ГУО «Белорусский государственный университет  
информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4

по дисциплине

«Операционные системы и системное программирование»

«Работа с реестром»

Выполнил:

Андросов И.С.

группа 851005

Проверил:

Деменковец Д.В.

Минск, 2020

**Код программы**

**Source.cpp**

#undef UNICODE

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <windows.h>

#include <stdio.h>

#include <AclAPI.h>

#include <sddl.h>

#include <string>

using namespace std;

#define MAX\_LINE 50

#define CREATE\_KEY 0

#define WRITE\_REG\_DWORD 1

#define READ\_REG\_DWORD 2

#define WRITE\_REG\_STRING 3

#define READ\_REG\_STRING 4

#define FIND\_REG\_KEY 5

#define READ\_KEY\_FLAG 6

#define NOTIFY\_KEY\_CHANGED 7

#define EXIT 8

HANDLE hKeyChangedEvent;

HANDLE hKeyChangedThread;

BOOL FindKey(HKEY currentKey, LPCSTR keyName);

int ReadKeyFlags(HKEY currentKey);

LPCSTR ReadStringFromRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName);

void WriteStringInRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName, LPCSTR value);

void WriteDwordInRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName, DWORD value);

DWORD ReadDwordValueRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName);

void CreateRegistryKey(HKEY key, LPCSTR subkey);

void CreateThreadAndEvent();

DWORD WINAPI onKeyChanged(LPVOID lpParam);

void CreateRegistryKey(HKEY key, LPCSTR subkey)

{

DWORD dwDisposition;

HKEY hKey;

DWORD Ret;

Ret =

RegCreateKeyEx(key, subkey, 0,

NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_ALL\_ACCESS, NULL, &hKey, &dwDisposition);

if (Ret != ERROR\_SUCCESS)

{

printf("Ошибка создания ключа\n");

}

else

printf("Ключ создан\n");

RegCloseKey(hKey);

}

void WriteDwordInRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName, DWORD value)

{

DWORD disposition;

HKEY openedKey;

DWORD result;

result = RegOpenKeyEx(key, subkey, 0, KEY\_WRITE, &openedKey);

if (result != ERROR\_SUCCESS)

{

printf("Ключ не может быть открыт\n");

}

result = RegSetValueEx(openedKey, valueName, 0, REG\_DWORD, (BYTE\*)&value, sizeof(DWORD));

if (result != ERROR\_SUCCESS)

{

printf("Нельзя произвести запись\n");

}

RegCloseKey(openedKey);

}

DWORD ReadDwordValueRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName)

{

DWORD result;

HKEY openedKey;

result = RegOpenKeyEx(key, subkey, 0, KEY\_READ, &openedKey);

if (result != ERROR\_SUCCESS)

{

printf("Ключ не может быть открыт\n");

return -1;

}

DWORD buffer;

DWORD len = sizeof(buffer);

result = RegQueryValueEx(openedKey, valueName, NULL, NULL, (BYTE\*)&buffer, &len);

if (result != ERROR\_SUCCESS)

{

printf("Нельзя прочитать значение");

return -1;

}

RegCloseKey(openedKey);

return buffer;

}

void WriteStringInRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName, LPCSTR value)

{

DWORD disposition;

HKEY openedKey;

DWORD result;

result = RegOpenKeyEx(key, subkey, 0, KEY\_WRITE, &openedKey);

if (result != ERROR\_SUCCESS)

printf("Нельзя создать ключ");

result = RegSetValueEx(openedKey, valueName, 0, REG\_SZ, (BYTE\*)value, strlen(value));

if (result != ERROR\_SUCCESS)

printf("Нельзя произвести запись");

RegCloseKey(openedKey);

}

LPCSTR ReadStringFromRegistry(HKEY key, LPCSTR subkey, LPCSTR valueName)

{

DWORD result;

HKEY openedKey;

result = RegOpenKeyEx(key, subkey, 0, KEY\_READ, &openedKey);

if (result != ERROR\_SUCCESS)

