Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики

и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе №5

на тему

«Реестр и журналы (Windows). Доступ к реестру Windows. Работа с журналами Windows. Другие вспомогательные средства управления.»

Выполнил:

студент гр. 153504

Сивый А.А.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цели работы 3](#_gjdgxs)

[2 Краткие теоретические сведения 4](#_30j0zll)

[3 Полученные результаты 5](#_1fob9te)

[Вывод 6](#_4q3p755gzys6)

[Приложение А (обязательное) Листинг кода 7](#_4q3p755gzys6)

**1 ЦЕЛИ РАБОТЫ**

1 Изучить основные понятия и принципы работы с реестром Windows.

2 Освоить работу с функциями Win 32 API для работы с реестром.

3 Изучить возможности работы с журналами Windows и научиться записывать в них информацию.

4 Разработать программный продукт согласно выбранный теме.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр — это иерархическая база данных, содержащая данные, критически важные для работы Windows, а также приложений и служб, работающих в Windows. Данные структурированы в формате дерева. Каждый узел в дереве называется ключом. Каждый ключ может содержать как подразделы, так и записи данных, называемые значениями. Иногда наличие ключа — это все данные, необходимые приложению; в других случаях приложение открывает ключ и использует значения, связанные с ключом. Ключ может иметь любое количество значений, а значения могут быть в любой форме.

Чтобы создать или открыть ключ, приложение всегда ссылается на ключ как на подраздел открытого ключа. Всегда открыты следующие стандартные ключи: **HKEY\_LOCAL\_MACHINE, HKEY\_CLASSES\_ROOT, HKEY\_USERS** и **HKEY\_CURRENT\_USER.** Для открытия ключа используется функция RegOpenKeyEx, а для его создания – RegCreateKeyEx. Дерево реестра может иметь глубину 512 уровней. С помощью одного вызова API реестра можно создавать до 32 уровней одновременно. Также после добавления ключа необходимо закрыть ключ с помощью функции RegCloseKey.

Приложение может использовать функцию RegSetValueEx для связывания значения и его данных с ключом. Чтобы удалить значение из ключа, приложение может использовать функцию RegDeleteValue. Чтобы удалить ключ, можно использовать функцию RegDeleteKey. Удаленный ключ не удаляется, пока не будет закрыт последний дескриптор. Подразделы и значения не могут быть созданы в удаленном ключе.

Чтобы получить данные из реестра, приложение обычно перечисляет подразделы раздела, пока не найдет конкретный ключ, а затем извлекает данные из значения или значений, связанных с ним. Приложение может вызывать функцию RegEnumKeyEx для перечисления подразделов заданного ключа.

Чтобы получить подробные данные о конкретном подразделе, приложение может вызвать функцию RegQueryInfoKey.

Приложение может использовать функцию RegEnumValue для перечисления значений для заданного ключа и функцию RegQueryValueEx для получения определенного значения для ключа.

**3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В результате выполнения лабораторной работы была создана программа, которая позволяет создавать и управлять реестровыми записями Windows, включая создание, изменение и удаление ключей и значений.

