Лекция №1. Принципы программирования в среде Windows

Направление подготовки

090900.62 – Информационная безопасность

Галимов Р.Р., 2013

Организационная часть

Длительность курса:2 семестра

1-ый семестр

Лекции – каждую неделю(18)

Практические занятия – раз в неделю

Лабораторные работы –каждую неделю (18)

Итоговый контроль: Зачет

2-ой семестр

Лекции -9 часов (4 занятия)

Лабораторные работы – 9 занятий

Итоговый контроль: экзамен

Организационная часть

Список литературы

1. Гильберт С., Маккарти Б. Самоучитель Visual C++6 в примерах.
2. Хортон Айвор. Visual C++2010:полный курс.: Пер. с англ. –М.: ООО «И.Д. Вильямс»,2011.-1216с.
3. Литвиненко Н.А. Технология программирования на C++. Win32 API.
4. Иванова Г.С, Ничушкина Т.Н., Пугачев Е.К. Объектно-ориентированное программирование: Учеб. для вузов/Под ред. Г.С. Ивановой. - М.: Изд-во МГТУ им. Н.Э. Баумана.
5. Павловская Т. А., Щупак Ю. А. C++. Объектно-ориентированное программирование: Практикум. — СПб.:Питер, 2006. — 265 с: ил.

Организационная часть

Язык программирования: C#

Среда программирования: Visual C# (начиная с Visual C++6.0).
Приведенные в лекциях лабораторные работы разработны в Visual Studio 2012

Пропуски: пропускать занятия запрещено!!!

1. **Снижается оценка за модуль на бал, соответственно, теряется шанс на «автомат»;**
2. **Пропущенные лекции должны быть переписаны. Студент должен отчитаться перед преподавателем по пропущенной теме;**
3. **Пропущенные практические занятия отрабатываются подготовкой докладов**
4. **Наличие неотработанных пропусков не позволяет получить зачет или экзамен.**
5. **График отработок будет сообщен дополнительно**

Организационная часть

Цель курса:

- **научиться программировать в среде Visual Studio C# для операционной системы Windows;**
- **получить навыки разработки программ с использованием технологии ООП;**

Лекция №1. Принципы программирования в среде Windows

1.1 Проблемы, возникающие при работе в DOS

1.1.1 Проблемы, связанные с пользовательским интерфейсом

Для большинства пользователей Windows GUI упростил работу за компьютером.

Но программисты DOS большую часть времени тратят на разработку пользовательского интерфейса.

1.1.2 Проблемы, связанные с зависимостью от аппаратных средств

Приложения DOS были сильно привязаны к аппаратным средствам , для которых они были первоначально разработаны.

Популярность текстового редактора WordPerfect объяснялась тем, что она поддерживала стек драйверов для работы с практически любым принтером.

Сильная аппаратная зависимость ставила перед программистом выбор:

- создавать приложение на уровне «наименьшего знаменателя»;
- поддерживать расширенные возможности только для выбранного списка устройств;
- потратить время и деньги на поддержку большого количества аппаратных средств.

1.1 Проблемы, возникающие при работе в DOS

1.1.3 Проблемы, связанные с одновременным выполнением

DOS- однозадачная операционная система. Под её управлением в каждый момент времени может выполняться только одна программа.

Программы не в состоянии обмениваться информацией друг с другом.

Пользователи желают иметь программы, которые работали бы совместно и без проблем бы обменивались информацией.

Windows была разработана для решения данных проблем.

1.2 Решения, предлагаемые ОС Windows

1.2.1 Основные свойства Windows:

- **Общий интерфейс пользователя;**
- **Общесистемный ввод, использующий очередь, и система передачи сообщений;**
- **Независящая от внешних устройств архитектура ввода-вывода**
- **Многозадачность на уровне приложений и взаимодействие между процессами**

1.2 Решения, предлагаемые ОС Windows

1.2.2 Общий интерфейс пользователя

Работа с общим интерфейсом значительно упрощает работу пользователя.

Эффект для программистов:

- Windows предлагает обширную библиотеку общих, встроенных подпрограмм пользовательского интерфейса. Причем эти подпрограммы будут гарантированно находится в системе;

-Windows поддерживает встроенный диспетчер, управляющий окнами и меню. Программист можно не уделять внимание таким деталям пользовательского интерфейса, как изменение размера окна и обработка пунктов меню.

