УО «Белорусский государственный университет информатики и радиоэлектроники»
Кафедра ПОИТ

Отчет по лабораторной работе №4

по предмету

Операционные системы и системное программирование

Выполнил
Каширский А.Е.

Проверил
Деменковец Д. В.

Группа:
851005

Минск 2020

Код программы

```
#undef UNICODE

#define _CRT_SECURE_NO_WARNINGS

#include <windows.h>

#include <stdio.h>

#include <AclAPI.h>

#include <sddl.h>

#include <string>


using namespace std;


#define MAX_STRING_LENGTH 100


#define HELP_MSG "1 - Find key\n2 - Read key string value\n3 - Write string
value\n4 - Show subkeys\n5 - Read key flags\n6 - Create new key\n7 - Close
programm\n"

#define INPUT_KEY_MSG "Enter key name: "

#define INPUT_VALUE_NAME_MSG "Enter value name: "

#define INPUT_VALUE_MSG "Enter new value: "

#define ERROR_MSG_CANNOT_OPEN_KEY "Can not open key!"

#define ERROR_MSG_CANNOT_READ_VALUE "Can not read value!"

#define ERROR_MSG_CANNOT_WRITE_REG "Can not write registry!"


#define FIND_REG_KEY 1

#define READ_REG_STRING 2

#define WRITE_REG_STRING 3

#define SHOW_SUBKEYS 4

#define READ_KEY_FLAG 5

#define CREATE_REG_KEY 6

#define CLOSE_PROGRAMM 7


int findSubKey(HKEY, LPCSTR);

LPCSTR readStringValue(HKEY, LPCSTR, LPCSTR);

DWORD writeStringValue(HKEY, LPCSTR, LPCSTR, LPCSTR);

void showSubKeys(HKEY);
```

```
void showKeyFlags(HKEY);
int createKeyByName(HKEY, LPCSTR);

int findSubKey(HKEY key, LPCSTR subKeyName)
{
	DWORD subKeysAmount, subKeyLen = 4096, currentSubKeyLen, result;
	RegQueryInfoKey(key, NULL, 0, NULL, &subKeysAmount, &subKeyLen,
		NULL, NULL, NULL, NULL, NULL, NULL);
	char* bufferName = new char[subKeyLen];
	for (int i = 0; i < subKeysAmount; i++)
	{
		currentSubKeyLen = 4096;
		result = RegEnumKeyEx(key, i, bufferName, &currentSubKeyLen, NULL,
NULL, NULL, NULL);
		if (result == ERROR_SUCCESS)
		{
			if (!strcmp(bufferName, subKeyName))
			{
				return 1;
			}
			HKEY subKey;
			result = RegOpenKey(key, bufferName, &subKey);
			if (result == ERROR_SUCCESS)
			{
				result = findSubKey(subKey, subKeyName);
				if (result)
				{
					RegCloseKey(subKey);
					return result;
				}
			}
			RegCloseKey(subKey);
		}
	}
	return 0;
}
```

```
LPCSTR readStringValue(HKEY key, LPCSTR subKey, LPCSTR valueName)
{
	HKEY openedKey;
	char* stringValue = new char[MAX_STRING_LENGTH];

	if (ERROR_SUCCESS != RegOpenKeyEx(key, subKey, NULL, KEY_READ,
&openedKey))
	{
		printf(ERROR_MSG_CANNOT_OPEN_KEY);
		return NULL;
	}

	DWORD length = MAX_STRING_LENGTH;

	if (ERROR_SUCCESS != RegQueryValueEx(openedKey, valueName, NULL, NULL,
(BYTE*)stringValue, &length))
	{
		printf(ERROR_MSG_CANNOT_READ_VALUE);
		return NULL;
	}

	RegCloseKey(openedKey);

	return stringValue;
}

DWORD writeStringValue(HKEY key, LPCSTR subKey, LPCSTR valueName, LPCSTR
value)
{
	HKEY openedKey;

	if (ERROR_SUCCESS != RegOpenKeyEx(key, subKey, NULL, KEY_ALL_ACCESS,
&openedKey))
	{
		printf(ERROR_MSG_CANNOT_OPEN_KEY);

		return -1;
```

```
	}


