|  |
| --- |
| **Лабораторная работа. Системный реестр Windows.** |
|  |  |
| **Цель работы**: изучить основные принципы работы с системным реестром в операционной системе Windows.    **Темы для предварительной проработки**: процессы, интерфейс прикладного программирования API WIN32.    **Теоретические сведения**    В системном реестре хранится информация о конфигурации системы в среде Win32. Структура системного реестра подобна структуре каталогов: эквивалентом каталога является ключ, а файла – значение. Ключ содержит подключи и значения. В целом системный реестр имеет древовидную структуру. Основными ветвями системного реестра являются:  HKEY\_CLASSES\_ROOT – содержит определения типов документов, связей с файлами и интерфейса командного процессора;  HKEY\_CURRENT\_USER – содержит параметры настройки текущего пользователя;  HKEY\_LOCAL\_MACHINE – хранит аппаратные конфигурации, сетевые протоколы и классы программного обеспечения;  HKEY\_USERS – используется для хранения выбранных пользователями глобальных параметров (цвет, звук и т.д.), а также параметры настройки рабочего стола;  HKEY\_CURRENT\_CONFIG – ключ устанавливает связь с ключом отображения, входящим в состав выбранной конфигурации config из HKEY\_LOCAL\_MACHINE;  HKEY\_PERFORMANCE\_DATA (Windows 2000 и выше) или HKEY\_DYN\_DATA (Windows 9x) – хранит данные о каждом аппаратном компоненте системы и данные о производительности системы.  Наиболее часто встречаемые типы значений в реестре: строковый (REG\_SZ), двоичный (REG\_BINARY), двойное слово (REG\_DWORD).  Функции для работы с реестром, используемые при выполнении индивидуальных заданий:  RegCloseKey – закрывает открытый ключ системного реестра  RegCreateKeyEx – создает новый подключ  RegDeleteKey – удаляет ключ из системного реестра  RegDeleteValue – удаляет значение из системного реестра  RegEnumKeyEx – перечисляет все ключи данного ключа  RegEnumValue – перечисляет все значения данного ключа  RegFlushKey – сразу же записывает все изменения, произведенные в системном реестре  RegNotifyChangeKeyValue – указывает на момент изменения ключа или значения в системном реестре  RegOpenKeyEx – открывает существующий ключ системного реестра  RegQueryInfoKey – возвращает информацию о ключе  RegQueryValueEx – возвращает значение ключа  RegSetValueEx – присваивает ключу значение    **ПРЕДУПРЕЖДЕНИЕ**: Никогда не изменяйте содержимое реестра, если не знаете его назначения!!! Некорректное изменение данных может привести к сбоям в работе операционной системы Windows!!!  ПРИМЕЧАНИЕ: В данной работе необходимо использовать только СВОИ уникальные ключи в реестре (**только в разделе HKEY\_CURRENT\_USER**), изменение или удаление существующих ключей ЗАПРЕЩЕНО!!!  Пример. Программа при отсутствии ключа в реестре создает его, при наличии – выводит все значения ключа и удаляет его.    #include <windows.h>  #include <stdio.h>    HKEY hk;  CHAR szBuf[80];  DWORD dwData, dwDsp, dwValues, dwMaxValueNameLen, dwMaxValueData,        dwNameLen, dwValueData, dwIndex, dwType;  LONG lReturn;  LPTSTR lpszVN1, lpszVN2;    int main()  {   //Создаем ключ в реестре  if (RegCreateKeyEx(  HKEY\_CURRENT\_USER, //ветвь                  "Software\\MyLabSPO", //ключ                  0, //зарезервировано (должно быть 0)                  "", // имя класса (нам не нужно)                  REG\_OPTION\_NON\_VOLATILE, // сохранить ключ на диске                  KEY\_ALL\_ACCESS, // полный доступ                  NULL, // защита по умолчанию                  &hk, //дескриптор нового ключа                  &dwDsp)) // состояние ключа (открыт, создан)  {       printf("Key Creation Error\n"); return(0);  }  if (dwDsp==REG\_OPENED\_EXISTING\_KEY)  {       printf("Key Found\n");        // закрытие дескриптора ключа       RegCloseKey(hk);        // открытие ключа        // приведено для примера - в данном случае достаточно        // проанализировать dwDsp==REG\_OPENED\_EXISTING\_KEY       if (RegOpenKeyEx(    HKEY\_CURRENT\_USER,                            "Software\\MyLabSPO",                            0,                            KEY\_ALL\_ACCESS,                            &hk))       {            printf("Key Open Error\n"); return(0);       }        // Определяем нужные параметры для чтения ключа        // задаем только нужные нам данные       