Тема1.
Введение в WinAPI

Литература

- Win32 API. Эффективная разработка приложений / Ю.А. Щупак. – СПб : Питер, 2007. – 572 с.
- Win32 API. Разработка приложений для Windows / Ю.А. Щупак. – СПб : Питер, 2008. – 592 с.
- Финогенов К. Г. Win32. Основы программирования. – М.: ДИАЛОГ-МИФИ, 2002.
- Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows. СПб: Питер, 2001. – 714 с.
- Петзолд Ч. Программирование для Windows 95; в двух томах: пер. с англ. — СПб.: BHV — Санкт-Петербург, 1997. — 368 с.

Преимущества ОС Windows

- графический интерфейс пользователя, "визуальный интерфейс" или "графическая оконная среда" – *GUI (Graphical User Interface)* ***WYSIWYG (What you see is what you get – что видите, то и получаете)***;
- многозадачность – *процесс (process)* и *поток (thread)* ;
- управление памятью – *библиотеки динамической компоновки DLL (dynamic link libraries)*;
- независимость от аппаратных средств – *графический интерфейс устройства (Graphics Device Interface, GDI)*.

Основные черты Windows-приложений

- стандартный оконный интерфейс со множеством элементов управления (кнопки, линейки, шкалы, списки и т. д.);
- элементы поддерживаются с помощью DLL, которые являются частью операционной системы;
- основаны на вызове функций Windows API.

Основные понятия

- *API – Application Programming Interface* (интерфейс прикладного программирования). API представляет собой совокупность функций (более 2000) и инструментов: структуры, более 700 сообщений, макросы и интерфейсы;
- *Win32 API* – это набор функций для создания программ, работающих под управлением Microsoft Windows (98/NT/2000/XP). Все функции этого набора являются 32-битными, что отражено в названии интерфейса.

Системные вызовы Windows

- *функции модуля KERNEL.DLL* – управление процессами, потоками, ресурсами, файлами и памятью;
- *функции модуля USER.DLL* – работа с пользовательским интерфейсом, например, с окнами, элементами управления, диалогами и сообщениями;
- *функции модуля GDI.DLL* – аппаратно-независимый графический вывод;
- *вспомогательные функции API* – для работы с электронной почтой (MAPI), модемами (TAPI), базами данных (ODBC) и др.

