Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №1 по теме: «Функции Win32 API для получения системной информации»

Выполнил:  
студент группы ПВ-31  
Адаменко И. И.

Проверил:  
к. т. н., доцент  
Михелёв В. М.

Белгород  
2014

Цель работы: получение практических навыков по программированию в Win32 API с использованием аппаратных и системных функций.

## Задание

Разработать приложение, обеспечивающее получение следующей системной информации:

1. Имя компьютера, имя пользователя;
2. Пути к системным каталогам Windows;
3. Версия операционной системы;
4. Системные метрики (не менее 50 метрик);
5. Системные параметры: SPI\_GETACCESSTIMEOUT, SPI\_GETKEYBOARDPREF, SPI\_GETSCREENREADER, SPI\_SCREENSAVERRUNNING, SPI\_SETFILTERKEYS, SPI\_SETMINIMIZEDMETRICS;
6. Системные цвета (определить цвет для следующих символьных констант и изменить его на любой другой): COLOR\_3DDKSHADOW, COLOR\_BTNTEXT, COLOR\_ACTIVECAPTION;
7. Функции для работы со временем: GetSystemTime, GetTimeZonelnformation, EnumCalendarlnfo;
8. Дополнительные API-функции: ActivateKeyboardLayout, GetCurrencyFormat, GetLastError, OemToChar.

## Отчёт

API — это интерфейс программного программирования, который используется для управления какой-либо системой. Win32 API используется для управления операционной системой Windows. API состоит из подпрограмм и функций, которые возвращают или устанавливают определённые параметры и характеристики системы. Поскольку Windows как система огромна, количество API-функций у неё подобающее, и исчисляется тысячами.

Все API-функции содержатся в DLL-файлах, т. е. в динамически подключаемых библиотеках. Эти файлы могут использовать различные приложения системы для доступа к возможностям системы. В данной лабораторной работе наиболее задействованными оказались:

* advapi32.dll — библиотека, содержащая функции, связанные с защитой объектов и работой с реестром;
* kernel32.dll — библиотека, содержащая функции для управления памятью и потоками;
* user32.dll — библиотека, содержащая функции для управления пользовательским интерфейсом.

Код программы поделён на файлы, каждый из которых определяет за реализацию отдельного «блока» подпрограмм или подключения API-функций из конкретного DLL-файла.

## Код программы

### DllImport\Advapi32.cs

Файл, отвечающий за подключение API-функций из библиотеки advapi32.dll:

|  |  |
| --- | --- |
| 01 | class Advapi32 { |
| 02 | [DllImport("Advapi32")] |
| 03 | public static extern bool GetUserName(System.Text.StringBuilder sb, ref Int32 length); |
| 04 | } |

### DllImport\Kernel32.cs

Файл, отвечающий за подключение API-функций из библиотеки kernel32.dll:

|  |  |
| --- | --- |
| 01 | class Kernel32 { |
| 02 | [DllImport("Kernel32")] |
| 03 | public static extern unsafe bool GetComputerName(byte\* lpBuffer, long\* nSize); |
| 04 |  |
| 05 | [DllImport("Kernel32")] |
| 06 | public static extern uint GetWindowsDirectory(StringBuilder lpBuffer, uint uSize); |
| 07 |  |
| 08 | [DllImport("Kernel32")] |
| 09 | public static extern uint GetSystemDirectory(  [Out] StringBuilder lpBuffer, uint uSize); |
| 10 |  |
| 11 | [DllImport("Kernel32")] |
| 12 | public static extern uint GetTempPath(uint nBufferLength,  [Out] StringBuilder lpBuffer); |
| 13 |  |
| 14 | [DllImport("Kernel32")] |
| 15 | public static extern bool GetVersionEx(ref OSVersionInfo info); |
| 16 |  |
| 17 | public struct OSVersionInfo { /\* struct description was here \*/ } |
| 18 |  |
| 19 | [StructLayout(LayoutKind.Sequential)] |
| 20 | public struct SystemTime { |
| 21 | // struct description is here |
| 22 |  |
| 23 | public SystemTime(DateTime dt) { /\* constr. description was here \*/ } |
| 24 | } |
| 25 |  |
| 26 | [DllImport("Kernel32")] |
| 27 | public static extern void GetSystemTime(out SystemTime lpSystemTime); |
| 28 |  |
| 29 | [DllImport("Kernel32", CharSet = CharSet.Auto)] |
| 30 | public static extern int GetTimeZoneInformation(out TimeZoneInformation lpTimeZoneInformation); |
| 31 |  |
| 32 | [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)] |
| 33 | public struct TimeZoneInformation { /\* struct description is here \*/ } |
| 34 |  |
| 35 | [DllImport("Kernel32", SetLastError = true)] |
| 36 | public static extern int EnumCalendarInfo(int lpCalInfoEnumProc,  int Locale, int Calendar, int CalType); |
| 37 |  |
| 38 | [DllImport("Kernel32")] |
| 39 | public static extern int GetCurrencyFormat(uint Locale, uint dwFlags, string lpValue, CurrencyFormat lpFormat,  IntPtr lpCurrencyStr, int cchCurrency); |
| 40 |  |
| 41 | [StructLayout(LayoutKind.Sequential)] |
| 42 | public class CurrencyFormat { |
| 43 | // struct description is here |
| 44 |  |
| 45 | public CurrencyFormat(uint nd, uint lz, uint g, string ds, string ts,  uint no, uint po, string cs) |
| 46 | { /\* constructor description was here \*/ } |
| 47 | }; |
| 48 |  |
| 49 | [DllImport("Kernel32")] |
| 50 | public static extern uint GetLastError(); |
| 51 | } |

