**Системное программирование**

**Отчет по ЛР1**

**Фролов Иван**

**А-05-22**

Краткое описание функций и параметров, а также используемых событий:

1. Получить координаты окна (прямоугольник) GetWindowRect;

Функция GetWindowRect используется для получения координат прямоугольника, описывающего окно. Эти координаты представляют собой область окна относительно экрана.

**Синтаксис функции:**

BOOL GetWindowRect(

HWND hWnd, // дескриптор окна

LPRECT lpRect // указатель на структуру RECT, в которую будут записаны координаты

);

**Параметры:**

* **hWnd**: Дескриптор окна, для которого вы хотите получить координаты. Этот дескриптор должен быть валидным.
* **lpRect**: Указатель на структуру RECT, в которой будут храниться координаты окна. Структура RECT содержит четыре целых числа, представляющие левый верхний и правый нижний углы прямоугольника:
  + left: Координата левого края окна.
  + top: Координата верхнего края окна.
  + right: Координата правого края окна.
  + bottom: Координата нижнего края окна.

1. Установить координаты окна SetWindowPos

Функция SetWindowPos используется для установки нового положения и размера окна. Она также может изменить порядок окна в Z-порядке, например, чтобы окно находилось поверх всех остальных окон.

**Синтаксис функции:**

BOOL SetWindowPos(

HWND hWnd, // Дескриптор окна

HWND hWndInsertAfter, // Дескриптор окна для изменения порядка Z

int X, // Новая позиция окна (левый верхний угол)

int Y, // Новая позиция окна (левый верхний угол)

int cx, // Новая ширина окна

int cy, // Новая высота окна

UINT uFlags // Флаги для определения изменяемых параметров

);

**Параметры:**

* **hWnd**: Дескриптор окна, которое нужно переместить или изменить его размер.
* **hWndInsertAfter**: Дескриптор окна, после которого должно быть размещено hWnd. Используйте HWND\_TOP для размещения окна поверх всех окон, HWND\_BOTTOM для размещения внизу, или другие предопределенные дескрипторы для изменения порядка Z.
* **X**: Новое значение для координаты X (левый верхний угол) окна.
* **Y**: Новое значение для координаты Y (левый верхний угол) окна.
* **cx**: Новая ширина окна.
* **cy**: Новая высота окна.
* **uFlags**: Флаги, определяющие, какие параметры следует изменить. Возможные флаги:
  + SWP\_NOSIZE — Не изменять размер окна.
  + SWP\_NOMOVE — Не изменять положение окна.
  + SWP\_NOACTIVATE — Не активировать окно.
  + SWP\_SHOWWINDOW — Показать окно.
  + SWP\_HIDEWINDOW — Скрыть окно.

1. Переместить окно MoveWindow

Функция MoveWindow в Windows API используется для перемещения и изменения размера окна. Она позволяет указать новое положение окна и его размеры, а также обновить окно сразу после изменения.

**Синтаксис функции:**

BOOL MoveWindow(

HWND hWnd, // Дескриптор окна

int X, // Новая позиция окна (левый верхний угол)

int Y, // Новая позиция окна (левый верхний угол)

int nWidth, // Новая ширина окна

int nHeight, // Новая высота окна

BOOL bRepaint // Флаг для обновления окна

);

**Параметры:**

* **hWnd**: Дескриптор окна, которое нужно переместить и изменить размер.
* **X**: Новое значение для координаты X (левый верхний угол) окна.
* **Y**: Новое значение для координаты Y (левый верхний угол) окна.
* **nWidth**: Новая ширина окна.
* **nHeight**: Новая высота окна.
* **bRepaint**: Флаг, указывающий, нужно ли обновлять окно после изменения его размера и положения. Если TRUE, окно будет перерисовано сразу после перемещения и изменения размера; если FALSE, окно не будет перерисовано до следующего вызова функции, который потребует перерисовки.

1. Вывод сообщения MessageBox

Функция MessageBox в Windows API используется для отображения модального окна сообщения пользователю. Это стандартный способ уведомления пользователя о какой-либо информации, ошибке, предупреждении или запросе подтверждения.