{

printf("Ключ не может быть открыт");

return NULL;

}

char\* buffer = new char[MAX\_LINE];

DWORD len = MAX\_LINE;

result = RegQueryValueEx(openedKey, valueName, NULL, NULL, (BYTE\*)buffer, &len);

if (result != ERROR\_SUCCESS)

{

printf("Нельзя прочитать значение");

return NULL;

}

RegCloseKey(openedKey);

return buffer;;

}

BOOL FindKey(HKEY currentKey, LPCSTR keyName)

{

DWORD subkeysAmount;

DWORD maxSubkeyLen, currentSubkeyLen;

BOOL result;

RegQueryInfoKey(currentKey, NULL, 0, NULL, &subkeysAmount, &maxSubkeyLen,

NULL, NULL, NULL, NULL, NULL, NULL);

char\* bufferName = new char[maxSubkeyLen];

for (int i = 0; i < subkeysAmount; i++)

{

currentSubkeyLen = maxSubkeyLen;

result = RegEnumKeyEx(currentKey, i, bufferName, &currentSubkeyLen, NULL, NULL, NULL, NULL);

if (result == ERROR\_SUCCESS)

{

if (!strcmp(bufferName, keyName))

{

return TRUE;

}

HKEY innerKey;

result = RegOpenKey(currentKey, bufferName, &innerKey);

if (result == ERROR\_SUCCESS)

{

result = FindKey(innerKey, keyName);

if (result)

{

RegCloseKey(innerKey);

return result;

}

}

RegCloseKey(innerKey);

}

}

return FALSE;

}

const char\* ConvertAceStringToString(char\* source)

{

string strSource(source);

string\* strKeyAccess = new string("");

int strLen = strSource.length();

int i = 0;

int semicolonCounter = 0;

while (i < strLen)

{

if (source[i] == ')')

{

semicolonCounter = 0;

}

if (source[i] == ';')

{

semicolonCounter++;

}

if (semicolonCounter == 2)

{

int start = i + 1;

do

{

i++;

} while (source[i] != ';');

strKeyAccess->append(strSource.substr(start, i - start));

semicolonCounter++;

}

if (semicolonCounter == 5)

{

int start = i + 1;

while (source[i] != ')')

{

i++;

}

strKeyAccess->append(" <-" + strSource.substr(start, i - start) + "\n");

semicolonCounter = 0;

}

i++;

}

return strKeyAccess->c\_str();

}

int readKeyFlags(HKEY currentKey)

{

int isSuccess = 1;

DWORD securityDescriptorSize;

DWORD subkeysNumber;

RegQueryInfoKey(currentKey, NULL, 0, NULL, &subkeysNumber,

NULL, NULL, NULL, NULL, NULL, &securityDescriptorSize, NULL);

char\* buffer = new char[securityDescriptorSize];

DWORD result;

result = RegGetKeySecurity(currentKey, DACL\_SECURITY\_INFORMATION, buffer, &securityDescriptorSize);

if (result != ERROR\_SUCCESS)

{

printf("Нельзя прочитать флаги\n");

isSuccess = 0;

}

else

{

SECURITY\_DESCRIPTOR\* security = reinterpret\_cast<SECURITY\_DESCRIPTOR\*>(buffer);

LPSTR strDacl;

ConvertSecurityDescriptorToStringSecurityDescriptor(security, SDDL\_REVISION\_1, DACL\_SECURITY\_INFORMATION, &strDacl, NULL);

printf("%s\n", strDacl);

printf("%s\n", ConvertAceStringToString(strDacl));

}

return isSuccess;

}

void notifyKeyChanged(HKEY currentKey)

{

hKeyChangedEvent = CreateEvent(NULL, TRUE, FALSE, "Ключ изменён");

DWORD threadId;

hKeyChangedThread = CreateThread(NULL, 0, onKeyChanged, NULL, 0, &threadId);

RegNotifyChangeKeyValue(currentKey, TRUE, REG\_NOTIFY\_CHANGE\_LAST\_SET, hKeyChangedEvent, TRUE);

