Модуль №3

Занятие №6

Версия 1.0.1

План занятия:

1. Повторение пройденного материала.
2. Общий элемент управления «строка состояния» (Status Bar).
3. Сообщения строки состояния.
4. Практическая часть.
5. Подведение итогов.
6. Домашнее задание.

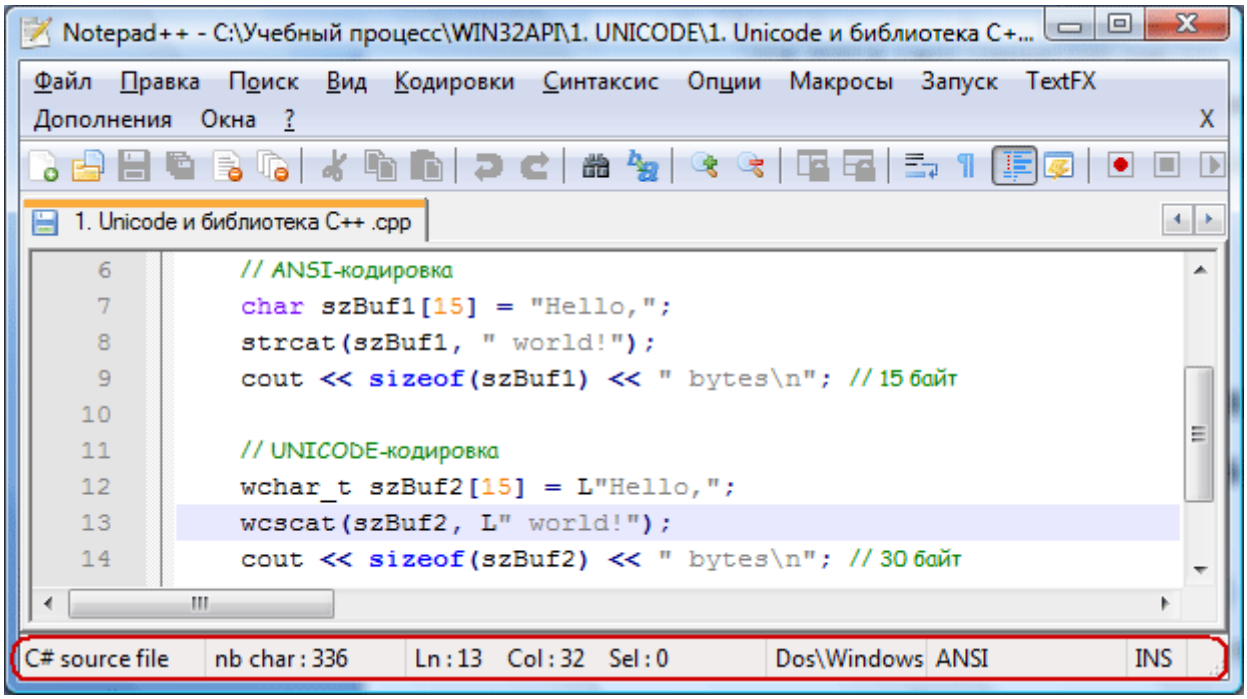
1. Повторение пройденного материала

Данное занятие необходимо начать с краткого повторения материала предыдущего занятия. При общении со слушателями можно использовать следующие контрольные вопросы:

1) Для чего применяется элемент управления **Slider Control**?
2) Каким образом можно установить диапазон ползунка (нижнюю и верхнюю границу)?
3) С помощью какого сообщения можно получить текущую позицию ползунка?
4) С помощью какого сообщения можно установить новую позицию ползунка?
5) Каким сообщением ползунок уведомляет родительское окно (диалог) о действиях пользователя?
6) Как программно создать **Slider Control**?
7) Какие два элемента управления образуют поле с прокруткой?