### DllImport\User32.cs

Файл, отвечающий за подключение API-функций из библиотеки user32.dll:

|  |  |
| --- | --- |
| 01 | class User32 { |
| 02 | [DllImport("User32")] |
| 03 | public static extern int GetSystemMetrics(int smIndex); |
| 04 |  |
| 05 | [DllImport("User32", CharSet = CharSet.Auto)] |
| 06 | public static extern bool SystemParametersInfo(int uiAction, int uiParam,  IntPtr pvParam, int fuWinIni); |
| 07 |  |
| 08 | [DllImport("User32")] |
| 09 | public static extern uint GetSysColor(int nIndex); |
| 10 |  |
| 11 | [DllImport("User32")] |
| 12 | public static extern int SetSysColors(int cElements, Int32[] lpaElements,  Int32[] lpaRgbValues); |
| 13 |  |
| 14 | [DllImport("User32", CharSet = CharSet.Auto, ExactSpelling = true)] |
| 15 | public static extern IntPtr ActivateKeyboardLayout(int hkl, uint uFlags); |
| 16 |  |
| 17 | [DllImport("User32")] |
| 18 | public static extern bool OemToChar(string lpszSrc,  [Out] StringBuilder lpszDst); |
| 19 |  |
| 20 | [DllImport("User32")] |
| 21 | public static extern bool CharToOem(string lpszSrc,  [Out] StringBuilder lpszDst); |
| 22 | } |

### GetMain.cs

Файл, отвечающий за получение значений основных API-параметров лабораторной работы (имя компьютера, имя пользователя и пр.):

|  |  |
| --- | --- |
| 001 | class GetMain { |
| 002 | public static string ComputerName() { |
| 003 | long nSize = 512; |
| 004 | byte[] buffor = new byte[nSize]; |
| 005 | unsafe { |
| 006 | long\* pSize = &nSize; |
| 007 | fixed (byte\* pBuffor = buffor) { |
| 008 | Kernel32.GetComputerName(pBuffor, pSize); |
| 009 | } |
| 010 | } |
| 011 |  |
| 012 | return Encoding.Default.GetString(buffor.Reverse()  .SkipWhile(x => x == 0).Reverse().ToArray()); |
| 013 | } |
| 014 |  |
| 015 | public static string UserName() { |
| 016 | int nSize = 64; |
| 017 | StringBuilder buffer = new StringBuilder(nSize); |
| 018 | Advapi32.GetUserName(buffer, ref nSize); |
| 019 |  |
| 020 | return buffer.ToString(); |
| 021 | } |
| 022 |  |
| 023 | public static string WindowsDirectory() { |
| 024 | int nSize = 100; |
| 025 | StringBuilder sb = new StringBuilder(nSize); |
| 026 | Kernel32.GetWindowsDirectory(sb, Convert.ToUInt32(nSize)); |
| 027 |  |
| 028 | return sb.ToString(); |
| 029 | } |
| 030 |  |
| 031 | public static string SystemDirectory() { |
| 032 | int nSize = 256; |
| 033 | StringBuilder sb = new StringBuilder(nSize); |
| 034 | Kernel32.GetSystemDirectory(sb, Convert.ToUInt32(nSize)); |
| 035 |  |
| 036 | return sb.ToString(); |
| 037 | } |
| 038 |  |
| 039 | public static string TempPath() { |
| 040 | int nSize = 256; |
| 041 | StringBuilder sb = new StringBuilder(nSize); |
| 042 | Kernel32.GetTempPath(Convert.ToUInt32(nSize), sb); |
| 043 |  |
| 044 | return sb.ToString(); |
| 045 | } |
| 046 |  |
| 047 | public static string WindowsVersion() { |
| 048 | const uint VER\_NT\_WORKSTATION = 0x0000001; |
| 049 |  |
| 050 | Kernel32.OSVersionInfo osvi = new Kernel32.OSVersionInfo(); |
| 051 | osvi.OSVersionInfoSize = (uint)Marshal.SizeOf(osvi); |
| 052 |  |
| 053 | Kernel32.GetVersionEx(ref osvi); |
| 054 |  |
| 055 | string OS = "Windows"; |
| 056 | switch (osvi.MajorVersion) { |
| 057 | case 5: |
| 058 | switch (osvi.MinorVersion) { |
| 059 | case 0: |
| 060 | OS = "Windows 2000"; |
| 061 | break; |
| 062 | case 1: |
| 063 | OS = "Windows XP"; |
| 064 | break; |
| 065 | case 2: |
| 066 | OS = "Windows Server 2003"; |
| 067 | break; |
| 068 | } |
| 069 | break; |
| 070 | case 6: |
| 071 | switch (osvi.MinorVersion) { |
| 072 | case 0: |
| 073 | if (osvi.ProductType == VER\_NT\_WORKSTATION) |
| 074 | { OS = "Windows Vista"; } |
| 075 | else { OS = "Windows Server 2008";} |
| 076 | break; |
| 077 | case 1: |
| 078 | if (osvi.ProductType == VER\_NT\_WORKSTATION) |
| 079 | { OS = "Windows 7"; } |
| 080 | else { OS = "Windows Server 2008 R2"; } |
| 081 | break; |
| 082 | case 2: |
| 083 | if (osvi.ProductType == VER\_NT\_WORKSTATION) |
| 084 | { OS = "Windows 8"; } |
| 085 | else { OS = "Windows Server 2012"; } |
| 086 | break; |
| 087 | case 3: |
| 088 | if (osvi.ProductType == VER\_NT\_WORKSTATION) |
| 089 | { OS = "Windows 8.1"; } |
| 090 | else { OS = "Windows Server 2012 R2"; } |
| 091 | break; |
| 092 | } |
| 093 | break; |
| 094 | **}** |
| 095 |  |
| 096 | return OS + " " + osvi.MajorVersion + "." + osvi.MinorVersion  + "." + osvi.BuildNumber + " " + osvi.CSDVersion; |
| 097 | } |
| 098 | } |