1.2 Решения, предлагаемые ОС Windows

1.2.3 Архитектура передачи сообщений Windows

Windows реализует систему связи – передача сообщений.

Традиционная программа DOS для ввода данных использует метод «поллинга», когда программа опрашивает устройство ввода на наличие данных.

Windows же принимает на себя исключительные права управления аппаратными средствами. Windows непосредственно получает каждое событие, возникающее при системном вводе, которые генерируются мышью, клавиатурой, таймером и записывает события. Эти записи затем передаются приложениям, которые нуждаются в них.

Все, что должна сделать ваша программа – ответить соответствующим образом на событие.

1.2 Решения, предлагаемые ОС Windows

1.2.4 Архитектура ввода-вывода, независимая от внешних устройств

При разработке DOS-программ необходимо было знать специфику аппаратных средств, используемых в компьютере с тем, чтобы создавать программу с учетом обеспечения прямого доступа к аппаратным средствам.

Конечно, это справедливо не для всех устройств. Например, DOS обеспечивает хорошую поддержку разных файловых систем.

Таким образом, под управлением DOS запись в файлов не зависит от внешних устройств, но вывод на принтер и дисплей таковыми не являются.

1.3 Основы программирования для ОС Windows

При разработке приложений в Visual C++ 2010 три основных способа разработки программ:

- использование **интерфейса API Windows**;
- использование классов **Microsoft Foundation Classes**;
- использование **Windows Forms**;