Архитектура, управляемая событиями

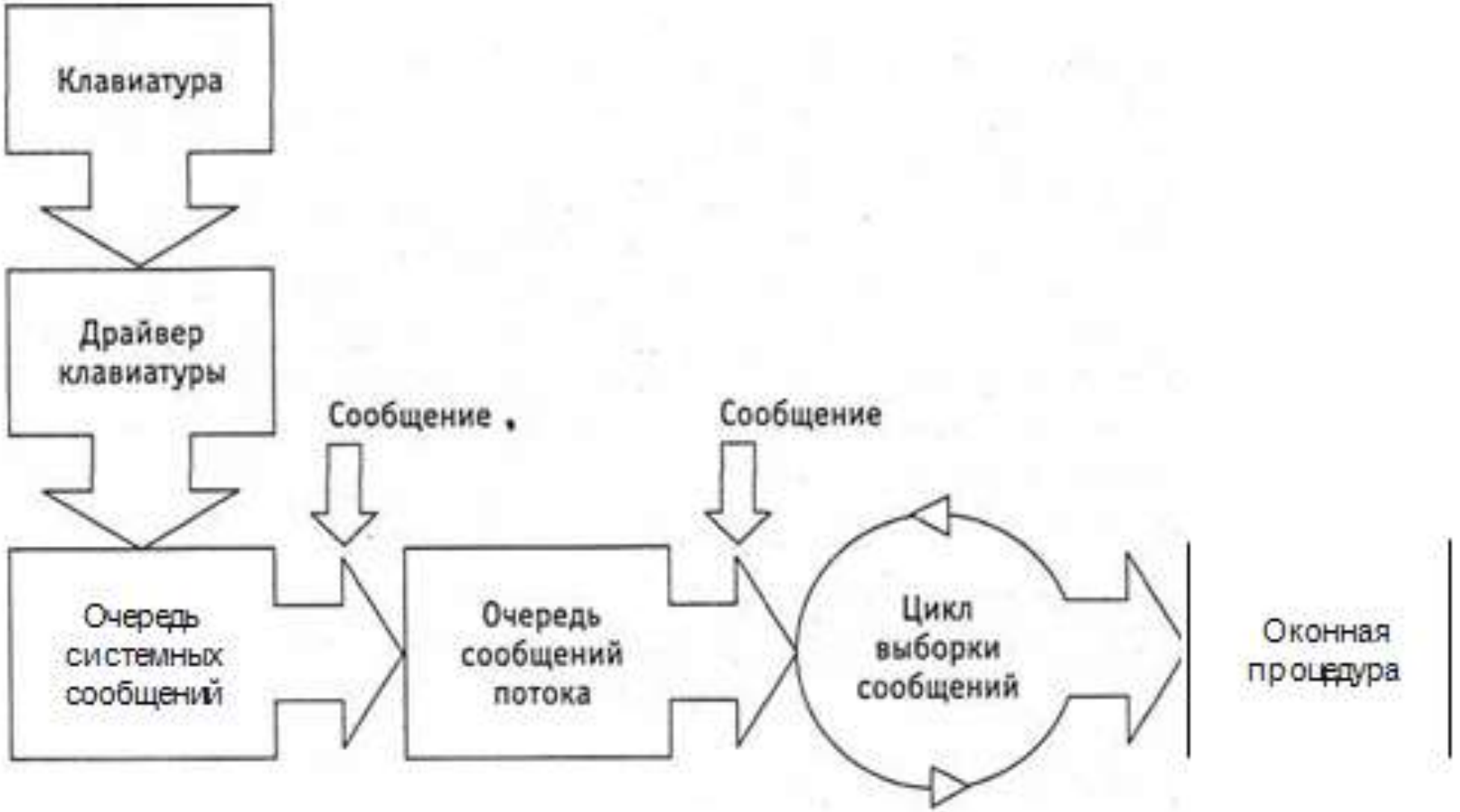
- Все Windows-приложения являются программами, управляемыми событиями (event-driven applications).
- В "глубине" ОС реализован механизм, преобразующий информацию от различных устройств ввода/вывода (*события*) в некоторую структуру данных – *сообщение,* которая содержит следующие элементы:
  - дескриптор окна, которому адресовано сообщение;
  - код (номер) сообщения;
  - дополнительную информацию, зависящую от кода сообщения.
- Windows –многозадачная ОС, основанная на *передаче сообщений.* Путь следования сообщений

аппаратное событие ->
системная очередь сообщений ->
очередь сообщений приложения

Архитектура, управляемая событиями

- Сообщения, хотя и посылаются приложениям, но адресуются *окнам*.
- Типичное приложение строится на базе каркаса, содержащего *цикл обработки сообщений*. В этом цикле выполняется прием сообщений и передача их в соответствующие *функции-обработчики сообщений*.

Обработка сообщений



Венгерская нотация

Имя переменной начинается с одной или нескольких строчных букв, которые обозначают тип данных для переменной, например: `lpText`

Префикс	Тип данных
`c`	символ
`by`	BYTE (беззнаковый символ)
`n`	короткое целое
`i`	целое
`x, y`	целое (используется в качестве координат *x* и *y*)
`cx, cy`	целое (используется в качестве длины *x* и *y*), *c* означает "счет" — (count)
`b или f`	BOOL (булево целое); *f* означает "флаг" — (flag)
`w`	WORD (беззнаковое короткое целое)
`l`	LONG (длинное целое)
`dw`	DWORD (беззнаковое длинное целое)
`fn`	функция
`s`	строка
`sz`	строка, завершаемая нулем (string terminated by zero)
`h`	описатель (handle)
`p`	указатель (pointer)

Типы данных Windows

В Windows используется большое количество специфических типов, которые отображают не физическую организацию данных, а их назначение.