### GetSystemMetrics.cs

Файл, отвечающий за получение значений системных метрик:

|  |  |
| --- | --- |
| 01 | class GetMetrics { |
| 02 | public static string Metric(int number) { |
| 03 | return User32.GetSystemMetrics(number).ToString(); |
| 04 | } |
| 05 | } |

### GetSystemParameters.cs

Файл, отвечающий за получение значений системных параметров:

|  |  |
| --- | --- |
| 01 | class GetParameters { |
| 02 | public struct AccessTimeOut { /\* struct description is here \*/ } |
| 03 |  |
| 04 | public static string Parameter(int number) { |
| 05 | AccessTimeOut ato = new AccessTimeOut(); |
| 06 | int size = ato.cbSize = Marshal.SizeOf(typeof(AccessTimeOut)); |
| 07 |  |
| 08 | IntPtr metrics = Marshal.AllocHGlobal(size); |
| 09 | Marshal.StructureToPtr(ato, metrics, true); |
| 10 |  |
| 11 | bool b = User32.SystemParametersInfo(number, size, metrics, 0); |
| 12 | AccessTimeOut atoin = (AccessTimeOut)Marshal.PtrToStructure(metrics,  typeof(AccessTimeOut)); |
| 13 | Marshal.FreeHGlobal(metrics); |
| 14 |  |
| 15 | return atoin.TimeOutMSec.ToString(); |
| 16 | } |
| 17 |  |
| 18 | public static MainWindow.FilterKeyStruct FilterKey(){ |
| 19 | MainWindow.FilterKeyStruct startupFilterKeys =  new MainWindow.FilterKeyStruct(); |
| 20 | int size = startupFilterKeys.cbSize =   Marshal.SizeOf(typeof(MainWindow.FilterKeyStruct)); |
| 21 |  |
| 22 | IntPtr metrics = Marshal.AllocHGlobal(size); |
| 23 | Marshal.StructureToPtr(startupFilterKeys, metrics, true); |
| 24 |  |
| 25 | bool b = User32.SystemParametersInfo(50, size, metrics, 0); |
| 26 | MainWindow.FilterKeyStruct result =   (MainWindow.FilterKeyStruct)Marshal.PtrToStructure(metrics,   typeof(MainWindow.FilterKeyStruct)); |
| 27 | Marshal.FreeHGlobal(metrics); |
| 28 |  |
| 29 | return result; |
| 30 | } |
| 31 |  |
| 32 | public static MainWindow.MinimizedMetricsStruct MinimizedMetrics(){ |
| 33 | MainWindow.MinimizedMetricsStruct mm = new   MainWindow.MinimizedMetricsStruct(); |
| 34 | int size = mm.cbSize =   Marshal.SizeOf(typeof(MainWindow.MinimizedMetricsStruct)); |
| 35 |  |
| 36 | IntPtr metrics = Marshal.AllocHGlobal(size); |
| 37 | Marshal.StructureToPtr(mm, metrics, true); |
| 38 |  |
| 39 | bool b = User32.SystemParametersInfo(43, size, metrics, 0); |
| 40 | MainWindow.MinimizedMetricsStruct result =   (MainWindow.MinimizedMetricsStruct)Marshal.PtrToStructure(metrics,   typeof(MainWindow.MinimizedMetricsStruct)); |
| 41 | Marshal.FreeHGlobal(metrics); |
| 42 |  |
| 43 | return result; |
| 44 | } |
| 45 | } |

### GetColors.cs

Файл, отвечающий за получение значений системных цветов:

|  |  |
| --- | --- |
| 01 | class GetColors { |
| 02 | public static Color Element(int n) { |
| 03 | uint value = User32.GetSysColor(n); |
| 04 | return Color.FromArgb((byte)((value >> 24) & 0xFF),   (byte)(value & 0xFF), (byte)((value >> 8) & 0xFF),  (byte)((value >> 16) & 0xFF)); |
| 05 | } |
| 06 | } |

### GetTime.cs

Файл, отвечающий за получение значений системного времени:

|  |  |
| --- | --- |
| 01 | class GetTime { |
| 02 | public static string SystemTime(){ |
| 03 | Kernel32.SystemTime st; |
| 04 | Kernel32.GetSystemTime(out st); |
| 05 | return "Сегодня " + FormatDate(st) + ". Текущее время по UTC: "  + FormatTime(st); |
| 06 | } |
| 07 |  |
| 08 | public static string FormatDate(Kernel32.SystemTime st) { |
| 09 | return String.Format("{0,2}", st.Day.ToString("D2")) + "." +   String.Format("{0,2}", st.Month.ToString("D2")) +  "." + st.Year; |
| 10 | } |
| 11 |  |
| 12 | public static string FormatTime(Kernel32.SystemTime st) { |
| 13 | return String.Format("{0,2}", st.Hour.ToString("D2")) + ":" +   String.Format("{0,2}", st.Minute.ToString("D2")) + ":" +   String.Format("{0,2}", st.Second.ToString("D2")); |
| 14 | } |
| 15 |  |
| 16 | public static Kernel32.TimeZoneInformation TimeZone() { |
| 17 | Kernel32.TimeZoneInformation tzi; |
| 18 | int currentTimeZone = Kernel32.GetTimeZoneInformation(out tzi); |
| 19 |  |
| 20 | return tzi; |
| 21 | } |
| 22 |  |
| 23 | public static String Bias() { |
| 24 | Kernel32.TimeZoneInformation t = TimeZone(); |
| 25 | Kernel32.SystemTime st; |
| 26 | Kernel32.GetSystemTime(out st); |
| 27 | int utc = (-(t.bias / 60 - (t.standardDate.Month > st.Month &&  st.Month < t.daylightDate.Month ? 1 : 0))); |
| 28 |  |
| 29 | return "Смещение относительно UTC: " + (utc > 0 ? "+" : "") +   utc.ToString() +" ч."; |
| 30 | } |
| 31 |  |
| 32 | public static String ShiftingTime(bool isWinter) { |
| 33 | Kernel32.TimeZoneInformation t = TimeZone(); |
| 34 | Kernel32.SystemTime to = isWinter ? t.standardDate : t.daylightDate; |
| 35 | Kernel32.SystemTime st; |
| 36 | Kernel32.GetSystemTime(out st); |
| 37 | if (st.Month > to.Month) st.Year++; |
| 38 |  |
| 39 | short days = 31; |
| 40 | Kernel32.SystemTime t1 = new Kernel32.SystemTime(new   DateTime(st.Year, to.Month, days)); |
| 41 | while (t1.DayOfWeek != to.DayOfWeek) { |
| 42 | days--; |
| 43 | t1 = new Kernel32.SystemTime(new DateTime(st.Year,   to.Month, days, to.Hour, to.Minute, to.Second)); |
| 44 | } |
| 45 |  |
| 46 | return "Переход на " + (isWinter ? "зимнее" : "летнее") + " время: "   + FormatDate(t1) + " в " + FormatTime(t1); |
| 47 | } |
| 48 | } |

### GetAdditional.cs

Файл, отвечающий за получение значений дополнительных API-функций:

|  |  |
| --- | --- |
| 01 | class GetAdditional { |
| 02 | public static string CurrencyFormat(double numberToConvert, uint nd,   uint g, string ds, string ts) { |
| 03 | string sCurrency = null; |
| 04 | long lRet = 0; |
| 05 | Kernel32.CurrencyFormat cf = new Kernel32.CurrencyFormat(nd, 1, g,  ds, ts, 2, 2, ""); |
| 06 |  |
| 07 | lRet = Kernel32.GetCurrencyFormat(0, 0,   numberToConvert.ToString(), cf, IntPtr.Zero, 0); |
| 08 |  |
| 09 | if (0 != lRet) { |
| 10 | IntPtr ptrCurrencyStr = Marshal.AllocHGlobal((int)lRet); |
| 11 | lRet = Kernel32.GetCurrencyFormat(0, 0,   numberToConvert.ToString(), cf,  ptrCurrencyStr, (int)lRet); |
| 12 | sCurrency = 0 == lRet ? null :   Marshal.PtrToStringAnsi(ptrCurrencyStr); |
| 13 | Marshal.FreeHGlobal(ptrCurrencyStr); |
| 14 |  |
| 15 | return sCurrency; |
| 16 | } else { return "Ошибка!"; } |
| 17 | } |
| 18 |  |
| 19 | [DllImport("kernel32.dll", SetLastError = true)] |
| 20 | static extern bool SetVolumeLabel(string lpRootPathName,  string lpVolumeName); |
| 21 |  |
| 22 | public static uint LastError(){ |
| 23 | SetVolumeLabel("XYZ:\\", "My Imaginary Drive"); |
| 24 |  |
| 25 | return Kernel32.GetLastError(); |
| 26 | } |
| 27 |  |
| 28 | public static string OemToChar(string str) { |
| 29 | if (string.IsNullOrEmpty(str)) return str; |
| 30 |  |
| 31 | StringBuilder strBuilder = new StringBuilder(str.Length); |
| 32 | User32.OemToChar(str.ToString(), strBuilder); |
| 33 |  |
| 34 | return strBuilder.ToString(); |
| 35 | } |
| 36 |  |
| 37 | public static string CharToOem(string str) { |
| 38 | if (string.IsNullOrEmpty(str)) return str; |
| 39 |  |
| 40 | StringBuilder strBuilder = new StringBuilder(str.Length); |
| 41 | User32.CharToOem(str.ToString(), strBuilder); |
| 42 |  |
| 43 | return strBuilder.ToString(); |
| 44 | } |
| 45 | } |

### SetColors.cs

Файл, отвечающий за установку значений системных цветов:

|  |  |
| --- | --- |
| 01 | class SetColors { |
| 02 | public static void Elements(int n, int[] els, Color[] c) { |
| 03 | int[] colors = new int[3]; |
| 04 | for (int i = 0; i < c.Length; i++) |
| 05 | colors[i] = (int)((c[i].B << 16) | (c[i].G << 8) |   (c[i].R << 0)) & 0x00FFFFFF; |
| 06 |  |
| 07 | User32.SetSysColors(n, els, colors); |
| 08 | } |
| 09 | } |