1.3 Основы программирования для ОС Windows

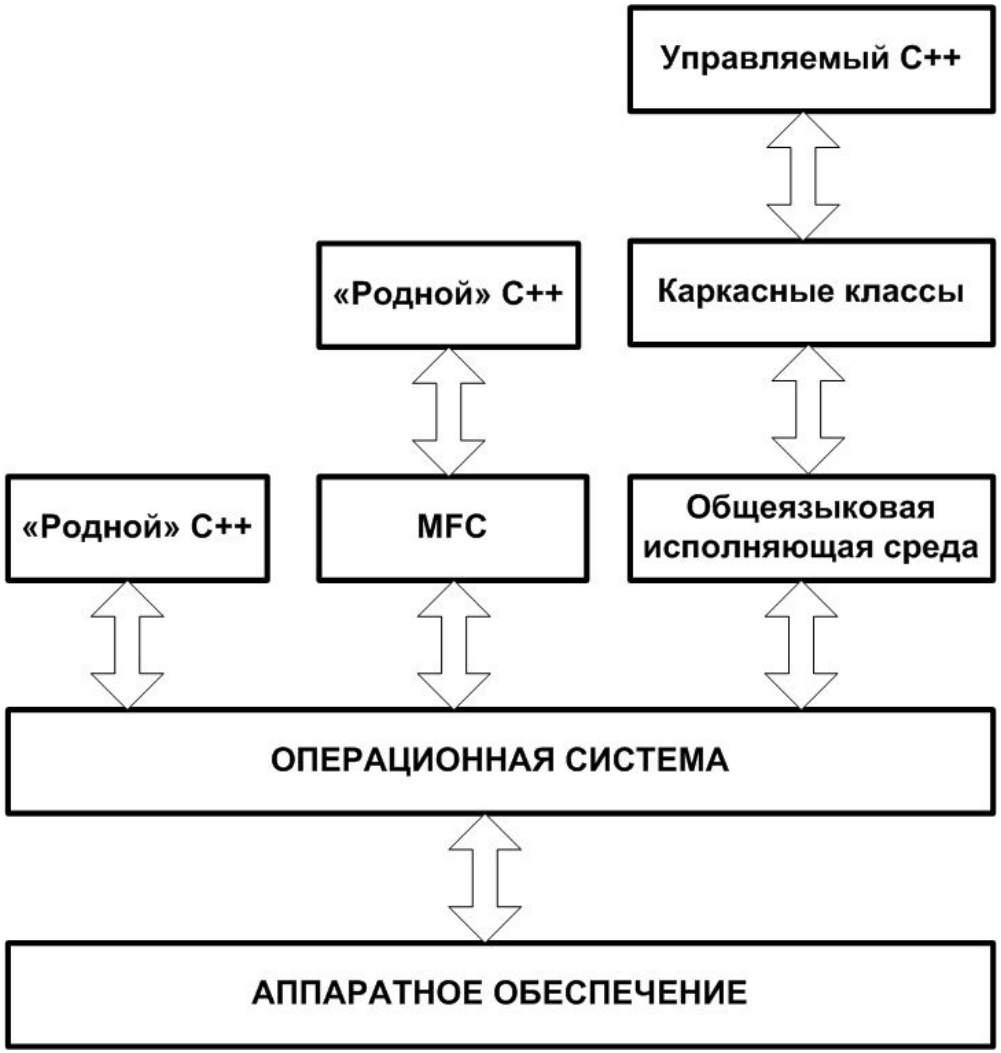

Рисунок 1.1 – Способы разработки приложений в среде Visual C++ 2010

1.3 Основы программирования для ОС Windows

1.3.1 Программирование, управляемое событиями

Традиционные программы организованы иерархическим образом. Например, в языке C++ наверху пирамиды находится функция main(), которая вызывает подпрограммы.

Однако программа Windows не организована иерархически в чистом виде. Программа содержит функции, но они предназначены для формирования отклика на внешние события.

Поэтому говорят, что программы Windows **управляются событиями**.

События в приложении Windows представляют собой различные происшествия, в частности:

- щелчок кнопкой мыши;
- Нажатие клавиши;
- Истечение определенного интервала времени;
- События могут быть сгенерированы функциями внутри вашей программы или других программ.