Тип данных	Описание
BOOL	Булевский тип (эквивалентный bool)
BYTE	Байт (8-битное целое без знака)
DWORD	32-битное целое без знака
INT	32-битное целое со знаком
LONG	32-битное целое со знаком
LPARAM	Тип, используемый для описания lParam, четвертого параметра оконной процедуры
LPCSTR	Указатель на константную С-строку
LPCTSTR	LPCWSTR, если определен макрос UNICODE, и LPCSTR в противном случае
LPCWSTR	Указатель на константную Unicode-строку
LPSTR	Указатель на С-строку
LPTSTR	LPWSTR, если определен макрос UNICODE, и LPSTR в противном случае
LPWSTR	Указатель на Unicode-строку
LRESULT	Значение типа LONG, возвращаемое оконной процедурой
NULL	((void*) 0)
TCHAR	Wchar_t (Unicode-символ), если определен макрос UNICODE, и char в противном случае
UINT	32-битное целое без знака
WPARAM	Тип, используемый для описания wParam, третьего параметра оконной

Описатели

Независимо от своего типа, любой объект в Windows идентифицируется своим *дескриптором*, или *описателем* (пер. с англ. handle).

Дескриптор — это своего рода ссылка на объект (просто число , обычно длиной в 32 разряда).

Тип данных	Описание
HANDLE	Дескриптор объекта
HBITMAP	Дескриптор растрового изображения (битмэпа)
HBRUSH	Дескриптор кисти
HCURSOR	Дескриптор курсора
HDC	Дескриптор контекста устройства
HFONT	Дескриптор шрифта
HICON	Дескриптор иконки (пиктограммы)
HINSTANCE	Дескриптор экземпляра приложения
HMENU	Дескриптор меню
HPEN	Дескриптор пера
HWND	Дескриптор окна

Константы

В Windows используется огромное количество символических констант, определяющих режимы работы тех или иных программных средств или свойства создаваемых объектов.

Префикс	Категория
CS_	Опция стиля класса
IDI_	Идентификационный номер иконки
IDC_	Идентификационный номер курсора
WS_	Стиль окна
CW_	Опция создания окна
WM_	Сообщение окна
SND_	Опция звука
DT_	Опция рисования текста