8) Каким образом можно установить минимальную и максимальную позиции счётчика?
9) С помощью какого сообщения можно получить текущую позицию счётчика?
10) С помощью какого сообщения можно установить новую позицию счётчика?
11) Как установить «приятеля» для счётчика?
12) С помощью какого сообщения можно установить правило, по которому будет осуществляться приращение счетчика?
13) Какое сообщение **Spin Control** посылает своему родительскому окну при нажатии одной из стрелок? Какая дополнительная информация приходит с этим сообщением?
14) Каким образом можно программно создать **Spin Control**?

2. Общий элемент управления «строка состояния» (Status Bar)



Повторив материал предыдущего занятия, ознакомить слушателей с ещё одним общим элементом управления – строкой состояния, которая часто применяется в оформлении главного окна приложения. Отметить, что **строка состояния** (**Status Bar**) — это окно, обычно располагающееся в нижней части главного окна приложения, которое предназначено для вывода информации о текущем состоянии программы, о выполняемых операциях и режимах. Практически любой редактор текста (например, **Notepad++)** обладает строкой состояния, в которой, в частности, отображается текущая позиция мигающей каретки.

Для создания строки состояния можно воспользоваться функцией API **CreateStatusWindow:**

```
HWND CreateStatusWindow(
    LONG style, // стили для строки состояния
    LPCTSTR lpszText, // указатель на строку, содержащую текст для первой
      // секции строки состояния
    HWND hwndParent, // дескриптор родительского окна
    UINT wID // идентификатор строки состояния
);
```

Поскольку строка состояния является окном, то она также может быть создана с помощью функции **CreateWindowEx**. В этом случае во втором параметре функции передается имя предопределенного оконного класса – **STATUSCLASSNAME**.

```
HWND hStatusBar = CreateWindowEx( 0, STATUSCLASSNAME, 0,
      WS_CHILD | WS_VISIBLE | SBT_TOOLTIPS | CCS_BOTTOM | SBARS_SIZEGRIP,
      0, 0, 0, 0, hDialog, HMENU(WM_USER), GetModuleHandle(NULL), 0);
```

Обратить внимание слушателей, что строка состояния может работать в одном из двух режимов: в стандартном режиме либо в простом режиме.

При работе в **стандартном режиме** строка состояния разбивается на несколько секций, в каждую из которых выводится отдельная строка

текста или иконка. При работе в **простом режиме** строка состояния реализована как единый элемент и отображает только одну секцию.

Следует отметить, что по умолчанию окно строки состояния имеет стили **CCS_BOTTOM** и **SBARS_SIZEGRIP**. Стиль **CCS_BOTTOM** обозначает, что элемент управления располагается внизу клиентской области родительского окна и имеет ширину, равную ширине родительского окна. Стиль **SBARS_SIZEGRIP** задает наличие «манипулятора размера» в правом углу строки состояния. Этот декоративный элемент создает область, за которую можно ухватиться при изменении размера окна приложения.

Если при создании строки состояния указать стиль **SBARS_TOOLTIPS** – это позволит включить режим всплывающих подсказок для строки состояния. При этом важно отметить, что подсказка появляется в одном из двух случаев:

- в секции присутствует только иконка;
- текст полностью не помещается в секции.

3. Сообщения строки состояния

Как было отмечено ранее, строка состояния может работать в простом или стандартном режиме. Переключение между режимами осуществляется посылкой сообщения **SB_SIMPLE** строке состояния. При этом если в **wParam** указывается значение **TRUE**, то устанавливается простой режим, в противном случае устанавливается стандартный режим. Параметр **lParam** с данным сообщением не используется (должен быть 0). Например:

```
// Установка простого режима
SendMessage(hwndStatusBar, SB_SIMPLE, TRUE, 0);
```

Если строка состояния используется в стандартном режиме, то её необходимо разделить на отдельные секции при помощи сообщения **SB_SETPARTS**. При отправке этого сообщения в **wParam** указывается количество секций строки состояния, а в **lParam** указывается адрес целочисленного массива, каждый элемент которого определяет позицию (в клиентских координатах) правой границы соответствующей секции. Если элемент массива равен -1, то границей соответствующей секции считается правая граница строки состояния.