}

void CloseEvents()

{

CloseHandle(hKeyChangedEvent);

}

DWORD WINAPI onKeyChanged(LPVOID lpParam)

{

WaitForSingleObject(hKeyChangedEvent, INFINITE);

printf("Ключ изменён\n");

return 0;

}

int main()

{

setlocale(LC\_ALL, "Russian");

string rules = "0 - Cоздать ключ\n";

rules += "1 - Записать число\n";

rules += "2 - Прочитать число\n";

rules += "3 - Записать строку\n";

rules += "4 - Прочитать строку\n";

rules += "5 - Поиск ключа\n";

rules += "6 - Прочитать флаги\n";

rules += "7 - Отслеживать иземения реестра\n";

rules += "8 - Выход\n";

printf("%s\n", rules.c\_str());

int isContinue = 1;

char buffer[1024];

char secondBuffer[1024];

DWORD dwValue;

DWORD dwResult;

LPCSTR strResult;

HKEY currentKey;

DWORD disposition;

char subkey[1024];

while (isContinue)

{

printf("Введите команду: ");

int command;

scanf("%d", &command);

switch (command)

{

case CREATE\_KEY:

printf("Введите имя ключа: ");

scanf("%s", buffer);

CreateRegistryKey(HKEY\_CURRENT\_USER, buffer);

break;

case WRITE\_REG\_DWORD:

printf("Введите имя подключа: ");

scanf("%s", subkey);

printf("Введите имя записи: ");

scanf("%s", buffer);

printf("Значение записи: ");

scanf("%d", &dwValue);

WriteDwordInRegistry(HKEY\_CURRENT\_USER, subkey, buffer, dwValue);

break;

case READ\_REG\_DWORD:

printf("Введите имя подключа: ");

scanf("%s", subkey);

printf("Введите имя записи: ");

scanf("%s", buffer);

dwResult = ReadDwordValueRegistry(HKEY\_CURRENT\_USER, subkey, buffer);

if (dwResult != -1)

printf("Значение записи: %d\n", dwResult);

break;

case WRITE\_REG\_STRING:

printf("Введите имя подключа: ");

scanf("%s", subkey);

printf("Введите имя записи: ");

scanf("%s", buffer);

printf("Значение записи: ");

scanf("%s", secondBuffer);

WriteStringInRegistry(HKEY\_CURRENT\_USER, subkey, buffer, secondBuffer);

break;

case READ\_REG\_STRING:

printf("Введите имя подключа: ");

scanf("%s", subkey);

printf("Введите имя записи: ");

scanf("%s", buffer);

strResult = ReadStringFromRegistry(HKEY\_CURRENT\_USER, subkey, buffer);

if (strResult != NULL)

printf("Значение записи: %s\n", strResult);

break;

case FIND\_REG\_KEY:

printf("Введите имя ключа: ");

scanf("%s", buffer);

if (FindKey(HKEY\_CURRENT\_USER, buffer))

printf("Ключ найден\n");

else

printf("Ключ не был найден\n");

break;

case READ\_KEY\_FLAG:

{

HKEY currentKey;

printf("Введите имя ключа: ");

scanf("%s", buffer);

dwResult = RegOpenKey(HKEY\_CURRENT\_USER, buffer, &currentKey);

if (dwResult != ERROR\_SUCCESS)

printf("Ключ не может быть открыт\n");

else

{

readKeyFlags(currentKey);

RegCloseKey(currentKey);

}

}

break;

case NOTIFY\_KEY\_CHANGED:

{

printf("Введите имя ключа:");

scanf("%s", buffer);

HKEY currentKey;

dwResult = RegOpenKey(HKEY\_CURRENT\_USER, buffer, &currentKey);

if (dwResult != ERROR\_SUCCESS)

printf("Ключ не может быть открыт\n");

else

notifyKeyChanged(currentKey);

}

break;

case EXIT:

isContinue = 0;

CloseEvents();

break;

default:

break;