	if (ERROR_SUCCESS != RegSetValueEx(openedKey, valueName, 0, REG_SZ,
(BYTE*)value, strlen(value)))
	{
		printf(ERROR_MSG_CANNOT_WRITE_REG);


		return -1;
	}


	RegCloseKey(openedKey);
	return 0;
}


void showSubKeys(HKEY key)
{
	DWORD subKeysAmount;
	DWORD maxSubKeyLength;
	char* subKeyName = new char[MAX_STRING_LENGTH];
	RegQueryInfoKey(key, NULL, NULL, NULL, &subKeysAmount, &maxSubKeyLength,
NULL, NULL, NULL, NULL, NULL, NULL);
	maxSubKeyLength = MAX_STRING_LENGTH; // hot fix for max length because
RegQueryInfo doesn't work correctly((((((


	if (subKeysAmount > 0)
	{
		printf("Subkeys: \n");
	}
	else
	{
		printf("There are no subkeys for this key!\n");
	}


	for (int i = 0; i < subKeysAmount; i++)
	{
		maxSubKeyLength = MAX_STRING_LENGTH;
		if (ERROR_SUCCESS != RegEnumKeyEx(key, i, subKeyName,
&maxSubKeyLength, NULL, NULL, NULL, NULL))
```

```
            {
                  printf("Can not get subkey\n");
            }
            else
            {
                  printf("%s\n", subKeyName);
            }
      }
}


int getSecurityFlags(HKEY key, DWORD securityDescriptorSize, long
securityCode, char* msg)
{
      bool isOK = true;
      char* securityInformation = new char[securityDescriptorSize];
      if (ERROR_SUCCESS != RegGetKeySecurity(key, securityCode,
securityInformation, &securityDescriptorSize))
      {
            printf("Can not get security\n");
            isOK = false;
      }
      else
      {
            SECURITY_DESCRIPTOR* security =
reinterpret_cast<SECURITY_DESCRIPTOR*>(securityInformation);
            LPSTR strSecurity;
            ConvertSecurityDescriptorToStringSecurityDescriptor(security,
SDDL_REVISION_1, securityCode, &strSecurity, NULL);
            printf(msg);
            printf("%s\n", strSecurity);
      }

      return isOK;
}

int readKeyFlags(HKEY key)
{
      int isSuccess = 1;
```

```
	DWORD securityDescriptorSize;
	RegQueryInfoKey(key, NULL, 0, NULL, NULL,
		NULL, NULL, NULL, NULL, NULL, &securityDescriptorSize, NULL);
	char msgOwner[] = "Owner security flags\n";
	char msgGroup[] = "Group security flags\n";
	char msgDACL[] = "DACL security flags\n";
	char msgSACL[] = "SACL security flags\n";
	getSecurityFlags(key, securityDescriptorSize,
OWNER_SECURITY_INFORMATION, msgOwner);
	getSecurityFlags(key, securityDescriptorSize,
GROUP_SECURITY_INFORMATION, msgGroup);
	getSecurityFlags(key, securityDescriptorSize, DACL_SECURITY_INFORMATION,
msgDACL);
	getSecurityFlags(key, securityDescriptorSize, SACL_SECURITY_INFORMATION,
msgSACL);

	return isSuccess;
}

int createKeyByName(HKEY key, LPCSTR keyName)
{
	DWORD disposition;
	HKEY openedKey;
	if (ERROR_SUCCESS != RegCreateKeyEx(HKEY_CURRENT_USER, keyName, NULL,
REG_OPTION_NON_VOLATILE, NULL, KEY_ALL_ACCESS, NULL, &openedKey,
&disposition))
	{
		printf("Can not create key\n");
		return -1;
	}
	else
	{
		if (REG_OPENED_EXISTING_KEY == disposition)
		{
			printf("Such key already exists!\n");
		}
		else
		{
```