Для отображения текста в строке состояния, ей посылается сообщение **SB_SETTEXT**. С этим сообщением в **wParam** передаётся номер секции в строке состояния (отсчёт ведётся с нуля) и графический стиль секции, а в **lParam –** указатель на строку, содержащую текст.

Графические стили секции могут иметь одно из следующих значений, приведенных в таблице:

Стиль	Описание
0 (значение по умолчанию)	Секция рисуется с вдавленной рамкой
SBT_NOBORDERS	Секция рисуется без рамки
SBT_POPOUT	Секция рисуется с выпуклой рамкой
SBT_OWNERDRAW	За прорисовку секции отвечает родительское окно

Обычно в каждую секцию строки состояния выводится отдельное текстовое сообщение. Однако можно разместить в секции элемент управления, например индикатор процесса. При таком размещении необходимо знать клиентские координаты этой секции, которые могут быть получены при помощи отправки сообщения **SB_GETRECT**. При посылке этого сообщения в **wParam** указывается номер секции в строке состояния, а в **lParam –** адрес структуры типа **RECT**, принимающей координаты секции.

Следующий фрагмент кода демонстрирует размещение индикатора в строке состояния:

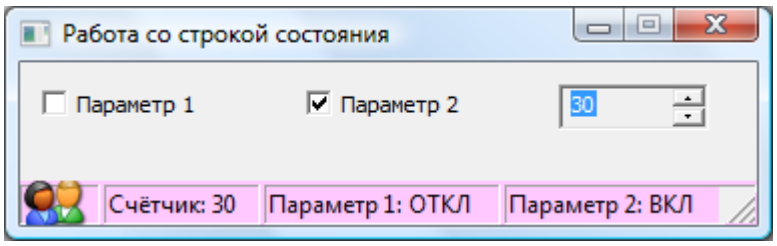
```
RECT r;
SendMessage(hStatus, SB_GETRECT, 1, (LPARAM)&r);
```

```
hProgressBar = CreateWindowEx(0, PROGRESS_CLASS, NULL, WS_CHILD | WS_VISIBLE,
                  r.left + 3, r.top + 3, r.right - r.left, r.bottom - r.top,
                  hStatusBar, NULL, GetModuleHandle(NULL), NULL);
```

В следующей таблице приводятся ещё несколько сообщений, часто используемых для управления строкой состояния.

Код сообщения	wParam	lParam	Описание
SB_SETTIPTEXT	iPart	(LPCTSTR) lpszTooltip	Устанавливает текст всплывающей подсказки для заданной секции
SB_SETICON	iPart	(HICON) hIcon	Устанавливает иконку на заданную секцию
SB_SETBKCOLOR	0	(COLORREF) clrBk	Устанавливает цвет фона для строки состояния
SB_GETTEXT	iPart	(LPSTR) szText	Получает текст из выбранной секции