1.3 Основы программирования для ОС Windows

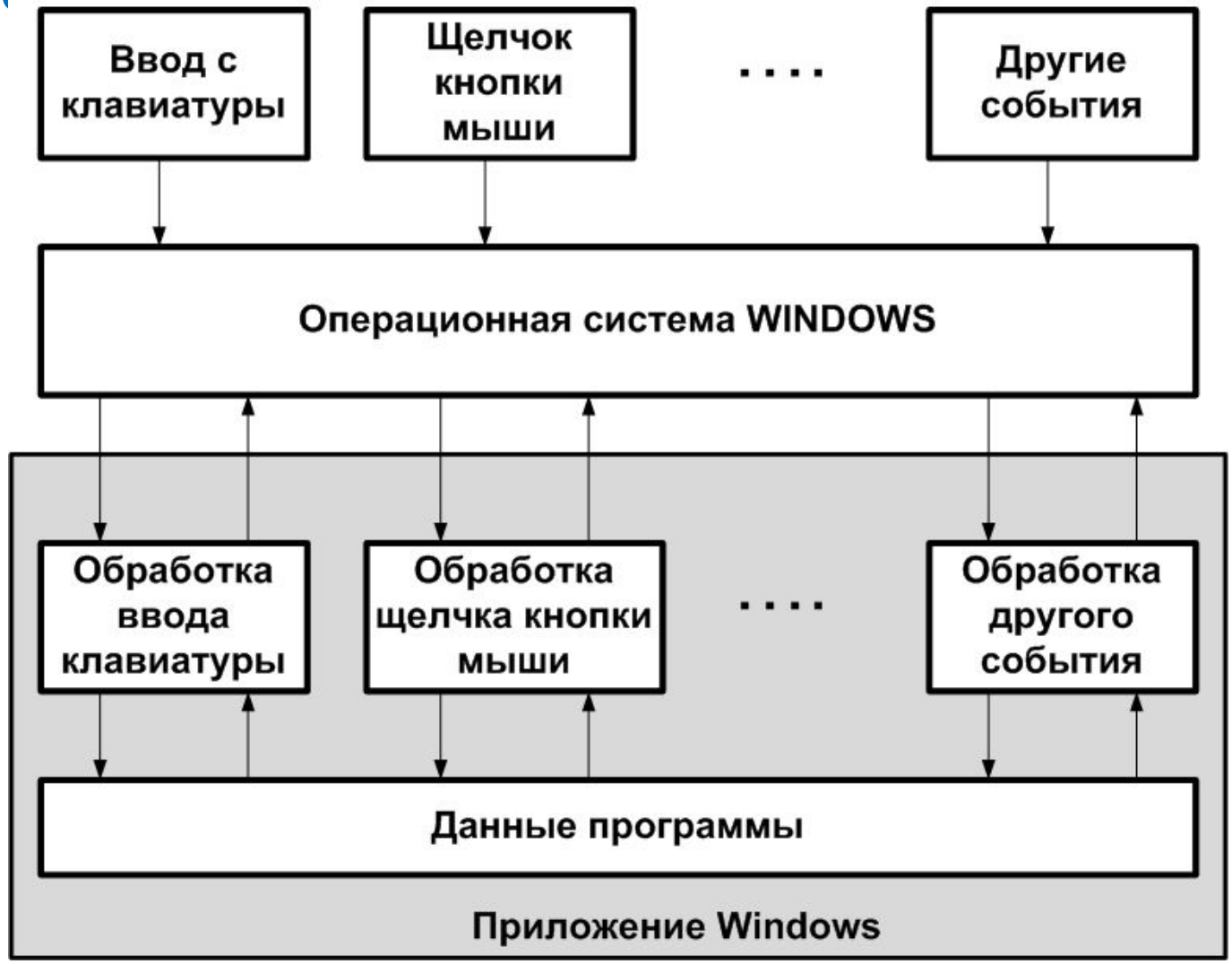

Рисунок 1.2 – Структура Windows-приложения

1.3 Основы программирования для ОС Windows

1.3.2 Генерация аппаратных событий

На аппаратном уровне каждое устройство ввода Windows управляется прерываниями.

Когда пользователь нажимает клавишу клавиатуры, генерируется аппаратное прерывание. Windows приостанавливает работу и передает управлению фрагменту кода –**Interrupt Service Routine** –ISR (Программа обработки прерываний).

ISR формирует специальные данные и записывает их в регистры. Затем вызывается специальная внутренняя программа Windows, которая извлекает данные из регистров и помещает запись о событии в аппаратную очередь Windows.

Далее аппаратные события передаются в очередь приложения.

Таким образом, даже при том, что аппаратный ввод является управляемым прерываниями, сообщения события обрабатываются приложением в порядке FIFO.

1.3 Основы программирования для ОС Windows

1.3.3 Назначение сообщений

Сообщения – это стандартный механизм связи внутри

Рисунок 1.3 – Источники сообщений

События аппаратного ввода генерируются мышью, клавиатурой, таймером и т.д.

События диспетчера окон генерируются Windows в ответ на действия пользователя, выполняющего операции типа перемещения или изменения размеров окна, выбора пунктов меню. Пример сообщения данного типа –**WM_PAINT**.

Индивидуальные окна могут посылать сообщения другим окнам. Например, для указания текстовому полю об удалении текста посылается сообщение **WM_SETTEXT**. Текстовое поле также использует данный механизм для сообщения об изменении текста путем посылки программисту сообщения EN_CHANGE.

1.3 Основы программирования для ОС Windows

1.3.3 Назначение сообщений

Сообщения Windows идентифицируются мнемоническими идентификаторами, которые соответствуют целочисленным константам.

Сообщения, сгенерированные непосредственно Windows, начинаются с символов «**WM**_» (Windows Message).

Сообщения, сгенерированные специфическими видами средств управления Windows, имеют различные префиксы. Например, сообщения, сгенерированные средствами редактирования имеют префикс «**EN**_» (Edit Notification).

Все эти константы определены в файле **WinUser.h.**

```
#define WM_NULL                 0x0000
#define WM_CREATE               0x0001
#define WM_DESTROY              0x0002
#define WM_MOVE                 0x0003
#define WM_SIZE                 0x0005
```

1.3 Основы программирования для ОС Windows

1.3.3 Назначение сообщений

```
typedef struct tagMSG {    // msg
  HWND hwnd;
  UINT message;
  WPARAM wParam;
  LPARAM lParam;
  DWORD time;
  POINT pt;
} MSG;
```

hwnd - Определяет окно которого процедура окна получает сообщение.