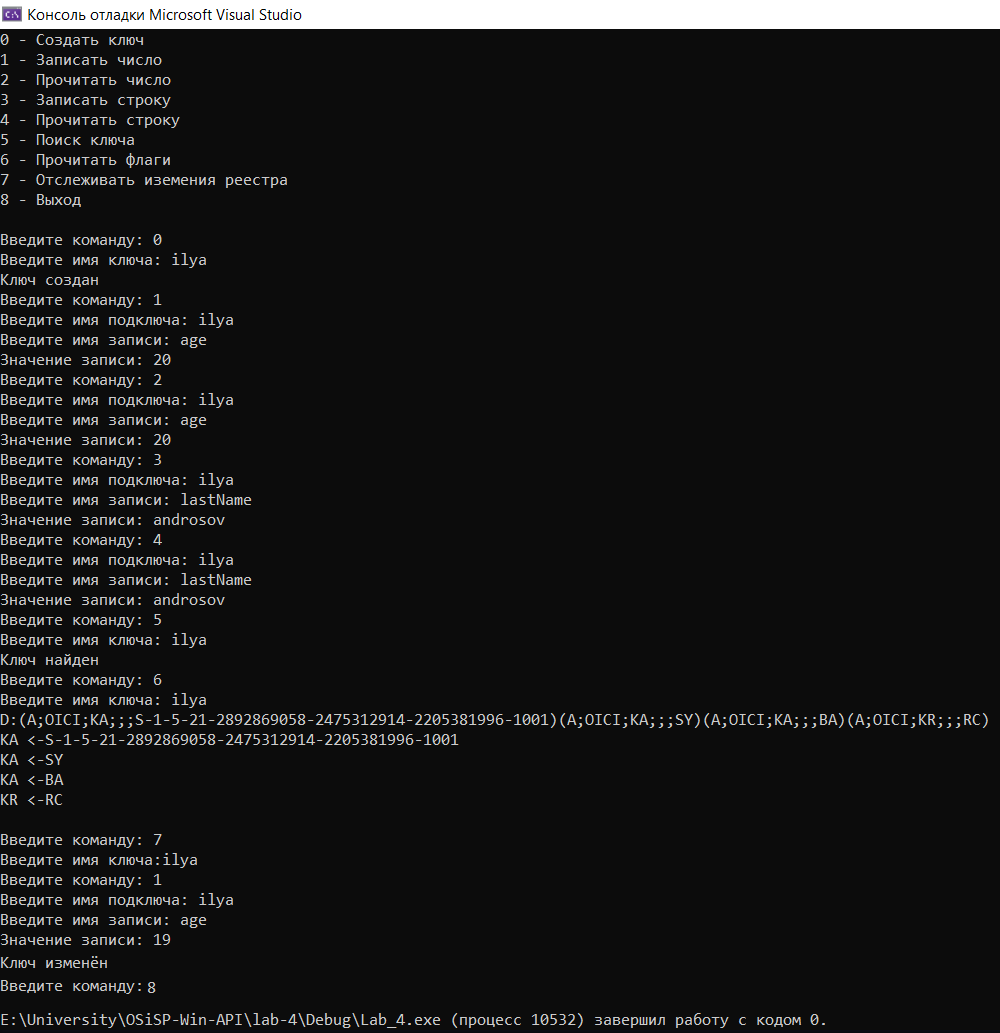
}

}

return 0;

}

**Скриншоты выполнения программы**

****

**Вывод:**

Была выполнена работа с реестром, а именно: создание и открытие ключа, закрытие ключа, добавление данных к ключу, поиск ключа, получение значений флагов, составляющих маску ключа, а также отслеживание изменений реестра. Данные ключи всегда открыты: **HKEY\_CLASSES\_ROOT, HKEY\_CURRENT\_CONFIG, HKEY\_CURRENT\_USER, HKEY\_LOCAL\_MACHINE, HKEY\_USERS**. Для создания ключа, который сохраняется при выключении системы необходимо было использовать флаг **REG\_OPTION\_NON\_VOLATILE.**