**Синтаксис функции:**

cpp

Копировать код

int MessageBox(

HWND hWnd, // Дескриптор родительского окна

LPCSTR lpText, // Текст сообщения

LPCSTR lpCaption, // Заголовок окна сообщения

UINT uType // Флаги определения типа и кнопок

);

**Параметры:**

* **hWnd**: Дескриптор родительского окна, под которым будет отображаться окно сообщения. Если значение NULL, то окно сообщения будет отображаться без родительского окна.
* **lpText**: Указатель на строку текста сообщения, которую вы хотите отобразить в окне сообщения.
* **lpCaption**: Указатель на строку, которая будет использоваться в качестве заголовка окна сообщения. Если значение NULL, то заголовок не будет отображаться.
* **uType**: Флаги, которые определяют, какие кнопки будут отображаться, а также стиль окна сообщения. Эти флаги включают тип сообщения и варианты кнопок.

**Основные флаги для uType:**

* **MB\_OK**: Отображает кнопку "OK".
* **MB\_OKCANCEL**: Отображает кнопки "OK" и "Cancel".
* **MB\_YESNO**: Отображает кнопки "Yes" и "No".
* **MB\_ICONEXCLAMATION**: Отображает значок восклицательного знака.
* **MB\_ICONINFORMATION**: Отображает значок информации.
* **MB\_ICONQUESTION**: Отображает значок вопроса.
* **MB\_ICONSTOP**: Отображает значок ошибки.

**Возвращаемое значение:**

* Функция возвращает идентификатор кнопки, на которую нажал пользователь. Например, IDOK, IDCANCEL, IDYES, IDNO и так далее, в зависимости от выбранной кнопки.

1. Создать окно CreateWindow

Функция CreateWindow в Windows API используется для создания нового окна. Она позволяет задать основные параметры окна, такие как класс окна, его заголовок, стиль и размеры.

**Синтаксис функции:**

HWND CreateWindow(

LPCTSTR lpClassName, // Имя класса окна

LPCTSTR lpWindowName, // Заголовок окна

DWORD dwStyle, // Стиль окна

int x, // Координата X (левый верхний угол) окна

int y, // Координата Y (левый верхний угол) окна

int nWidth, // Ширина окна

int nHeight, // Высота окна

HWND hWndParent, // Дескриптор родительского окна

HMENU hMenu, // Дескриптор меню или идентификатор управления

HINSTANCE hInstance, // Дескриптор экземпляра приложения

LPVOID lpParam // Дополнительные параметры

);

**Параметры:**

* **lpClassName**: Имя класса окна, зарегистрированного ранее с помощью RegisterClass или RegisterClassEx. Это имя должно совпадать с именем класса, которое было использовано при регистрации окна.
* **lpWindowName**: Заголовок окна, который будет отображаться на панели заголовка окна. Если значение NULL, то заголовок не будет отображаться.
* **dwStyle**: Стиль окна, который определяет его внешний вид и поведение. Например, WS\_OVERLAPPEDWINDOW, WS\_CHILD, WS\_VISIBLE, и т. д.
* **x**: Координата X для левого верхнего угла окна относительно родительского окна или экрана, если родительское окно равно NULL.
* **y**: Координата Y для левого верхнего угла окна относительно родительского окна или экрана, если родительское окно равно NULL.
* **nWidth**: Ширина окна в пикселях.
* **nHeight**: Высота окна в пикселях.
* **hWndParent**: Дескриптор родительского окна. Если окно не имеет родителя (например, это основное окно приложения), это значение должно быть NULL.
* **hMenu**: Дескриптор меню или идентификатор управления. Если окно имеет меню, укажите дескриптор меню, если это окно управления, укажите идентификатор управления. Если у окна нет меню или управления, это значение должно быть NULL.
* **hInstance**: Дескриптор экземпляра приложения, который был возвращен функцией GetModuleHandle или передан в WinMain.
* **lpParam**: Указатель на дополнительные параметры, которые могут быть переданы функции CreateWindow. Обычно это значение NULL.

1. Установка идентификатора: SetWindowLong

Функция SetWindowLong используется в Windows API для установки различных длинных параметров окна, таких как указатель на оконную процедуру, стили и дополнительные параметры окна. Она позволяет изменять параметры окна в зависимости от типа устанавливаемого значения.

**Синтаксис функции:**

LONG SetWindowLong(

HWND hWnd, // Дескриптор окна

int nIndex, // Индекс параметра окна для изменения

LONG dwNewLong // Новое значение для установки

);

**Параметры:**

* **hWnd**: Дескриптор окна, для которого устанавливаются параметры.
* **nIndex**: Индекс параметра окна, который нужно изменить. Возможные значения включают:
  + GWLP\_WNDPROC — указатель на оконную процедуру.
  + GWLP\_HINSTANCE — дескриптор экземпляра приложения.
  + GWLP\_ID — идентификатор управляющего элемента.
  + GWLP\_USERDATA — дополнительное пользовательское значение.
  + GWLP\_STYLE — стили окна.
  + GWLP\_EXSTYLE — расширенные стили окна.
* **dwNewLong**: Новое значение для установки.