### SetSystemParameters.cs

Файл, отвечающий за установку значений системных параметров:

|  |  |
| --- | --- |
| 01 | class SetParameters { |
| 02 | public static void FilterKey(MainWindow.FilterKeyStruct fk) { |
| 03 | int size = fk.cbSize; |
| 04 |  |
| 05 | IntPtr metrics = Marshal.AllocHGlobal(size); |
| 06 | Marshal.StructureToPtr(fk, metrics, true); |
| 07 |  |
| 08 | User32.SystemParametersInfo(51, size, metrics, 0); |
| 09 | } |
| 10 |  |
| 11 | public static void MinimizedMetrics(MainWindow.MinimizedMetricsStruct mm) { |
| 12 | int size = mm.cbSize; |
| 13 |  |
| 14 | IntPtr metrics = Marshal.AllocHGlobal(size); |
| 15 | Marshal.StructureToPtr(mm, metrics, true); |
| 16 |  |
| 17 | User32.SystemParametersInfo(44, size, metrics, 0); |
| 18 | } |
| 19 | } |

### MainWindow.xaml.cs

Файл, отвечающий за основную логику работы программы:

|  |  |
| --- | --- |
| 001 | public partial class MainWindow : Window { |
| 002 | private FilterKeyStruct tmp\_fk; |
| 003 | private FilterKeyStruct fk; |
| 004 | private MinimizedMetricsStruct tmp\_mm; |
| 005 | private MinimizedMetricsStruct mm; |
| 006 | private uint FKF\_FILTERKEYSON = 0x00000001; |
| 007 | private Dictionary<string, uint> valueFilterKeys =  new Dictionary<string, uint>(); |
| 008 | private int[] arColors = { 21, 18, 2 }; |
| 009 | private Color[] arColorsValues = new Color[3]; |
| 010 | private Color[] tmp\_arColorsValues = new Color[3]; |
| 011 | private DispatcherTimer timer = null; |
| 012 | private bool allRight = true; |
| 013 |  |
| 014 | public MainWindow() { |
| 015 | InitializeComponent(); |
| 016 |  |
| 017 | #region Main |
| 018 | var main = new ObservableCollection<DataObject>(); |
| 019 |  |
| 020 | main.Add(new DataObject() { Name = "Имя компьютера",  Value = GetMain.ComputerName() }); |
| 021 | main.Add(new DataObject() { Name = "Имя пользователя",  Value = GetMain.UserName() }); |
| 022 | main.Add(new DataObject() { Name = "Путь к каталогу Windows",   Value = GetMain.WindowsDirectory() }); |
| 023 | main.Add(new DataObject() { Name = "Путь к системному каталогу",   Value = GetMain.SystemDirectory() }); |
| 024 | main.Add(new DataObject() {  Name = "Путь к каталогу временных файлов",   Value = GetMain.TempPath() }); |
| 025 | main.Add(new DataObject() { Name = "Версия операционной системы",   Value = GetMain.WindowsVersion() }); |
| 026 |  |
| 027 | this.mainGrid.ItemsSource = main; |
| 028 | #endregion |
| 029 |  |
| 030 | #region System Metrics |
| 031 | string px = " пикс."; |
| 032 | var metrics = new ObservableCollection<DataObject>(); |
| 033 |  |
| 034 | metrics.Add(new DataObject() { Name = "Ширина экрана",  Value = GetMetrics.Metric(0) + px }); |
| 035 | metrics.Add(new DataObject() { Name = "Высота экрана",  Value = GetMetrics.Metric(1) + px }); |
| 036 | metrics.Add(new DataObject() {  Name = "Ширина вертикальной прокрутки", Value = GetMetrics.Metric(2) + px }); |
| 037 | metrics.Add(new DataObject() {  Name = "Высота горизонтальной прокрутки", Value = GetMetrics.Metric(3) + px }); |
| 038 | metrics.Add(new DataObject() { Name = "Высота заголовка окна",  Value = GetMetrics.Metric(4) + px }); |
| 039 | metrics.Add(new DataObject() { Name = "Ширина границы окна", Value = GetMetrics.Metric(5) + px }); |
| 040 | metrics.Add(new DataObject() { Name = "Высота границы окна", Value = GetMetrics.Metric(6) + px }); |
| 041 | metrics.Add(new DataObject() {  Name = "Ширина рамки диалогового окна", Value = GetMetrics.Metric(7) + px }); |
| 042 | metrics.Add(new DataObject() {  Name = "Высота рамки диалогового окна", Value = GetMetrics.Metric(8) + px }); |
| 043 | metrics.Add(new DataObject() {  Name = "Высота вертикального бегунка прокрутки", Value = GetMetrics.Metric(9) + px }); |
| 044 | metrics.Add(new DataObject() {  Name = "Ширина горизонтального бегунка прокрутки", Value = GetMetrics.Metric(10) + px }); |
| 045 | metrics.Add(new DataObject() { Name = "Ширина значка", Value = GetMetrics.Metric(11) + px }); |
| 046 | metrics.Add(new DataObject() { Name = "Высота значка", Value = GetMetrics.Metric(12) + px }); |
| 047 | metrics.Add(new DataObject() { Name = "Ширина курсора", Value = GetMetrics.Metric(13) + px }); |
| 048 | metrics.Add(new DataObject() { Name = "Высота курсора", Value = GetMetrics.Metric(14) + px }); |
