Лабораторная работа № 1

Функции получения системной информации

Цель работы: получение практических навыков по программированию в Win32 API.

Windows API (Application Programming Interface) —программный интерфейс пользовательского режима для ОС семейства Windows. До появления 64-разрядных версий Windows программный интерфейс 32-разрядных версий ОС Windows назывался Win32 API в отличие от исходного 16-разрядного Windows API, программного интерфейса 16-разрядных версий Windows. В настоящее время термин Windows API может относиться как к 32-разрядным, так и 64-разрядным программным интерфейсам Windows. Описание Windows API содержится в документации Windows SDK. Документация доступна бесплатно по адресу https://developer.microsoft.com/en-us/windows/desktop/develop. Также она включается во все уровни подписки MSDN (Microsoft Developer Network), программы поддержки разработчиков от компании Microsoft.

Win32 API состоит из набора функций и подпрограмм, предоставляющих программный доступ к возможностям операционной системы. Win32 API содержит тысячи функций для реализации всех видов сервисов операционной системы.

API-функции Windows входят в состав библиотек динамической компоновки (DLL). Библиотека динамической компоновки является исполняемым файлом, который содержит несколько экспортируемых функций, то есть функций, к которым могут обращаться другие исполняемые приложения (EXE или DLL). Файлы DLL намного проще файлов EXE, например, в них нет кода, который управлял бы графическим интерфейсом или обрабатывал сообщения Windows.

Для размещения API-функций Windows использует несколько DLL. В действительности большая часть функций Win32 API содержится в трех DLL:

KERNEL32.DLL - содержит около 700 функций, которые предназначены для управления памятью, процессами и потоками;

USER32.DLL - предоставляет порядка 600 функций для управления пользовательским интерфейсом, например, созданием окон и передачей сообщений;

GDI.DLL - экспортирует около 400 функций для рисования графических образов, отображения текста и работы со шрифтами.

Кроме этих библиотек Windows также содержит несколько других DLL более узкой специализации. Например,

COMDLG32.DLL - открывает доступ почти к 20 функциям управления стандартными диалоговыми окнами Windows;

LZ32.DLL - хранит примерно 12 функций архивирования и разархивирования файлов;

ADVAPI32.DLL - экспортирует около 400 функций, связанных с защитой объектов и работой с реестром;

WINMM.DLL - содержит около 200 функций, относящихся к мультимедиа.

В Windows 8 появились новый API и исполнительная среда поддержки Windows Runtime (WinRT). Windows Runtime состоит из платформенных сервисов, предназначенных для разработчиков приложений Windows Apps (ранее назывались Metro Apps, Modern Apps, Immersive Apps и Windows Store Apps). Приложения Windows Apps подходят для различных форм-факторов устройств, от миниатюрных IoT-устройств до телефонов, планшетов, ноутбуков, настольных систем, и даже таких устройств, как Xbox One и Microsoft HoloLens. Приложения Windows Apps строятся по новым правилам и не похожи на привычные приложения Windows, которые теперь называются десктопными или классическими приложениями Windows. Классические приложения могут использовать подмножество WinRT API и, наоборот, приложения Windows Apps могут использовать

подмножество Win32. Информация о том, какие API доступны для каждой из платформ приложений, содержится документации MSDN.

Основные Win32 API-функции получения системной информации:

GetComputerName	GetSystemMetrics	GetWindowsDirectoty
GetKeyboardType	GetTempPath	SetComputerName
GetSysColor	GetUserName	SetSysColors
CecSystemDirectory	GetVersion	SystemParametersInfo
GetSystemlnfo	GetVersioriEx	GetUserName

Имя компьютера

Функция GetComputerName используется для получения текущего имени компьютера. Связанная с ней SetComputerName используется для присвоения имени компьютеру.

```
BOOL GetComputerName(
   LPTSTR  IpBuffer,    // Адрес  буфера  имени.
   LPDWORD nSize      // Размер буфера имени.
);
```

В соответствии с документацией, выполнение функции GetComputerName в Windows 9x завершится неудачей, если размер буфера входных данных меньше, чем величина константы MAX_COMPUTERNAME_LENGTH + 1.