RegQueryInfoKey(     hk,            NULL,            NULL,            NULL,            NULL,            NULL,            NULL,            &dwValues,  // число значений в ключе            &dwMaxValueNameLen, // максимальная длина имени значения            &dwMaxValueData, // максимальная длина значения            NULL,            NULL );       printf("Values:%d,Max Value Name Length:%d,Max Data Length:%d\n",            dwValues, dwMaxValueNameLen, dwMaxValueData);        // так как строки завершаются 0 - добавляем 1 байт       dwMaxValueNameLen++;       dwMaxValueData++;        // выделяем память для хранения строк       lpszVN1 = (LPTSTR)HeapAlloc( GetProcessHeap(), HEAP\_ZERO\_MEMORY, dwMaxValueNameLen );       lpszVN2 = (LPTSTR)HeapAlloc( GetProcessHeap(), HEAP\_ZERO\_MEMORY, dwMaxValueData );        // читаем значения ключа       do       {            dwNameLen = dwMaxValueNameLen;            dwValueData = dwMaxValueData;            lReturn = RegEnumValue( hk, dwIndex, lpszVN1,                            &dwNameLen, 0, &dwType,                            (unsigned char \*)lpszVN2, &dwValueData );            if (lReturn != ERROR\_NO\_MORE\_ITEMS )            {                  printf("%s = %s\n", lpszVN1, lpszVN2);                  dwIndex++;            }       }       while( lReturn != ERROR\_NO\_MORE\_ITEMS );        // освобождаем занятую ранее память       if ( lpszVN1 ) HeapFree( GetProcessHeap(), 0, lpszVN1 );       if ( lpszVN2 ) HeapFree( GetProcessHeap(), 0, lpszVN2 );        // удаляем параметры ключа       if (RegDeleteValue( hk, "File Name" ))            printf("Value 1 Delete Error\n");       if (RegDeleteValue( hk, "Application Run" ))            printf("Value 2 Delete Error\n");        // удаляем ключ       if (RegDeleteKey( hk, "" ))       {            printf("Key Delete Error\n");       } else       {            printf("Key Deleted Successfull\n");       }  }  else  {       printf("Key Not Found\nCreated Successfull\n");       strcpy(szBuf, "My Program");        // создаем параметр ключа       if (RegSetValueEx(   hk, // дескриптор ключа                       "File Name", // имя значения                       0, //зарезервировано (должно быть 0)                      REG\_SZ, // тип значения                       (LPBYTE) szBuf, // значение                       strlen(szBuf) + 1)) //длина значения       {            printf("Value 1 Set Error\n"); RegCloseKey(hk); return(0);       }       dwData = 1;       if (RegSetValueEx(   hk,                       "Application Run",                       0,                       REG\_DWORD,                       (LPBYTE) &dwData,                       sizeof(DWORD)))       {            printf("Value 2 Set Error\n"); RegCloseKey(hk); return(0);       }  }  RegCloseKey(hk);  return 0;  }    **Постановка задачи**: разработать оконное приложение в среде программирования Visual C в соответствии с заданным вариантом. Для хранения промежуточных результатов можно использовать средства WIN32 API для работы с памятью.    **Порядок выполнения работы**    1. Ознакомиться с постановкой задачи и исходными данными. В соответствии с номером по журналу определить вариант задачи.  2. Согласно рекомендациям, приведенным в исходных данных, сконструировать структуру программы.  3. Составить тексты программ и утвердить их у преподавателя.  4. Пункты 1 - 3 должны быть выполнены предварительно до проведения данной лабораторной работы. Утверждение преподавателем текстов программ является допуском к лабораторной работе.  5. Набрать текст программы.  6. Выполнить компиляцию программы.  7. Провести анализ и исправление обнаруженных синтаксических ошибок в тексте программы и повторить пункты 6. и 7. При устранении всех синтаксических ошибок перейти к выполнению пункта 8.  8. Получить решение и, в случае обнаружения логических ошибок, описать и устранить их. Продемонстрировать преподавателю окончательный вариант программы и ее работу.    **Содержание отчета**    1. Тема лабораторной работы.  2. Цель работы.  3. Индивидуальное задание.  4. Метод и алгоритм решения задачи.  5. Текст программы.  6. Результаты работы программы.  7. Выводы по работе.    **Индивидуальные задания**    15. Написать программу, которая отслеживает изменение даты на компьютере в сторону уменьшения. | |