| 049 | metrics.Add(new DataObject() {  Name = "Высота отдельной строки меню", Value = GetMetrics.Metric(15) + px }); |
| 050 | metrics.Add(new DataObject() {  Name = "Ширина полноэкранного окна клиента", Value = GetMetrics.Metric(16) + px }); |
| 051 | metrics.Add(new DataObject() {  Name = "Высота полноэкранного окна клиента", Value = GetMetrics.Metric(17) + px }); |
| 052 | metrics.Add(new DataObject() { Name = "Статус мыши", Value = (GetMetrics.Metric(19) == "0"   ? "не " : "") + "подключена" }); |
| 053 | metrics.Add(new DataObject() {  Name = "Высота стрелки на вертикальной прокрутке", Value = GetMetrics.Metric(20) + px }); |
| 054 | metrics.Add(new DataObject() {   Name = "Ширина стрелки на горизонтальной прокрутке",   Value = GetMetrics.Metric(21) + px }); |
| 055 | metrics.Add(new DataObject() {  Name = "Статус отладочной версии USER.EXE", Value = (GetMetrics.Metric(22) == "0" ?   "не " : "") + "установлена" }); |
| 056 | metrics.Add(new DataObject() {  Name = "Статус левой и правой кнопок мыши", Value = (GetMetrics.Metric(23) == "0" ?   "не " : "") + "поменялись" }); |
| 057 | metrics.Add(new DataObject() { Name = "Минимальная ширина окна", Value = GetMetrics.Metric(28) + px }); |
| 058 | metrics.Add(new DataObject() { Name = "Минимальная высота окна", Value = GetMetrics.Metric(29) + px }); |
| 059 | metrics.Add(new DataObject() { Name = "Ширина кнопки заголовка", Value = GetMetrics.Metric(30) + px }); |
| 060 | metrics.Add(new DataObject() { Name = "Высота кнопки заголовка", Value = GetMetrics.Metric(31) + px }); |
| 061 | metrics.Add(new DataObject() {  Name = "Ширина границы окна изменяемого размера", Value = GetMetrics.Metric(32) + px }); |
| 062 | metrics.Add(new DataObject() {  Name = "Высота границы окна изменяемого размера", Value = GetMetrics.Metric(33) + px }); |
| 063 | metrics.Add(new DataObject() {  Name = "Ширина прямоугольника для двойного клика", Value = GetMetrics.Metric(36) + px }); |
| 064 | metrics.Add(new DataObject() {  Name = "Высота прямоугольника для двойного клика", Value = GetMetrics.Metric(37) + px }); |
| 065 | metrics.Add(new DataObject() {  Name = "Ширина ячейки для позиционирования значков",   Value = GetMetrics.Metric(38) + px }); |
| 066 | metrics.Add(new DataObject() {  Name = "Высота ячейки для позиционирования значков",   Value = GetMetrics.Metric(39) + px }); |
| 067 | metrics.Add(new DataObject() { Name = "Положение всплывающего меню", Value = (GetMetrics.Metric(40) == "0" ?   "слева" : "справа") }); |
| 068 | metrics.Add(new DataObject() {  Name = "Набор символов с двойным байтом USER.EXE", Value = (GetMetrics.Metric(42) == "0" ?   "не " : "") + "установлен" }); |
| 069 | metrics.Add(new DataObject() { Name = "Защита", Value = (GetMetrics.Metric(44) == "0" ?   "отсутствует или неактивна" :   "присутствует и активна") }); |
| 070 | metrics.Add(new DataObject() { Name = "Подключение к сети", Value = ((Convert.ToInt32(GetMetrics.Metric(63)) & 0x1) == 0 ?   "отсутствует" : "присутствует") }); |
| 071 | metrics.Add(new DataObject() {  Name = "Визуальная команда вызова программы", Value = (GetMetrics.Metric(70) == "0" ?   "не " : "") + "показывается" }); |
| 072 | metrics.Add(new DataObject() { Name = "Процессор", Value = (GetMetrics.Metric(73) == "0" ?   "быстрый" : "медленный") }); |
| 073 | metrics.Add(new DataObject() { Name = "Еврейские или арабские языки", Value = (GetMetrics.Metric(74) == "0" ?   "не " : "") + "используются" }); |
| 074 | metrics.Add(new DataObject() {  Name = "Мышь с колесом для вертикальной прокрутки", Value = (GetMetrics.Metric(75) == "0" ?   "не " : "") + "используется" }); |
| 075 | metrics.Add(new DataObject() {  Name = "Мышь с колесом для горизонтальной прокрутки",   Value = (GetMetrics.Metric(91) == "0" ?   "не " : "") + "используется" }); |
| 076 | metrics.Add(new DataObject() { Name = "Ширина виртуального экрана", Value = GetMetrics.Metric(78) + px }); |
| 077 | metrics.Add(new DataObject() { Name = "Высота виртуального экрана", Value = GetMetrics.Metric(79) + px }); |
| 078 | metrics.Add(new DataObject() {  Name = "Количество используемых мониторов", Value = GetMetrics.Metric(80) }); |
| 079 | metrics.Add(new DataObject() { Name = "Цветовая модель мониторов", Value = (GetMetrics.Metric(81) == "0" ?   "различная" : "одинаковая у всех") }); |
| 080 | metrics.Add(new DataObject() { Name = "Статус сервиса IMM/IME", Value = (GetMetrics.Metric(82) == "0" ?   "не " : "") + "запущен" }); |