Заголовочные файлы

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

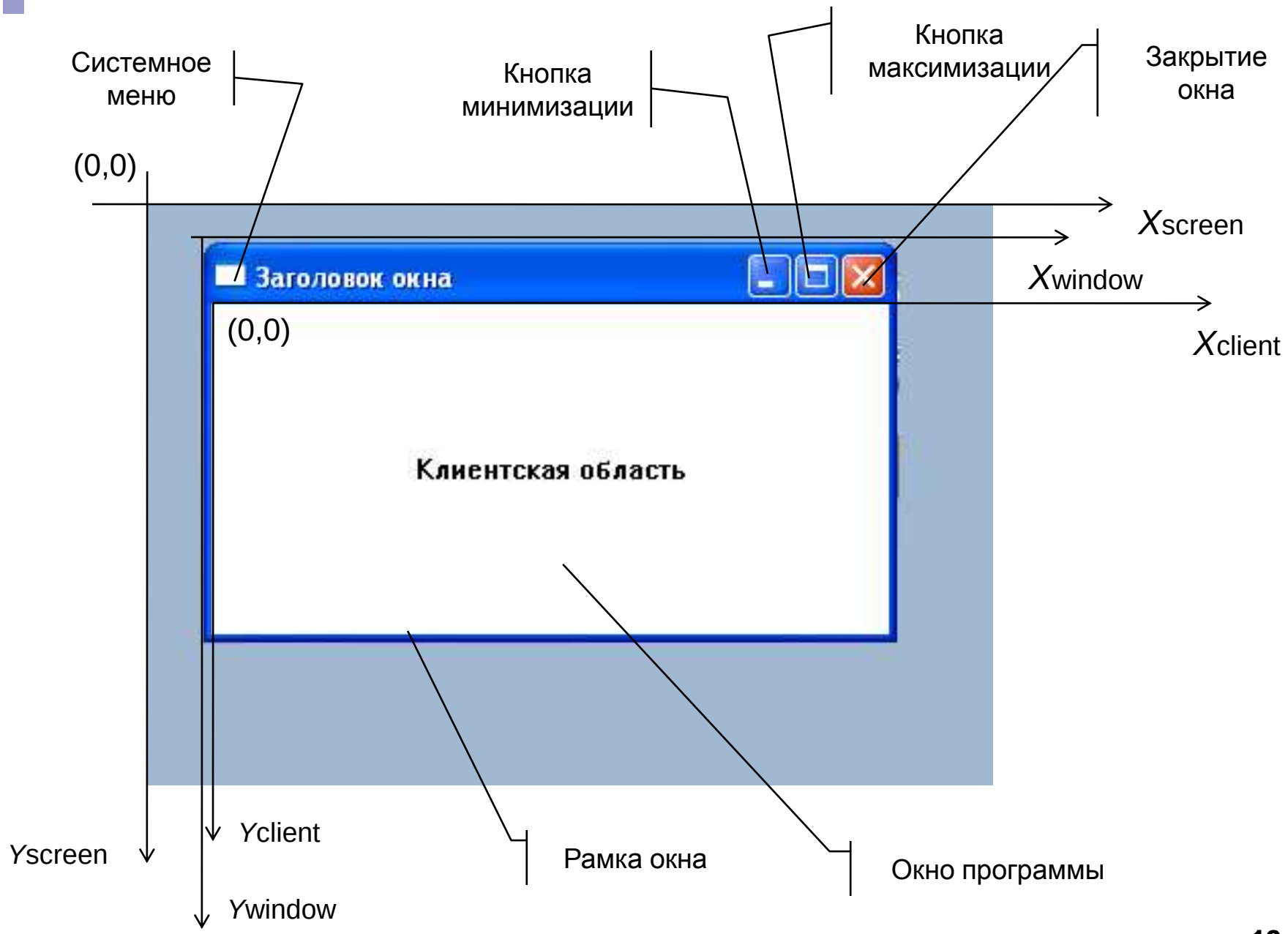
В заголовочных файлах хранится главным образом информация следующего вида:

- вложенные директивы **#include**, с помощью которых в программу включаются дополнительные системные заголовочные файлы (**`winuser.h`**, **`wingdi.h`**, **`winnt.h`** и др.);
- определения констант, используемых функциями Windows;
- определения новых типов данных Windows;
- прототипы функций Windows;
- макросы (**`windowsx.h`**).