Message - Указывает номер сообщения.

wParam - Определяет дополнительную информацию о сообщении. Точный смысл зависит от значения элемента сообщение.

lParam - Определяет дополнительную информацию о сообщении. Точный смысл зависит от значения элемента сообщение.

time - Указывает время, когда сообщение было создано.

pt -Задает положение курсора в координатах экрана, когда сообщение было создано.

1.3.4 Основные понятия. Элементы окна

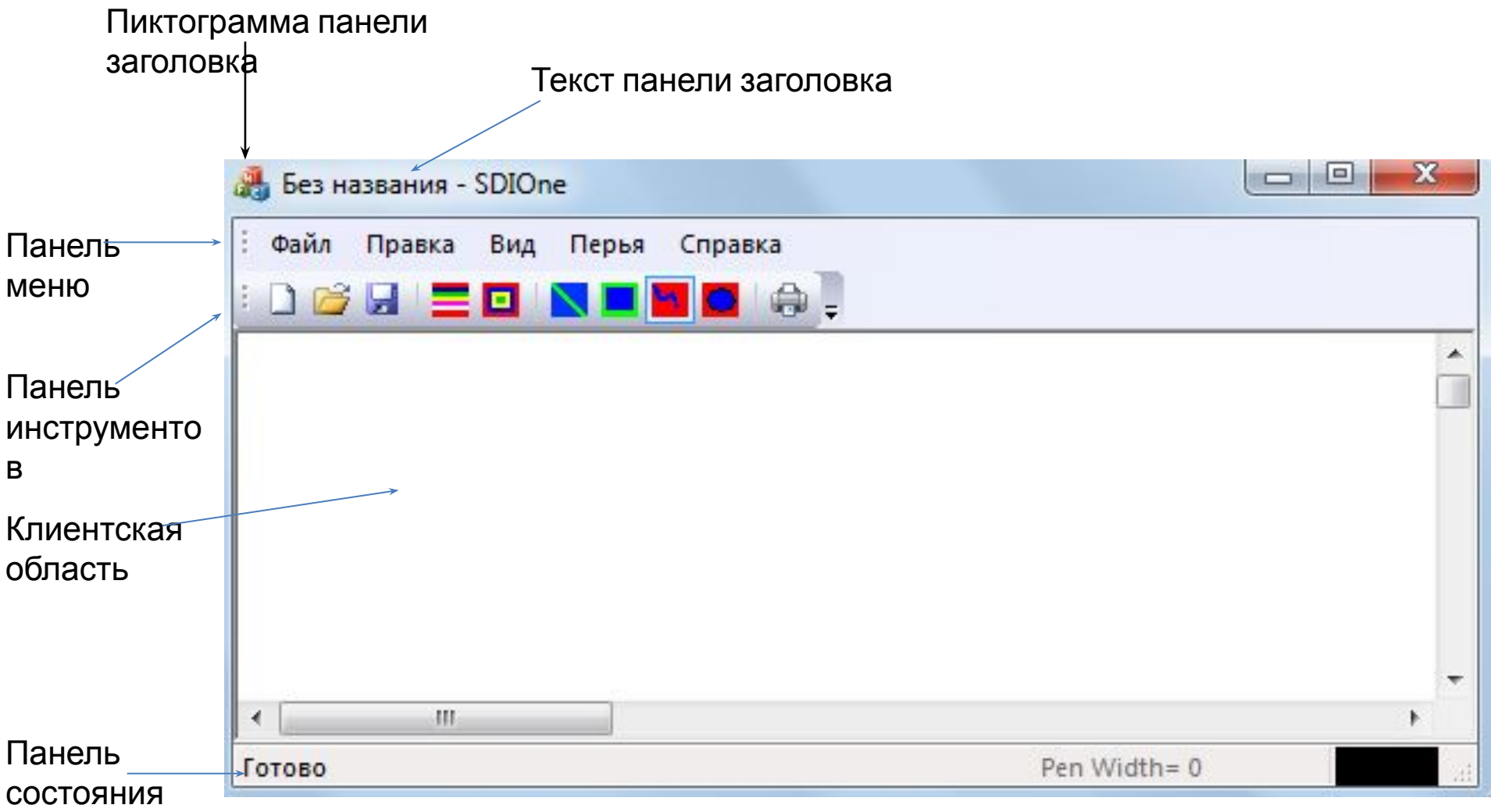

Рисунок 1.4 – Стандартные окна программы Windows

1.4 Интерфейс Windows API

Разработчиками ОС Windows была создана библиотека функций, при помощи которых происходит взаимодействие приложения с операционной системой, так называемые функции Программного интерфейса приложений (Application Program Interface).