**Возвращаемое значение:**

* **Возвращает старое значение** для указанного индекса параметра. Если функция выполняется успешно, возвращаемое значение будет предыдущее значение, которое было установлено для данного параметра. Если функция не удается, возвращается значение 0 и для получения дополнительной информации можно вызвать функцию GetLastError.

1. Функции GetWindowText и SetWindowText

**Функция GetWindowText**

Функция GetWindowText получает текст, который отображается в указанном окне или элементе управления.

**Синтаксис:**

int GetWindowText(

HWND hWnd, // Дескриптор окна или элемента управления

LPTSTR lpString, // Буфер для хранения текста

int nMaxCount // Размер буфера, в символах

);

**Параметры:**

* **hWnd**: Дескриптор окна или элемента управления, из которого нужно получить текст.
* **lpString**: Указатель на буфер, в который будет записан текст. Буфер должен быть достаточно большим, чтобы вместить весь текст.
* **nMaxCount**: Размер буфера lpString в символах, включая завершающий нулевой символ.

**Функция SetWindowText**

Функция SetWindowText устанавливает текст для указанного окна или элемента управления.

**Синтаксис:**

BOOL SetWindowText(

HWND hWnd, // Дескриптор окна или элемента управления

LPCTSTR lpString // Указатель на строку, которая будет установлена

);

**Параметры:**

* **hWnd**: Дескриптор окна или элемента управления, для которого нужно установить текст.
* **lpString**: Указатель на строку, которая будет установлена в качестве нового текста.

1. Функции SetParent и GetParent

**Функция SetParent**

Функция SetParent изменяет родительское окно для указанного окна или элемента управления.

**Синтаксис:**

HWND SetParent(

HWND hWndChild, // Дескриптор дочернего окна или элемента управления

HWND hWndNewParent // Дескриптор нового родительского окна

);

**Параметры:**

* **hWndChild**: Дескриптор дочернего окна или элемента управления, для которого нужно изменить родителя.
* **hWndNewParent**: Дескриптор нового родительского окна. Если вы хотите удалить родителя, укажите значение NULL.

**Функция GetParent**

Функция GetParent получает дескриптор родительского окна для указанного дочернего окна или элемента управления.

**Синтаксис:**

HWND GetParent(

HWND hWnd // Дескриптор дочернего окна или элемента управления

);

**Параметры:**

* **hWnd**: Дескриптор дочернего окна или элемента управления, для которого нужно получить дескриптор родителя.

1. Функция Sleep (пауза)

Функция Sleep в Windows API используется для приостановки выполнения текущего потока на определенное количество миллисекунд. Это позволяет приложению временно "уснуть", освобождая ресурсы процессора, которые могут быть использованы другими задачами.

**Синтаксис функции:**

void Sleep(

DWORD dwMilliseconds // Количество миллисекунд, на которое нужно приостановить выполнение

);

**Параметры:**

* **dwMilliseconds**: Количество миллисекунд, на которое нужно приостановить выполнение текущего потока. Значение должно быть положительным и может быть 0 (в этом случае функция просто позволяет другим потокам и приложениям получить доступ к процессору, но не останавливает выполнение).

1. Функции для работы с регионами:

**CombineRgn**

Комбинирует два региона с помощью одной из указанных логических операций: объединение, пересечение, исключение или разность.

**Синтаксис:**

int CombineRgn(

HRGN hRgnDest, // Дескриптор целевого региона

HRGN hRgnSrc1, // Дескриптор первого исходного региона

HRGN hRgnSrc2, // Дескриптор второго исходного региона

int nCombineMode // Режим комбинации (RGN\_AND, RGN\_OR, RGN\_XOR, RGN\_DIFF)

);

**2. CreateEllipticRgn**

Создает регион в виде эллипса, который может быть также окружностью.

**Синтаксис:**

HRGN CreateEllipticRgn(

int xLeft, // X-координата левого края прямоугольника

int yTop, // Y-координата верхнего края прямоугольника

int xRight, // X-координата правого края прямоугольника

int yBottom // Y-координата нижнего края прямоугольника

);

**3. CreatePolygonRgn**

Создает регион в виде многоугольника, заданного массивом точек.