Пути к системным каталогам Windows

Функции GetWindowsDirectory, GetSystemDirectory и GetTempPath находят путь к каталогу, к системному каталогу и к каталогу временных файлов Windows. Например, функция GetSystemDirectory определена как:

```
U1NT GetSystemDirectory(
   LPTSTR IpBuffer,                  // Адрес буфера системного каталога.
   UINT nSize
);                                   //  Размер буфера каталога.

UINT GetWindowsDirectory(
   LPTSTR IpBuffer,                  // Адрес буфера каталога Windows.
   UINT nSize                        // Размер буфера каталога.
);
DWORD GetTempPath(
   DWORD nBufferLength,              // Размер буфера в символах.
    LPTSTR IpBuffer                  // Указатель на буфер пути к каталогу
                                     // временных файлов.
);
```

Версия операционной системы

В табл. 1 перечислены названия продуктов семейства Windows, их внутренние номера версий и даты релиза. Начиная с Windows 7, нумерация версий перестала быть очевидной. Системе был присвоен номер версии 6.1 вместо 7. В момент выхода Windows Vista (номер версии 6.0) Windows XP (версия 5.1) оставалась еще весьма популярной, и в некоторых приложениях проверка версии ОС стала работать некорректно, так как проверялись два условия: номер основной версии больше или равен 5, номер дополнительной версии больше или равен 1. Соответственно, для Windows Vista эти условия не выполнялись. Во избежание в будущем подобных коллизий компания Microsoft

решила оставить номер основной версии равным 6 с повышением номера дополнительной версии для новых продуктов. Однако в Windows 10 номер версии был обновлен до 10.0.

Таблица 1. Релизы операционной системы Windows

Название продукта	Внутренний номер версии	Дата релиза
Windows NT 3.1	3.1	июль 1993 г.
Windows NT 3.5	3.5	сентябрь 1994 г.
Windows NT 3.51	3.51	май 1995 г.
Windows NT 4.0	4.0	июль 1996 г.
Windows 2000	5.0	декабрь 1999 г.
Windows XP	5.1	август 2001 г.
Windows Server 2003	5.2	март 2003 г.
Windows Server 2003 R2	5.2	декабрь 2005 г.
Windows Vista	6.0	январь 2007 г.
Windows Server 2008	6.0 (Service Pack 1)	март 2008 г.
Windows 7	6.1	октябрь 2009 г.
Windows Server 2008 R2	6.1	октябрь 2009 г.
Windows 8	6.2	октябрь 2012 г.
Windows Server 2012	6.2	октябрь 2012 г.
Windows 8.1	6.3	октябрь 2013 г.
Windows Server 2012 R2	6.3	октябрь 2013 г.
Windows 10	10.0 (сборка 10240)	июль 2015 г.
Windows 10 версия 1511	10.0 (сборка 10586)	ноябрь 2015 г.
Windows 10 версия 1607 (Anniversary Update)	10.0 (сборка 14393)	июль 2016 г.
Windows Server 2016	10.0 (сборка 14393)	октябрь 2016 г.

Функция GetVersionEx возвращает информацию о версии операционной системы Windows и может использоваться для определения рабочей системы -Windows 95, Windows 98 или Windows NT. Она объявляется как

```
BOOL GetVersionEx(
    LPOSVERSIONINFO IpVersionlnformation.      // Указатель на структуру
                                               //с информацией о версии.
);
```

где *IpVersionlnformation* - указатель на структуру *OSVERSIONINFO*, которая определена следующим образом:

```
typedef  struct _OSVERSIONINFO (
    DWORD dwOSVersionlnfoSise;
    DWORD dwMajorVersion;
    DWORD dwMinorVersion;
    DWORD dwBuildNumber;
    DWORD dwPlatformId;
    TCHAR  szCSDVersion[  128  ];
)  OSVERSIONINFO;
```