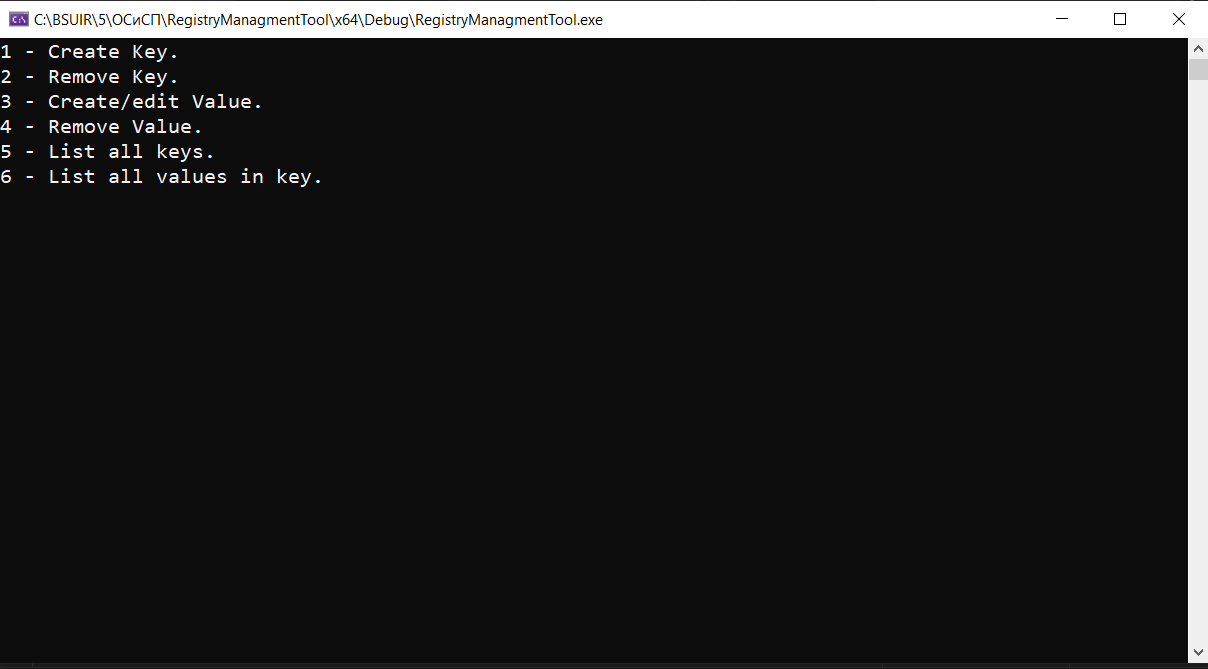


Рисунок 3.1 – Результат работы программы.

Программа позволяет создавать и удалять ключи, также создавать и добавлять значения. Название ключей и значений пользователь вводит с клавиатуры. Также есть возможность вывести на экран список всех ключей или список всех значений по ключу.

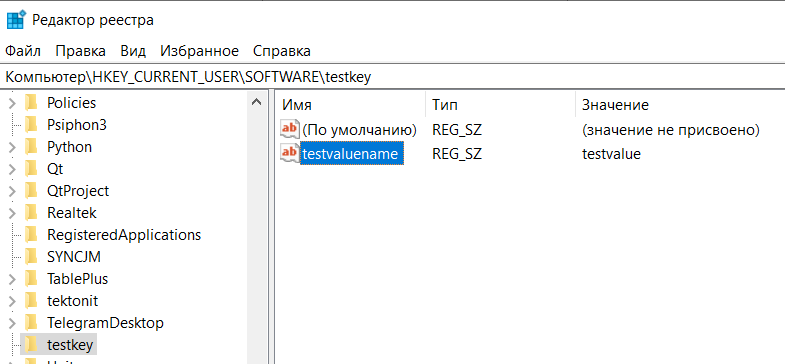


Рисунок 3.2 – Содержимое реестра после добавления ключа и значение.

На рисунке 3.2 продемонстрировано содержимое реестра после добавления ключа «testkey» и параметра «testvaluename» со значением «testvalue».

**ВЫВОД**

В результате выполнения лабораторной работы были изучены основные понятия и принципы работы с реестром Windows. Также был разработан программный продукт, который позволяет создавать, изменять и удалять ключи и значения в реестре Windows. Также есть возможность просмотра содержимого реестра и вывода всех значений по ключу.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг кода**

#include "main.h"

void AddRegistryKey() {

std::cout << "Enter key name: " << std::endl;

std::wstring keyName;

std::wcin >> keyName;

std::wstring subKey = L"SOFTWARE\\" + keyName;

HKEY hKey;

if (RegCreateKeyEx(HKEY\_CURRENT\_USER, subKey.c\_str(), 0, NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_ALL\_ACCESS, NULL, &hKey, NULL) == ERROR\_SUCCESS) {

std::cout << "Key successfully created.\n";

RegCloseKey(hKey);

}

else {

std::cout << "The registry key was not created.\n";

}

}

void RemoveRegistryKey()

{

std::cout << "Enter key name: ";

std::wstring keyName;

std::wcin >> keyName;

std::wstring subKey = L"SOFTWARE\\" + keyName;

// Открываем раздел реестра HKEY\_CURRENT\_USER

HKEY hKey;

if (RegOpenKeyExW(HKEY\_CURRENT\_USER, L"Software", 0, KEY\_ALL\_ACCESS, &hKey) != ERROR\_SUCCESS)

{

std::cerr << "Unable to access registry key." << std::endl;

return;

}

// Удаляем ключ из реестра

if (RegDeleteKeyW(hKey, keyName.c\_str()) != ERROR\_SUCCESS)

{

std::cerr << "Key is not deleted" << std::endl;

RegCloseKey(hKey);

return;

}

std::cout << "Key successfully deleted." << std::endl;

// Закрываем открытый реестровый ключ

RegCloseKey(hKey);

}

void AddValue()

{

HKEY hKey;

std::cout << "Enter key name" << std::endl;

std::wstring keyName;

std::wcin >> keyName;

std::wstring subKey = L"SOFTWARE\\" + keyName;

// Открытие указанного ключа

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, subKey.c\_str(), 0, KEY\_ALL\_ACCESS, &hKey) == ERROR\_SUCCESS) {