Для разработки минимальной программы с использованием Win API необходимо написать две функции:

- WinMain(), с которой начинается выполнение программы и происходит её инициализация;
- WndProc(), вызываемая Windows для обработки сообщений приложе-ния[Хортон, Айвор. Visual C++2010:Полный курс.: Пер. с англ. –М.: ООО «И.Д. Вильямс», 2011. -1216с.].

1.4 Интерфейс Windows API

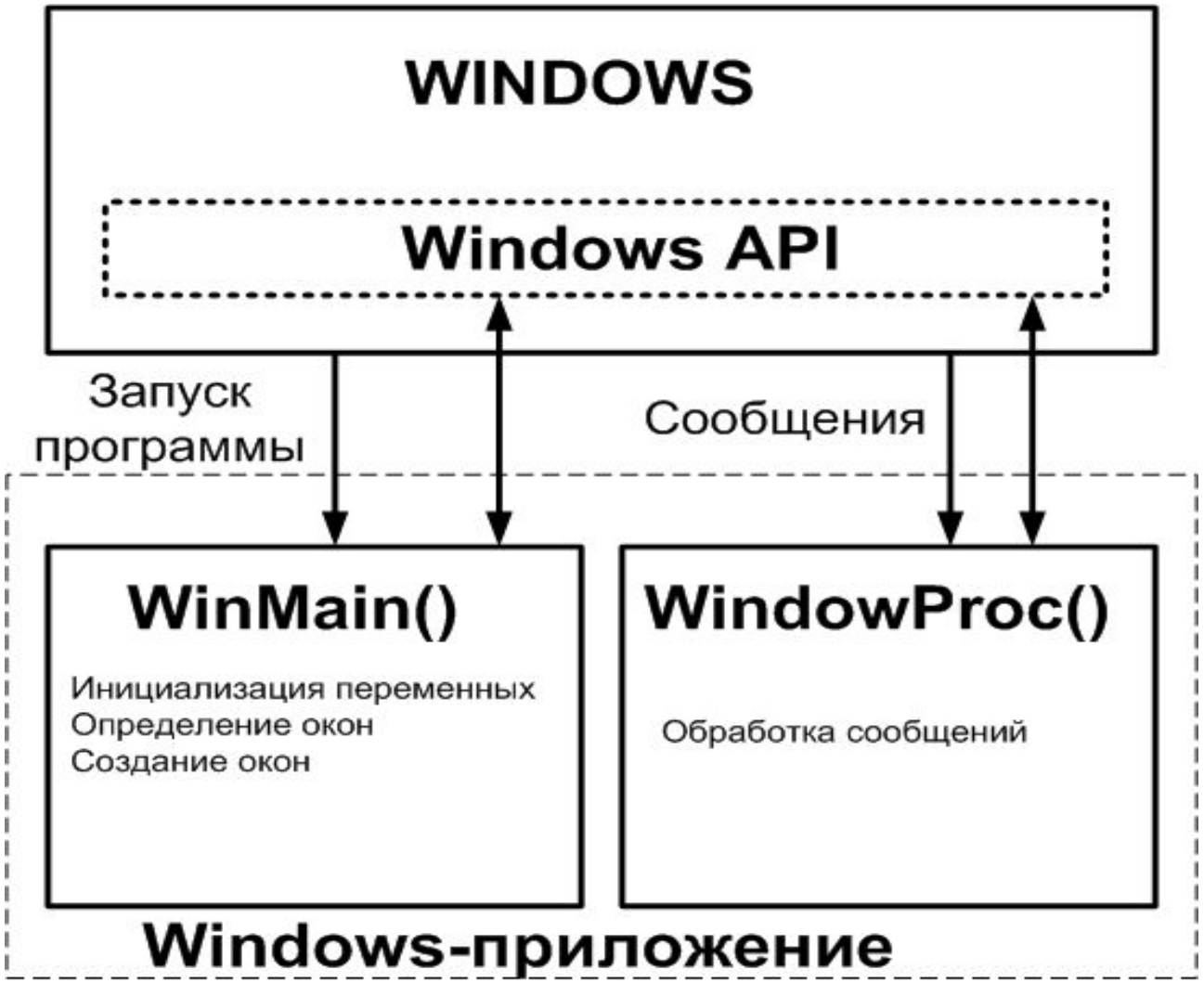

Рисунок 1.5 – Структура Windows-программы

1.4 Интерфейс Windows API

Функция WinMain() выполняет четыре основные функции:

- сообщает ОС, какого вида окно требуется программе;
- создает окно программы;
- инициализировать окно программы;
- извлекает сообщения Windows, предназначенные программе.

На псевдокоде функцию WinMain() можно представить следующим образом:

```
WinMain(список аргументов)
{
    Подготовить и зарегистрировать класс окна с нужными
    характеристиками;
Создать экземпляр зарегистрированного класса;
Пока не произошло необходимое для выхода события
{
     Извлечь очередное сообщение из очереди сообщений;
     Передать его через Windows оконной функции;
}
Возврат из программы
}
```

1.4 Интерфейс Windows API

Функция WinMain()

После того как Windows загружает программу, то она вызывает функцию WinMain.

```
int APIENTRY _tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR    lpCmdLine, int
      nCmdShow)
{
      MyRegisterClass(hInstance);
      // Выполнить инициализацию приложения:
      if (!InitInstance (hInstance, nCmdShow))
      {
        return FALSE;
      }
      hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_WIN32_API));
      // Цикл основного сообщения:
      while (GetMessage(&msg, NULL, 0, 0))
      {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
      }

      return (int) msg.wParam;
}
```

1.4 Интерфейс Windows API

Функция WindowProc()