В документации об этой структуре говорится следующее:

dwOSVersionlnfoSize

задает размер структуры OSVERSIONINFO в байтах. Для структур это является общим требованием. Так как DWORD - четырехбайтовое беззнаковое типа long и поскольку Delphi и VB преобразуют строку из 128 символов в массив символов ANSI из 128 байт, общий размер структуры составляет 4x5+ 128 = 148 байт.

dwMajorVersion

указывает номер основной версии операционной системы. Например, для Windows NT версии 3.51 номер основной версии - 3. Для Windows NT 4.0 и Windows 9x номер основной версии - 4.

dwMinorVersion

указывает дополнительный номер версии операционной системы.

dwBuildNumber

указывает номер сборки операционной системы для Windows NT. Для Windows 9x два младших байта содержат номер сборки операционной системы, а два старших байта - номер основной версии и дополнительный номер версии.

dwPiatformId

идентифицирует платформу операционной системы, может иметь одно из следующих значений:

VER_PLATFORM_WIN32s (= 0) .	Win32s, работающая на Windows
VER_PLATFORM_WIN32_WINDOWS (= 1).	Win32, работающая на Windows 95 или Windows 98.
VER_PLATFORM_WIN32_NT (= 2).	Win32, работающая на Windows NT

szCSDVersion

в Windows NT содержит строку завершающеюся нулевым символом, например «Service Pack3», которая указывает самую последнюю версию установленного в системе служебного пакета программ (service pack). Строка будет пустой, если служебный пакет не установлен. В Windows 95 включает строку с завершающим нулевым символом, в которой может быть произвольная дополнительная информация об операционной системе.

Начиная с Windows 8, функция Windows API GetVersionEx по умолчанию возвращает номер версии 6.2 (Windows 8) независимо от фактической версии ОС. Это сделано для того, чтобы свести к минимуму проблемы совместимости. Кроме того, некоторые компоненты могут устанавливаться «раньше времени», без согласования с официальным выпуском Windows. В настоящее время эта функция объявлена устаревшей. При необходимости узнать фактическую версию ОС, имеется возможность получить ее при помощи функции VerifyVersionlnfo или более новых вспомогательных API проверки версий: IsWindows80rGreater, IsWindows8Pointl0rGreater, IsWindowsl0OrGreater, IsWindowsServer и т. д. Кроме того, совместимость с разными операционными системами может быть обозначена в манифесте исполняемого файла, что приводит к изменению результатов вызова этой функции.

Для просмотра информации о версии Windows можно воспользоваться программой командной строки ver или ее графической версией winver. Графическая версия также выводит номер сборки Windows, эта информация может быть полезена для участников программы предварительной оценки Windows Insider, а также для управления обновлениями безопасности, так как в ней указан уровень установленного исправления.

Системные метрики

Функция GetSystemMetrics получает информацию о метриках (системе единиц измерения) объектов операционной системы.

```
Int GetSystemMetrics(
int nlndex                    // Системная метрика или установки конфигурации.
) ;
```

Параметр nIndex принимает значение одной из 84 возможных констант. Функция возвращает запрошенные единицы измерения (в общем случае в пикселях или в безразмерных единицах).

Чтобы дать общее представление о типе возвращаемой информации, здесь приведены образцы некоторых констант для этой функции. Единицы измерения высоты и ширины приведены в пикселях:

```
SM_CMOUSEBUTTOMS = 43          // Delphi     ' Количество клавиш мыши.
Const SM_CMOUSEBUTTOMS = 43    // VB
SM_MOUSEWHEELPRESENT = 75                 ' Истина (True), если мышь имеет
                                          ' колесо прокрутки.
```

```
                                        ' (Только Win NT 4 или Win 98.)
SM_SWAPBUTTON = 23                      ' Истина (True), если клавиши мыши
                                        ' можно поменять местами (мышь
                                        ' для левши).
 SM_CXBORDER = 5                        ' Ширина и высота рамки окна.
SM_CYBORDER = 6
SM_CXSCREEN = 0                         ' Ширина и высота экрана.
SM_CYSCREEN = 1
SM_CXFULLSCREEN = 16                     ' Ширина и высота области
                                         ' приложения в полноэкранном
SM_CYFULLSCREEN = 17                    'режиме.
SM_CXHTHUMB = 10                         'Ширина прямоугольного курсора
                                         'в горизонтальной полосе
                                        'прокрутки.
SM_CXICONS PACING = 38                  'Размеры ячейки сетки для
SM  CYICONSPACING = 39                   'значка в режиме просмотра с
                                         'крупными значками.
SM_CYCAPTION = 4                        ' Высота стандартной области
                                        'заголовка.
```