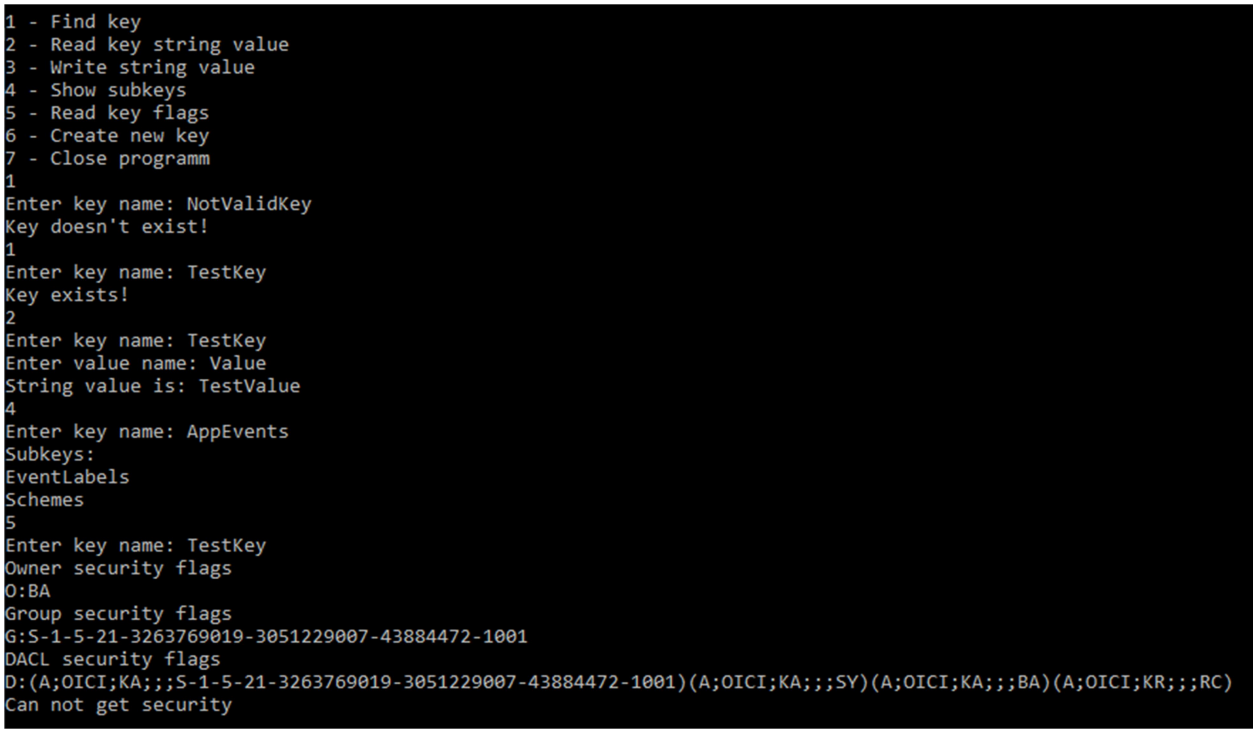
```
                printf("Key sucessfully created\n");
            }
        }
        RegCloseKey(openedKey);
}

int main()
{
        char keyName[MAX_STRING_LENGTH];
        char valueName[MAX_STRING_LENGTH];
        char newValue[MAX_STRING_LENGTH];
        int command;
        bool isNotClosing = true;
        printf(HELP_MSG);

        while (isNotClosing)
        {
            scanf("%d", &command);

            switch (command)
            {
                case FIND_REG_KEY:
                {
                    printf(INPUT_KEY_MSG);
                    scanf("%s", keyName);
                    if (findSubKey(HKEY_CURRENT_USER, keyName))
                    {
                        printf("Key exists!\n");
                    }
                    else
                    {
                        printf("Key doesn't exist!\n");
                    }
                }
                break;
```