В качестве примера, демонстрирующего работу строки состояния, привести слушателям следующее приложение, в котором строка состояния используется для отображения состояния элементов управления в диалоге.




```
                                    // header.h

#pragma once
#include <windows.h>
#include <windowsX.h>
#include <tchar.h>
#include <commctrl.h>
#include "resource.h"

#pragma comment(lib,"comctl32")


                                  // StatusBarDlg.h

#pragma once
#include "header.h"
```

```
class CStatusBarDlg
{
public:
	CStatusBarDlg(void);

public:
	static BOOL CALLBACK DlgProc(HWND hWnd, UINT mes, WPARAM wp, LPARAM lp);
	static CStatusBarDlg* ptr;
	BOOL Cls_OnInitDialog(HWND hwnd, HWND hwndFocus, LPARAM lParam);
	void Cls_OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify);
	void Cls_OnClose(HWND hwnd);
	void Cls_OnVScroll(HWND hwnd, HWND hwndCtl, UINT code, int pos);
	void Cls_OnSize(HWND hwnd, UINT state, int cx, int cy);
	HWND hDialog, hSpin, hCheck1, hCheck2, hStatus, hEdit;
	HICON hIcon;
};

                                // StatusBarDlg.cpp

#include "StatusBarDlg.h"

CStatusBarDlg* CStatusBarDlg::ptr = NULL;

CStatusBarDlg::CStatusBarDlg(void)
{
	ptr = this;
}

void CStatusBarDlg::Cls_OnClose(HWND hwnd)
{
	EndDialog(hwnd, 0);
}

BOOL CStatusBarDlg::Cls_OnInitDialog(HWND hwnd, HWND hwndFocus,
                                     LPARAM lParam)
{
	hDialog = hwnd;
	// Получим дескрипторы элементов управления
	hSpin = GetDlgItem(hDialog, IDC_SPIN1);
	hEdit = GetDlgItem(hDialog, IDC_EDIT2);
	hCheck1 = GetDlgItem(hDialog, IDC_CHECK1);
	hCheck2 = GetDlgItem(hDialog, IDC_CHECK2);
	// Установим диапазон счётчика
	SendMessage(hSpin, UDM_SETRANGE32, 0, 100);
	// Установим приятеля для счётчика
	SendMessage(hSpin, UDM_SETBUDDY, WPARAM(hEdit), 0);
	SetWindowText(hEdit, TEXT("0"));
	// Создадим строку состояния
	hStatus = CreateStatusWindow(WS_CHILD | WS_VISIBLE | CCS_BOTTOM |
	            SBARS_TOOLTIPS | SBARS_SIZEGRIP, 0, hDialog, WM_USER);
	// Разделим строку состояния на отдельные секции
	int parts[4] = {40, 120, 240, -1};
	SendMessage(hStatus, SB_SETPARTS, 4, (LPARAM)parts);

	// Для каждой секции установим текст и всплывающую подсказку
	SendMessage(hStatus, SB_SETTEXT, 1, (LPARAM) TEXT("Счетчик: 0"));
	SendMessage(hStatus, SB_SETTIPTEXT, 1, (LPARAM) TEXT("Счетчик: 0"));
	SendMessage(hStatus, SB_SETTEXT, 2, (LPARAM) TEXT("Параметр 1: ОТКЛ"));
```

```
	SendMessage(hStatus, SB_SETTIPTEXT, 2,
			(LPARAM) TEXT("Параметр 1: ОТКЛ"));

	SendMessage(hStatus, SB_SETTEXT, 3, (LPARAM) TEXT("Параметр 2: ОТКЛ"));
	SendMessage(hStatus, SB_SETTIPTEXT, 3,
			(LPARAM) TEXT("Параметр 2: ОТКЛ"));

	// Загрузим иконку из ресурсов
	hIcon = LoadIcon(GetModuleHandle(NULL), MAKEINTRESOURCE(IDI_ICON1));
	// Установим иконку в первую секцию строки состояния
	SendMessage(hStatus, SB_SETICON, 0, (LPARAM) hIcon);

	// Установим цвет фона строки состояния
	SendMessage(hStatus, SB_SETBKCOLOR, 0, (LPARAM) RGB(255, 200, 255));
	return TRUE;
}

void CStatusBarDlg::Cls_OnCommand(HWND hwnd, int id, HWND hwndCtl,
                                  UINT codeNotify)
{
	// отобразим статус флажков в строке состояния
	if(id == IDC_CHECK1)
	{
		LRESULT lResult = SendMessage(hCheck1, BM_GETCHECK, 0, 0);
		if(lResult == BST_CHECKED)
		{
			SendMessage(hStatus, SB_SETTEXT, 2,
					(LPARAM) TEXT("Параметр 1: ВКЛ"));
			SendMessage(hStatus, SB_SETTIPTEXT, 2, (LPARAM)