Системные параметры

Функция SystemParamterslnfo - это мощная функция, предназначенная для получения или установки всех системных параметров. Также она может в процессе установки параметра обновлять пользовательские профили. Ниже приведена ее декларация:

```
BOOL  SystemParametersInfo   (
      UINT uiAction,                  //  Запрашиваемый или устанавливаемый
                                      //  системный  параметр.
      UINT uiParam,                   //  Зависит  от принятого  системного
                                      //  параметра.
      PVOID pvParam,                  // Зависит от принятого системного параметра.
      UINT fWinIni                // Флаг обновления пользовательского профиля.
};
```

Эта функция может принимать, по меньшей мере, 90 различных значений uiAction. Ниже приведены некоторые константы uiAction:

SPI_GETACCESSTIMEOUT - используется для определения данных о временных интервалах, относящихся к специальным возможностям Windows;

SPI_GETANIMATION – используется для определения данных об анимации, используемой при сворачивании и восстановлении окон;

SPI_GETBEEP - признак разрешения звуковых сигналов;

SPI_GETBORDER – параметру присваивается коэффициент, управляющий толщиной рамки для изменения размеров окна;

SPI_GETDEFAULTINPUTLANG – параметру присваивается 32-х разрядный дескриптор раскладки клавиатуры по умолчанию;

SPI_GETDRAGFULLWINDOWS – характеристики перемещения окна мышью;

SPI_GETFASTTASKSWITCH – признак, определяющий быстрое переключение задач;

SPI_GETFILTERKEYS - используется для определения данных о специальных возможностях, относящихся к работе с клавиатурой;

SPI_GETFONTSMOOTHING - режимы сглаживания шрифтов;

SPI_GETGRIDGRANULARITY – гранулярность сетки рабочего стола;

SPI_GETICONMETRICS - используется для определения информации о характеристиках иконок.

Системные цвета

Функции GetSysColor и SetSysColors используются для получения и установки цветов различных элементов системы, таких как кнопки, строки заголовков и т.д. Цветовой палитрой также может управлять пользователь с помощью апплета Display (Экран) на панели Control Panel (Панель управления).

Декларация GetSysColor:

```
DWORD GetSysColor (
    int nIndex                          // Элемент экрана.
);
```

nIndex может принимать значение одной из множества символьных констант, например

```
#define  COLOR_ACTIVECAPTION      3
```

Возвращаемое значение - это цвет в формате RGB. В частности, каждый цвет занимает один байт в возвращаемом значении типа unsigned long: красный цвет - младший байт, зеленый - следующий байт, далее - синий цвет. Самый старший байт равен нулю. Байты цветов представлены в переменной типа long в обратном порядке, поскольку при записи переменной в память байты располагаются от младших к старшим.

Объявление функции SetsysColors:

```
BOOL WINAPI SetSysColors (
    int cElements,                      // Количество изменяемых
                                        // элементов.
    CONST INT *lpaElements,             // Адрес массива элементов.
    CONST COLORREF *lpaRgbValues        // Адрес массива значений RGB.
```

Здесь cElements определяет количество системных элементов, цвет которых требуется изменить; IpaElements - указатель на целочисленный массив VC++, который содержит индексы изменяемых элементов; IpaRgbvalues ссылается на целочисленный массив VC++ новых значений цвета в формате RGB.