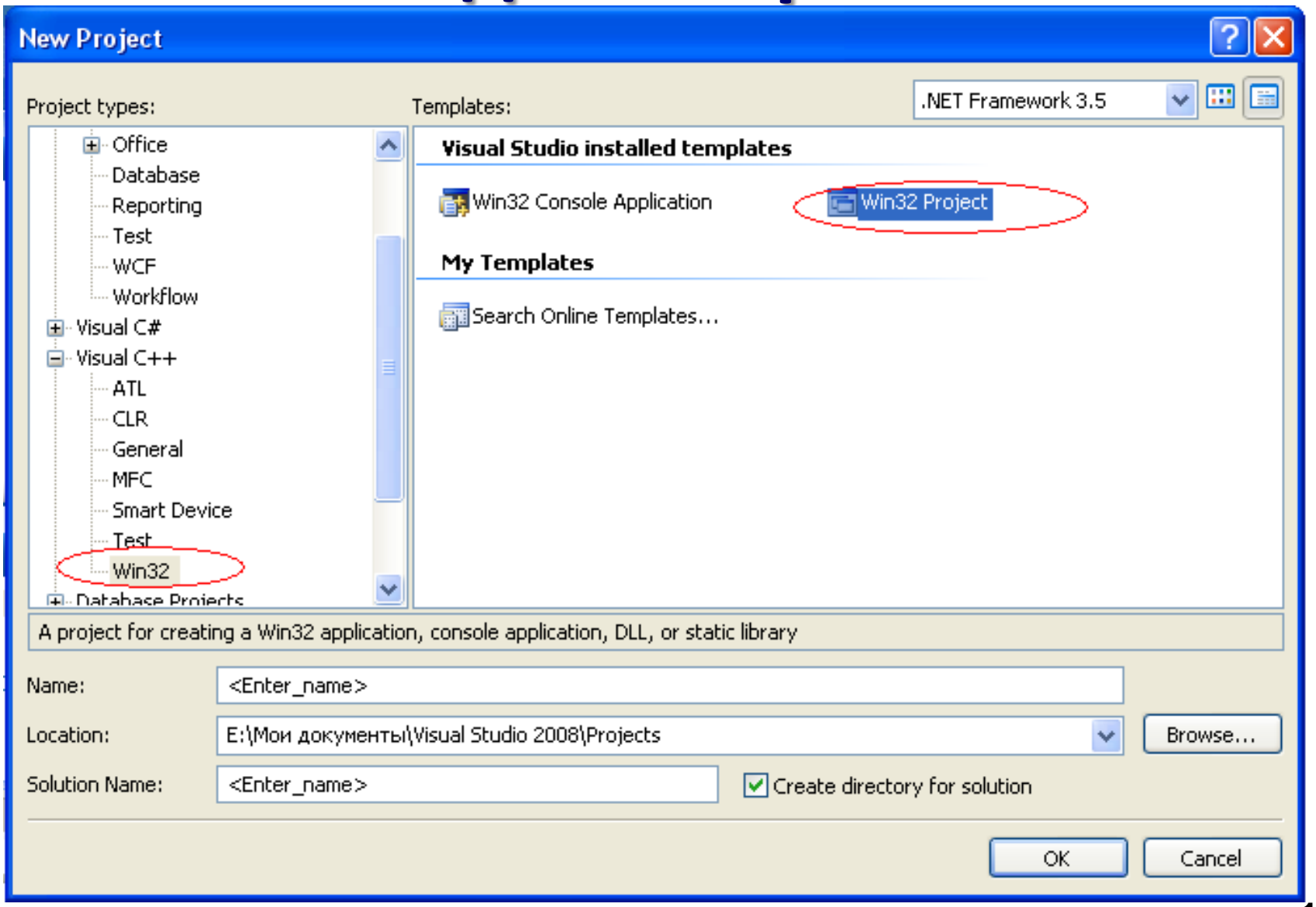
Системы координат Windows

В функциях Win32 может использоваться одна из следующих систем координат:

- *экранные координаты (screen coordinates);*
- *оконные координаты (window coordinates);*
- *координаты клиентской области (client coordinates).*