Функция WndProc() является «функцией обратного вызова». Такие функции вызываются операционной системой, а не самой программой.

WindowProc() обрабатывает все сообщения, которые поступают окну программы.

LRESULT CALLBACK WINDOWPROC(HWND hWnd, UINT message,WPARAM wParam, LPARAM lParam);

Аргументы функции:

hWnd – дескриптор окна, в котором произошло событие;

message – тип сообщения;

1.4 Интерфейс Windows API

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
	int wmId, wmEvent;
	PAINTSTRUCT ps;
	HDC hdc;
	switch (message)
	{
	case WM_COMMAND:
	   wmId    = LOWORD(wParam);
	   wmEvent = HIWORD(wParam);
	   // Разобрать выбор в меню:
	   switch (wmId)
	   {
	   case IDM_ABOUT:
			DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
			break;
	   case IDM_EXIT:
			DestroyWindow(hWnd);
			break;
	   default:
			return DefWindowProc(hWnd, message, wParam, lParam);
	   }
	   break;
	case WM_PAINT:
	   hdc = BeginPaint(hWnd, &ps);
	   // TODO: добавьте любой код отрисовки...
	   EndPaint(hWnd, &ps);
	   break;
	case WM_DESTROY:
	   PostQuitMessage(0);
	   break;
	default:
	   return DefWindowProc(hWnd, message, wParam, lParam);
	}
	return 0;
}
```

1.4 Интерфейс Windows API

Передача сообщений

```
while (GetMessage(&msg, NULL, 0, 0))
   {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
   }
```

1.4 Интерфейс Windows API

```
  case WM_PAINT:
    hdc = BeginPaint(hWnd, &ps);
    // TODO: добавьте любой код отрисовки...
    GetClientRect(hWnd,&rect);
    DrawText(hdc,L"Здравствуй, МИР!",-1,&rect,
  DT_SINGLELINE|DT_CENTER|DT_VCENTER);
    EndPaint(hWnd, &ps);
    break;
  case WM_DESTROY:
    PostQuitMessage(0);
    break;
  default:
    return DefWindowProc(hWnd, message, wParam, lParam);
  }
  return 0;
```

Список источников

1. Гильберт С. Самоучитель Visual C++ 6 в примерах/С. Гильберт,Б. Маккарти