимков экрана выше, калькулятор в польностью рабочем состоянии.

4. 2. Задание №2

Запустим приложение.

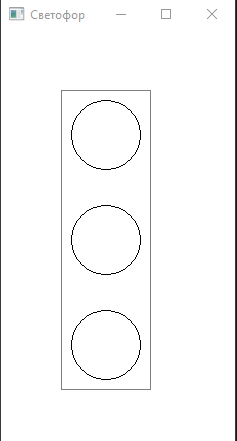


Рис. 3. Окно после запуска

Выберем первый круг, он должен окраситься в красный цвет.

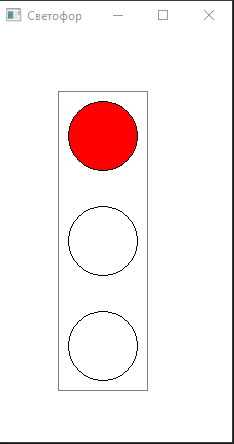


Рис. 4. Выбор первого круга

Аналогично для двух оставшихся объектов.



Список литературы

1. Белов, Е. А. Программирование приложений Windows. API графического интерфейса пользователя: учеб. пособие / Е. А. Белов. – Брянск: БГТУ, 2010. – 350 с.
2. Рихтер, Дж. Windows для профессионалов: создание эффективных Win32-приложений (с CD-ROM) / Дж. Рихтер. – М.: Русская редакция, 2000. – 752 с.
3. Рихтер, Дж. WINDOWS для профессионалов. Создание эффективных WIN32-приложений с учетом специфики 64-разрядной версии WINDOWS: [перевод] / Дж. Рихтер – 4-е изд. – СПб. [и др.]: Питер, Русская редакция, 2001. – 722 с.
4. Инструменты разработки и языки программирования [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/. – Загл. с экрана. – Яз. рус.
5. RSDN [Электронный ресурс]. – Режим доступа: http://www.rsdn.ru/. – Загл. с экрана. – Яз. рус.