| 081 | metrics.Add(new DataObject() { Name = "Статус сеанса", Value = (GetMetrics.Metric(0x2000) == "0" ?   "активен" : "завершается") }); |
| 082 |  |
| 083 | this.systemGrid.ItemsSource = metrics; |
| 084 | #endregion |
| 085 |  |
| 086 | #region System parameters |
| 087 | var parameters = new ObservableCollection<DataObject>(); |
| 088 | parameters.Add(new DataObject() { Name = "Время ожидания", Value = GetParameters.Parameter(60) + " мс" }); |
| 089 | parameters.Add(new DataObject() { Name = "Клавиатура вместо мыши", Value = (GetParameters.Parameter(68) == "0" ?   "не " : "") + "используется" }); |
| 090 | parameters.Add(new DataObject() { Name = "Экранный диктор", Value = (GetParameters.Parameter(70) == "0" ?   "не " : "") + "используется" }); |
| 091 |  |
| 092 | this.parametersGrid.ItemsSource = parameters; |
| 093 |  |
| 094 | tmp\_fk = fk = GetParameters.FilterKey(); |
| 095 |  |
| 096 | InitFilter(); |
| 097 |  |
| 098 | tmp\_mm = mm = GetParameters.MinimizedMetrics(); |
| 099 | InitMinimizedMetrics(); |
| 101 | #endregion |
| 102 |  |
| 103 | #region Colors |
| 104 | arColorsValues[0] = GetColors.Element(arColors[0]); |
| 105 | arColorsValues[1] = GetColors.Element(arColors[1]); |
| 106 | arColorsValues[2] = GetColors.Element(arColors[2]); |
| 107 | arColorsValues.CopyTo(tmp\_arColorsValues, 0); |
| 108 |  |
| 109 | InitColors(); |
| 110 | #endregion |
| 111 |  |
| 112 | #region Time info |
| 113 | timer = new DispatcherTimer(); |
| 114 | timer.Tick += (sender, e) =>   { currentTime.Content = GetTime.SystemTime(); }; |
| 115 | timer.Interval = new TimeSpan(0, 0, 0, 1); |
| 116 | timer.Start(); |
| 117 | biasTime.Content = GetTime.Bias(); |
| 118 | summerTime.Content = GetTime.ShiftingTime(false); |
| 119 | winterTime.Content = GetTime.ShiftingTime(true); |
| 120 | #endregion |
| 121 |  |
| 122 | #region Additional |
| 123 | resultNumber.Content = GetAdditional.CurrencyFormat(   Convert.ToDouble(number.Text), Convert.ToUInt32(digits.Text),   Convert.ToUInt32(grouping.Text), decimalsep.Text,   thousandsep.Text); |
| 124 |  |
| 125 | errorCode.Content = GetAdditional.LastError().ToString(); |
| 126 | #endregion |
| 127 | } |
| 128 |  |
| 129 | [StructLayout(LayoutKind.Sequential)] |
| 130 | public struct MinimizedMetricsStruct { /\* struct description was here \*/ } |
| 131 |  |
| 132 | [Flags] |
| 133 | public enum MinimizedMetricsArrangement { /\* enum description was here \*/ } |
| 134 |  |
| 135 | [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)] |
| 136 | public struct FilterKeyStruct { /\* struct description was here \*/ } |
| 137 |  |
| 138 | public class DataObject { |
| 139 | public string Name { get; set; } |
| 140 | public string Value { get; set; } |
| 141 | } |
| 142 |  |
| 143 | private void InitColors() { |
| 144 | shadowColor.SelectedColor = arColorsValues[0]; |
| 145 | commandKeysColor.SelectedColor = arColorsValues[1]; |
| 146 | captionColor.SelectedColor = arColorsValues[2]; |
| 147 | } |
| 148 |  |
| 149 | private void InitMinimizedMetrics() { |
| 150 | enableMinimized.IsChecked = mm.iArrange.ToString().Contains("Hide"); |
| 151 | } |
| 152 |  |
| 153 | private void enableFilter\_Checked(object sender, RoutedEventArgs e) { |
| 154 | fk.dwFlags |= FKF\_FILTERKEYSON; |
| 155 | SetParameters.FilterKey(fk); |
| 156 | } |
| 157 |  |
| 158 | private void enableFilter\_Unchecked(object sender, RoutedEventArgs e) { |
| 159 | fk.dwFlags &= ~FKF\_FILTERKEYSON; |
| 160 | SetParameters.FilterKey(fk); |
| 161 | } |
| 162 |  |
| 163 | private void UpdateColors() { |
| 164 | SetColors.Elements(3, arColors, arColorsValues); |
| 165 | } |
| 166 |  |
| 167 | private void iWaitMSecFilter\_SelectionChanged(object sender,   SelectionChangedEventArgs e) { |
| 168 | if (!this.IsLoaded) return; |
| 169 | fk.iWaitMSec =   valueFilterKeys[iWaitMSecFilter.SelectedValue.ToString()]; |
| 170 | SetParameters.FilterKey(fk); |
| 171 | } |
| 172 |  |
| 173 | private void iDelayMSecFilter\_SelectionChanged(object sender,   SelectionChangedEventArgs e) { |
| 174 |  |
| 175 | if (!this.IsLoaded) return; |
| 176 | fk.iDelayMSec =   valueFilterKeys[iDelayMSecFilter.SelectedValue.ToString()]; |
| 177 | SetParameters.FilterKey(fk); |
| 178 | } |
| 179 |  |