**Синтаксис:**

HRGN CreatePolygonRgn(

const POINT\* lpPoints, // Указатель на массив точек

int nCount, // Количество точек

int nFillMode // Режим заливки (ALTERNATE или WINDING)

);

**4. CreateRectRgn**

Создает регион в виде прямоугольника.

**Синтаксис:**

HRGN CreateRectRgn(

int xLeft, // X-координата левого края прямоугольника

int yTop, // Y-координата верхнего края прямоугольника

int xRight, // X-координата правого края прямоугольника

int yBottom // Y-координата нижнего края прямоугольника

);

**5. CreateRoundRectRgn**

Создает регион в виде прямоугольника со скругленными углами.

**Синтаксис:**

HRGN CreateRoundRectRgn(

int xLeft, // X-координата левого края прямоугольника

int yTop, // Y-координата верхнего края прямоугольника

int xRight, // X-координата правого края прямоугольника

int yBottom, // Y-координата нижнего края прямоугольника

int nWidthEllipse, // Ширина эллипса, который образует скругление

int nHeightEllipse // Высота эллипса, который образует скругление

);

**6. SetWindowRgn**

Прикрепляет созданный регион к указанному окну, изменяя его форму в соответствии с регионом.

**Синтаксис:**

int SetWindowRgn(

HWND hWnd, // Дескриптор окна

HRGN hRgn, // Дескриптор региона

BOOL bRedraw // Флаг перерисовки окна

)

1. Описание событий мыши с параметрами, а также событий WM\_CREATE, WM\_COMMAND, WM\_CLOSE, WM\_DESTROY и WM\_QUIT.

**События мыши**

1. **WM\_LBUTTONDOWN**:
   * **Описание**: Левый кнопка мыши нажата.
   * **Параметры**:
     + wParam: Флаги состояния кнопок мыши и клавиш.
     + lParam: Координаты указателя мыши (x и y).
2. **WM\_LBUTTONUP**:
   * **Описание**: Левый кнопка мыши отпущена.
   * **Параметры**:
     + wParam: Флаги состояния кнопок мыши и клавиш.
     + lParam: Координаты указателя мыши (x и y).
3. **WM\_RBUTTONDOWN**:
   * **Описание**: Правая кнопка мыши нажата.
   * **Параметры**:
     + wParam: Флаги состояния кнопок мыши и клавиш.
     + lParam: Координаты указателя мыши (x и y).
4. **WM\_RBUTTONUP**:
   * **Описание**: Правая кнопка мыши отпущена.
   * **Параметры**:
     + wParam: Флаги состояния кнопок мыши и клавиш.
     + lParam: Координаты указателя мыши (x и y).
5. **WM\_MOUSEMOVE**:
   * **Описание**: Указатель мыши перемещен.
   * **Параметры**:
     + wParam: Флаги состояния кнопок мыши и клавиш.
     + lParam: Координаты указателя мыши (x и y).

**Сообщения Windows**

Параметром wParam может быть комбинация следующих значений:

MK\_CONTROL Нажата клавиша CTRL.

MK\_LBUTTON Нажата левая кнопка мыши.

MK\_MBUTTON Нажата средняя кнопка мыши.

MK\_RBUTTON Нажата правая кнопка мыши.

MK\_SHIFT Нажата клавиша SHIFT.

1. **WM\_CREATE**:
   * **Описание**: Отправляется окну при его создании.
   * **Параметры**:
     + wParam: Не используется.
     + lParam: Указатель на структуру CREATESTRUCT, содержащую информацию о создании окна.
2. **WM\_COMMAND**:
   * **Описание**: Отправляется окну, когда пользователь взаимодействует с элементами управления (меню, кнопки и т.д.).
   * **Параметры**:
     + wParam: Идентификатор команды или управляющего элемента.
     + lParam: Дескриптор управляющего элемента (если применимо).
3. **WM\_CLOSE**:
   * **Описание**: Отправляется окну при запросе на его закрытие (например, при нажатии на кнопку закрытия окна).
   * **Параметры**:
     + wParam и lParam: Не используются.
4. **WM\_DESTROY**:
   * **Описание**: Отправляется окну перед его уничтожением.
   * **Параметры**:
     + wParam и lParam: Не используются.
5. **WM\_QUIT**:
   * **Описание**: Отправляется потоку сообщений для завершения выполнения приложения.
   * **Параметры**:
     + wParam: Код завершения приложения.
     + lParam: Не используется.

Код работы