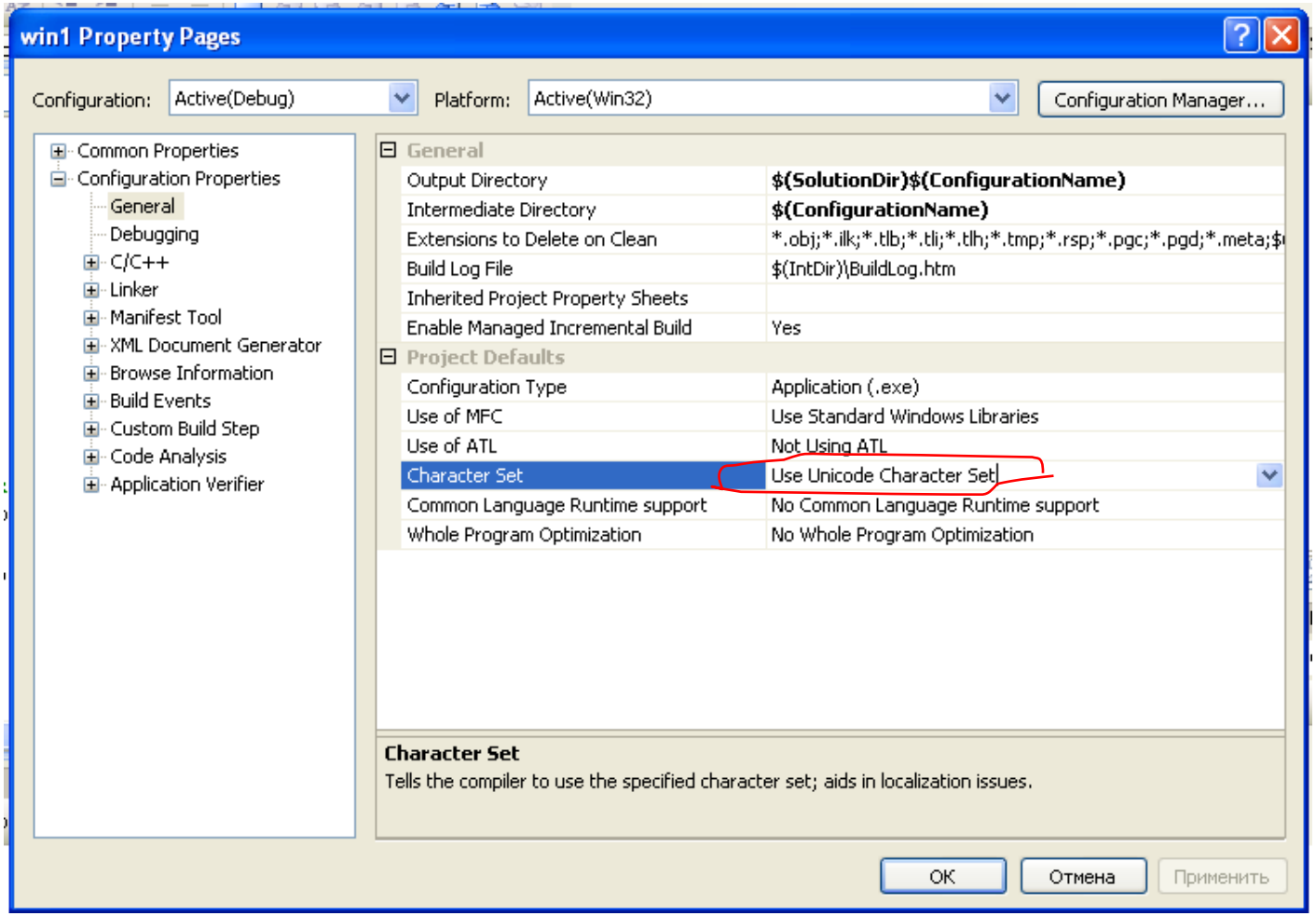

Системное меню
Кнопка минимизации
Кнопка максимизации
Закрытие окна
(0,0)
Xscreen
Xwindow
Xclient
Заголовок окна
(0,0)
Клиентская область
Yscreen
Yclient
Ywindow
Рамка окна
Окно программы

Создание проекта



Настройка среды разработки

1) Для UNICODE *Use Unicode Character Set*



2) Для ANSII *Use Multi-Byte Character Set*

Простейшее Windows-приложение

1) Для UNICODE

```
int WINAPI WinMain (HINSTANCE d1, HINSTANCE d2, LPSTR d3, int d4)
{
        MessageBox(NULL, L"Hello, World!", L"", MB_OK);
        return 0;
}
```

2) Для ANSII

```
int WINAPI WinMain (HINSTANCE d1, HINSTANCE d2, LPSTR d3, int d4)
{
        MessageBox(NULL, "Hello, World!", "", MB_OK);
        return 0;
}
```

Простейшее Windows-приложение

Функция имеет следующий прототип:

```
int MessageBox(HWND hWnd, LPCTSTR lpText,
               LPCTSTR lpCaption, UINT uType);
```

Параметры:

- *hWnd* – дескриптор родительского окна. Он принимает значение NULL, если родительского окна нет.
- *lpText* – указатель на строку, содержащую текст сообщения.
- *lpCaption* — указатель на строку, содержащую текст заголовка диалогового окна.
- *uType* – параметр содержит комбинацию флагов, задающих количество и типы кнопок в диалоговом окне, а также наличие заданной пиктограммы.

Спасибо за внимание!