| 180 | private void iRepeatMSecFilter\_SelectionChanged(object sender,   SelectionChangedEventArgs e) { |
| 181 |  |
| 182 | if (!this.IsLoaded) return; |
| 183 | fk.iRepeatMSec =   valueFilterKeys[iRepeatMSecFilter.SelectedValue.ToString()]; |
| 184 | SetParameters.FilterKey(fk); |
| 185 | } |
| 186 |  |
| 187 | private void Button\_Click(object sender, RoutedEventArgs e) { |
| 188 | SetParameters.FilterKey(tmp\_fk); |
| 189 | fk = tmp\_fk; |
| 190 | InitFilter(); |
| 191 | SetParameters.MinimizedMetrics(tmp\_mm); |
| 192 | } |
| 193 |  |
| 194 | private void InitFilter() { |
| 195 |  |
| 196 | if ((fk.dwFlags & FKF\_FILTERKEYSON) == 0) { |
| 197 | enableFilter.IsChecked = false; |
| 198 | } else { enableFilter.IsChecked = true; } |
| 199 |  |
| 200 | foreach (KeyValuePair<string, uint> pair in valueFilterKeys) { |
| 201 | if (fk.iWaitMSec.Equals(pair.Value)) |
| 202 | iWaitMSecFilter.SelectedValue = pair.Key; |
| 203 |  |
| 204 | if (fk.iDelayMSec.Equals(pair.Value)) |
| 205 | iDelayMSecFilter.SelectedValue = pair.Key; |
| 206 |  |
| 207 | if (fk.iRepeatMSec.Equals(pair.Value)) |
| 208 | iRepeatMSecFilter.SelectedValue = pair.Key; |
| 209 | } |
| 210 | } |
| 211 |  |
| 212 | private void enableMinimized\_Checked(object sender, RoutedEventArgs e) { |
| 213 | mm.iArrange = MinimizedMetricsArrangement.Hide; |
| 214 | SetParameters.MinimizedMetrics(mm); |
| 215 | } |
| 216 |  |
| 217 | private void enableMinimized\_Unchecked(object sender, RoutedEventArgs e) { |
| 218 | mm.iArrange = MinimizedMetricsArrangement.Left; |
| 219 | SetParameters.MinimizedMetrics(mm); |
| 220 | } |
| 221 |  |
| 222 | private void HideReset\_Click(object sender, RoutedEventArgs e) { |
| 223 | SetParameters.MinimizedMetrics(tmp\_mm); |
| 224 | mm = tmp\_mm; |
| 225 | InitMinimizedMetrics(); |
| 226 | } |
| 227 |  |
| 228 | private void Button\_Click\_1(object sender, RoutedEventArgs e) { |
| 229 | tmp\_arColorsValues.CopyTo(arColorsValues, 0); |
| 230 | UpdateColors(); |
| 231 | InitColors(); |
| 232 | } |
| 233 |  |
| 234 | private void shadowColor\_SelectedColorChanged(object sender,   RoutedPropertyChangedEventArgs<Color> e) { |
| 235 | arColorsValues[0] = shadowColor.SelectedColor; |
| 236 | UpdateColors(); |
| 237 | } |
| 238 |  |
| 239 | private void commandKeysColor\_SelectedColorChanged(object sender,   RoutedPropertyChangedEventArgs<Color> e) { |
| 240 | arColorsValues[1] = commandKeysColor.SelectedColor; |
| 241 | UpdateColors(); |
| 242 | } |
| 243 |  |
| 244 | private void captionColor\_SelectedColorChanged(object sender,   RoutedPropertyChangedEventArgs<Color> e) { |
| 245 | arColorsValues[2] = captionColor.SelectedColor; |
| 246 | UpdateColors(); |
| 247 | } |
| 248 |  |
| 249 |  |
| 250 | private void Button\_Click\_2(object sender, RoutedEventArgs e) { |
| 251 | User32.ActivateKeyboardLayout(0, 0); |
| 252 | } |
| 253 |  |
| 254 | private void Button\_Click\_3(object sender, RoutedEventArgs e) { |
| 255 | User32.ActivateKeyboardLayout(1, 0); |
| 256 | } |
| 257 |  |
| 258 | private void number\_KeyUp(object sender, KeyEventArgs e) { |
| 259 | TextBox t = (TextBox)sender; |
| 260 | allRight = true; |
| 261 |  |
| 262 | if (! (t.Text.Length == 0)) { |
| 263 | try { |
| 264 | if (t.Name == "digits" || t.Name == "grouping") { |
| 265 | uint result = Convert.ToUInt32(t.Text); |
| 266 | } else { double result = Convert.ToDouble(t.Text); } |
| 267 | } |
| 268 | catch (Exception ex) { |
| 269 | e.Handled = true; |
| 270 | allRight = false; |
| 271 | } |
| 272 |  |
| 273 | if (allRight) |
| 274 | resultNumber.Content = GetAdditional.CurrencyFormat(   Convert.ToDouble(number.Text),   Convert.ToUInt32(digits.Text),   Convert.ToUInt32(grouping.Text),   decimalsep.Text, thousandsep.Text); |
| 275 | allRight = true; |
| 276 | } |
| 277 | } |
| 278 |  |
| 279 | private void decimalsep\_KeyUp(object sender, KeyEventArgs e) { |
| 280 | if (allRight) |
| 281 | resultNumber.Content = GetAdditional.CurrencyFormat(   Convert.ToDouble(number.Text),   Convert.ToUInt32(digits.Text),   Convert.ToUInt32(grouping.Text),   decimalsep.Text, thousandsep.Text); |
| 282 | } |
| 283 |  |
| 284 | private void toOEM\_KeyUp(object sender, KeyEventArgs e) { |
| 285 | inOEM.Content = GetAdditional.CharToOem(toOEM.Text); |
| 286 | fromOEM.Content = GetAdditional.OemToChar(inOEM.Content.ToString()); |
| 287 | } |
| 288 | } |