// Добавление параметра и его значения

std::cout << "Enter value name: " << std::endl;

std::wstring valueName;

std::wcin >> valueName;

std::cout << "Enter value: " << std::endl;

std::wstring value;

std::wcin >> value;

if (RegSetValueEx(hKey, valueName.c\_str(), 0, REG\_SZ, (BYTE\*)value.c\_str(), (wcslen(value.c\_str()) + 1) \* sizeof(wchar\_t)) == ERROR\_SUCCESS) {

std::cout << "Value set successfully." << "\n";

}

else {

std::cout << "Error creating value.\n";

}

RegCloseKey(hKey);

}

else {

std::cout << "Unable to access registry key.\n";

}

return;

}

void RemoveValue() {

HKEY hKey;

std::cout << "Enter key name: " << std::endl;

std::wstring keyName;

std::wcin >> keyName;

std::wstring subKey = L"SOFTWARE\\" + keyName;

// Открытие указанного ключа

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, subKey.c\_str(), 0, KEY\_ALL\_ACCESS, &hKey) == ERROR\_SUCCESS) {

std::cout << "Enter value name:" << std::endl;

std::wstring valueName;

std::wcin >> valueName;

// Удаление параметра

if (RegDeleteValue(hKey, valueName.c\_str()) == ERROR\_SUCCESS) {

std::cout << "Value deleted successfully." << "\n";

}

else {

std::cout << "Error deleting value.\n";

}

RegCloseKey(hKey);

}

else {

std::cout << "Unable to access registry key.\n";

}

}

void ListAllKeys() {

HKEY hKey;

DWORD dwIndex = 0;

TCHAR szKeyName[MAX\_PATH];

DWORD dwKeyNameSize = MAX\_PATH;

// Открытие указанного ключа

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, TEXT("Software"), 0, KEY\_READ, &hKey) == ERROR\_SUCCESS) {

std::cout << "List of all keys in Software:\n";

// Перебор всех подключей

while (RegEnumKeyEx(hKey, dwIndex, szKeyName, &dwKeyNameSize, NULL, NULL, NULL, NULL) == ERROR\_SUCCESS) {

std::wcout << szKeyName << "\n";

dwIndex++;

dwKeyNameSize = MAX\_PATH; // Сброс размера имени ключа

}

RegCloseKey(hKey);

}

else {

std::cout << "Unable to access registry key.\n";

}

}

void ListValueNamesAndValuesByKey() {

HKEY hKey;

DWORD dwIndex = 0;

TCHAR szValueName[MAX\_PATH];

DWORD dwValueNameSize = MAX\_PATH;

BYTE szValueData[MAX\_PATH];

DWORD dwValueDataSize = MAX\_PATH;

DWORD dwType = 0;

std::cout << "Enter key name" << std::endl;

std::wstring keyName;

std::wcin >> keyName;

std::wstring subKey = L"SOFTWARE\\" + keyName;

// Открытие указанного ключа

if (RegOpenKeyEx(HKEY\_CURRENT\_USER, subKey.c\_str(), 0, KEY\_READ, &hKey) == ERROR\_SUCCESS) {

std::cout << "List of all values in key:" << ":\n";

// Перебор всех параметров ключа

while (RegEnumValue(hKey, dwIndex, szValueName, &dwValueNameSize, NULL, &dwType, szValueData, &dwValueDataSize) == ERROR\_SUCCESS) {

std::wcout << szValueName << ": ";

switch (dwType) {

case REG\_SZ:

std::wcout << (wchar\_t\*)szValueData << "\n";

break;

case REG\_DWORD:

std::wcout << \*(DWORD\*)szValueData << "\n";

break;

default:

std::wcout << "Unsupported data type.\n";

}

dwIndex++;

dwValueNameSize = MAX\_PATH; // Сброс размера имени параметра

dwValueDataSize = MAX\_PATH; // Сброс размера данных параметра

}

RegCloseKey(hKey);

}

else {

std::cout << "Unable to access registry key.\n";

}

}

INT main() {

BOOL manage\_var = TRUE;

while (manage\_var) {

printf("1 - Create Key.\n");

printf("2 - Remove Key.\n");

printf("3 - Create/edit Value.\n");

printf("4 - Remove Value.\n");

printf("5 - List all keys.\n");

printf("6 - List all values in key.\n");

int choice;

if (scanf\_s("%d", &choice) != 1) {

printf("Wrong input. Try one more time.\n");

break;

}

switch (choice) {

case 1:

AddRegistryKey();

break;

case 2:

RemoveRegistryKey();

break;

case 3:

AddValue();

break;

case 4:

RemoveValue();

break;

case 5:

ListAllKeys();

break;

case 6:

ListValueNamesAndValuesByKey();

break;

default:

manage\_var = FALSE;

break;

}

}

getchar();

return 0;}