Функции для работы со временем

Во внутренней работе Windows используется универсальное координированное время UTC (Universal Coordinated Time); также встречается термин GMT, то есть «среднее время по Гринвичу» (Greenwich Mean Time), поскольку за точку отсчета принят Гринвич, Англия. Преобразования между системным и местным временем в Windows осуществляются при помощи поправок для местного часового пояса, заданного в системе. Функции Win32 позволяют работать как в местном, так и в системном времени и преобразовывать их по мере необходимости. Win32 также включает ряд функций для работы с файловым временем и датой, то есть временем и датой файлов, хранящихся в файловой системе

Функции Windows, предназначенные для получения информации о времени, перечислены в таблице 2. Следует учитывать, что во внутреннем представлении системы время изменяется в тактах таймера, продолжительность которых может изменяться в зависимости от используемого процессора и версии операционной системы. Интервал измерения времени в Win32 обычно занимает от 10 до 15 миллисекунд. Длительность такта определяет точность результатов, возвращаемых этими функциями.

Таблица 2. Основные функции Windows для работы со временем

Функция	Описание
EnumCalendarlnfo	Перечисляет календарную информацию, зависящую от локального контекста
EnuinDate Formats	Перечисляет форматы даты, доступные в заданном локальном контексте
EnumTimeFormats	Перечисляет форматы времени, доступные в заданном локальном контексте
GetLocalTime	Получает текущее местное время
GetMessageTime	Возвращает время (в миллисекундах) поступления последнего сообщения s очередь приложения. Время отсчитывается от начала текущего сеанса работы в Windows
GetSystemTime	Получает текущее системное время
GetSystemTimeAdjustment	Определяет, применяется ли в системе периодическая поправка, повышающая точность отсчета системного времени
GetTickCount	Получает продолжительность работы текущего сеанса работы в Windows в миллисекундах
GetTimeFormat	Форматирует время в заданном локальном контексте
GetTimeZonelnformation	Получает информацию о текущем часовом поясе
SetLocalTime	Задает местное время
SetSystemTime	Задает системное время
SetSystemTimeAdjustment	Задает периодическую поправку, применяемую системой для повышения точности отсчета времени
SetTimeZonelnformation	Задает часовой пояс
SystemTimeToTzSpecificLocalTime	Преобразует системное время в местное

СОДЕРЖАНИЕ ОТЧЕТА

1. Наименование лабораторной работы, ее цель.
2. Разработанное программное обеспечение приложения, обеспечивающего получение следующей системной информации:
   - Имя компьютера, имя пользователя;
   - Пути к системным каталогам Windows;
   - Версия операционной системы;
   - Системные метрики (не менее 2 метрик);
   - Системные параметры (не менее 2 параметров);
   - Системные цвета (определить цвет для некоторых символьных констант и изменить его на любой другой);
   - Функции для работы со временем;
   - Дополнительные API-функции:

То, про что я говорила :)
Старые задания:

Вариант	Название API-функции
1	ActivateKeyboardLayout, GetCurrencyFormat, GetLastError, OemToChar
2	CharToOem, GetCursor, GetLocaleInfo, OemToCharBuff,
3	AnsiToOemBuff, GetCursorPos, GetNumberFormat, SetCaretPos,
4	ClipCursor, GetDoubleClickTime, GetOEMCP, SetCursor
5	CreateCaret, GetEnvironmentStrings, GetQueueStatus, SetCursorPos
6	DestroyCaret, GetEnvironmentVariable, GetSystemDefaultLangID, SetDoubleClickTime
7	EnumSystemCodePages, GetInputState, GetLastError, SetKeyboardState
8	ExitWindowsEx, GetKBCodePage, GetSystemDefaultLCID, SetCaretBlinkTime
9	GetACP, GetKeyboardLayout, GetSystemPowerStatus, SetComputerName
10	GetAsyncKeyState, GetKeyboardLayoutList, GetTickCount, SetLocaleInfo
11	GetCaretBlinkTime, GetKeyboardLayoutName, GetLastError, SetSystemCursor
12	GetCaretPos, GetKeyboardState, GetUserDefaultLangID, SetSystemPowerState
13	GetClipCursor, GetKeyboardType, GetUserDefaultLCID, ShowCursor
14	GetCommandLine, GetKeyNameText, MessageBeep, SwapMouseButton
15	GetCPInfo, GetKeyState, GetLastError, UnloadKeyboardLayout

3. Примеры разработанных приложений (результаты и тексты программ).