					TEXT("Параметр 1: ВКЛ"));
		}
		else
		{
			SendMessage(hStatus, SB_SETTEXT, 2,
					(LPARAM) TEXT("Параметр 1: ОТКЛ"));
			SendMessage(hStatus, SB_SETTIPTEXT, 2,
					(LPARAM) TEXT("Параметр 1: ОТКЛ"));
		}
	}

	if(id == IDC_CHECK2)
	{
		LRESULT lResult = SendMessage(hCheck2, BM_GETCHECK, 0, 0);
		if(lResult == BST_CHECKED)
		{
			SendMessage(hStatus, SB_SETTEXT, 3,
					(LPARAM) TEXT("Параметр 2: ВКЛ"));
			SendMessage(hStatus, SB_SETTIPTEXT, 3,
					(LPARAM) TEXT("Параметр 2: ВКЛ"));
		}
		else
		{
			SendMessage(hStatus, SB_SETTEXT, 3,
					(LPARAM) TEXT("Параметр 2: ОТКЛ"));
			SendMessage(hStatus, SB_SETTIPTEXT, 3,
					(LPARAM) TEXT("Параметр 2: ОТКЛ"));
		}
	}
}
```

```
void CStatusBarDlg::Cls_OnVScroll(HWND hwnd, HWND hwndCtl, UINT code,
                                  int pos)
{
      // обработчик сообщения нажатия на одну из стрелок счётчика
      TCHAR buf[30];
      wsprintf(buf, TEXT("Счётчик: %d"), pos);
      // отобразим состояние счётчика в строке состояния
      SendMessage(hStatus, SB_SETTEXT, 1,(LPARAM)buf);
      SendMessage(hStatus, SB_SETTIPTEXT, 1,(LPARAM)buf);
}

void CStatusBarDlg::Cls_OnSize(HWND hwnd, UINT state, int cx, int cy)
{
      // установим размер строки состояния,
      // равный ширине клиентской области главного окна
      SendMessage(hStatus, WM_SIZE, 0, 0);
}

BOOL CALLBACK CStatusBarDlg::DlgProc(HWND hwnd, UINT message, WPARAM wParam,
                                     LPARAM lParam)
{
      switch(message)
      {
            HANDLE_MSG(hwnd, WM_CLOSE, ptr->Cls_OnClose);
            HANDLE_MSG(hwnd, WM_INITDIALOG, ptr->Cls_OnInitDialog);
            HANDLE_MSG(hwnd, WM_COMMAND, ptr->Cls_OnCommand);
            HANDLE_MSG(hwnd, WM_VSCROLL, ptr->Cls_OnVScroll);
            HANDLE_MSG(hwnd, WM_SIZE, ptr->Cls_OnSize);
      }
      return FALSE;
}
```

```
                              // StatusBarDlg.cpp

#include "StatusBarDlg.h"

int WINAPI _tWinMain(HINSTANCE hInst, HINSTANCE hPrev, LPTSTR lpszCmdLine,
                     int nCmdShow)
{
      INITCOMMONCONTROLSEX icc = {sizeof(INITCOMMONCONTROLSEX)};
      icc.dwICC = ICC_WIN95_CLASSES;
      InitCommonControlsEx(&icc);
      CStatusBarDlg dlg;
      return DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG1), NULL,
                       CStatusBarDlg::DlgProc);
}
```

4. Практическая часть

Дополнить строкой состояния главное окно приложения «Программное создание статиков» из домашнего задания (см. Модуль 2 Занятие 1). При щелчке правой кнопкой мыши над поверхностью «стати-

ка» в строку состояния необходимо вывести информацию о «статике» (порядковый номер «статика», ширина и высота, а также координаты «статика» относительно родительского окна).

5. Подведение итогов

Подвести общие итоги занятия. Ещё раз подчеркнуть основные два способа создания строки состояния. Выделить режимы работы строки состояния. Перечислить сообщения, наиболее часто используемые для управления строкой состояния. Акцентировать внимание слушателей на наиболее тонких моментах изложенной темы.

6. Домашнее задание

Модифицировать интерфейс приложения «Пятнашки» следующим образом: дополнить главное окно приложения строкой состояния, разделённой на две секции. В первую секцию необходимо выводить время продолжения игры, а во вторую секцию следует поместить индикатор процесса, который будет отображать процесс собирания «пятнашек».