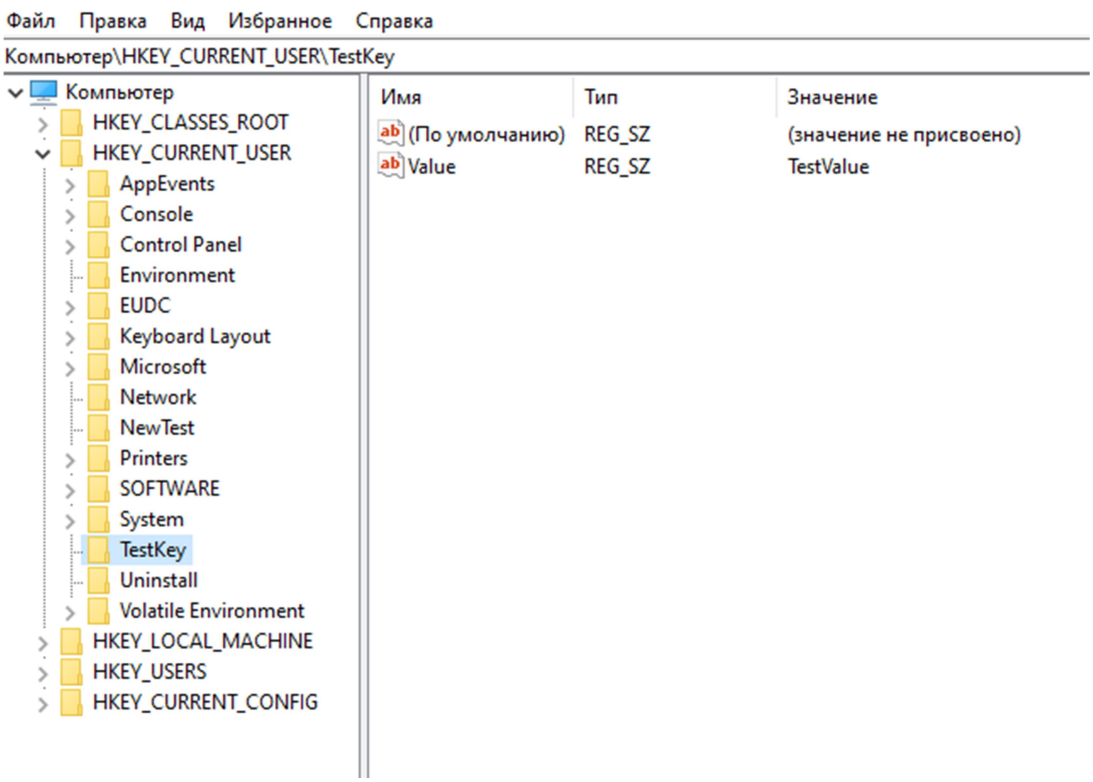
```
                case READ_REG_STRING:
                {
                        printf(INPUT_KEY_MSG);
                        scanf("%s", keyName);
                        printf(INPUT_VALUE_NAME_MSG);
                        scanf("%s", valueName);
                        LPCSTR strResult = readStringValue(HKEY_CURRENT_USER,
keyName, valueName);
                        if (strResult != NULL)
                        {
                              printf("String value is: %s\n", strResult);
                        }
                }
                break;

                case WRITE_REG_STRING:
                {
                        printf(INPUT_KEY_MSG);
                        scanf("%s", keyName);
                        printf(INPUT_VALUE_NAME_MSG);
                        scanf("%s", valueName);
                        printf(INPUT_VALUE_MSG);
                        scanf("%s", newValue);
                        writeStringValue(HKEY_CURRENT_USER, keyName,
valueName, newValue);
                }
                break;

                case SHOW_SUBKEYS:
                {
                        HKEY key;
                        printf(INPUT_KEY_MSG);
                        scanf("%s", keyName);

                        if (ERROR_SUCCESS != RegOpenKey(HKEY_CURRENT_USER,
keyName, &key))
                        {
```

```
                        printf(ERROR_MSG_CANNOT_OPEN_KEY);
                }
                else
                {
                        showSubKeys(key);

                        RegCloseKey(key);
                }
        }
        break;

        case READ_KEY_FLAG:
        {
                HKEY key;
                printf(INPUT_KEY_MSG);
                scanf("%s", keyName);

                if (ERROR_SUCCESS != RegOpenKey(HKEY_CURRENT_USER,
keyName, &key))
                {
                        printf(ERROR_MSG_CANNOT_OPEN_KEY);
                }
                else
                {
                        readKeyFlags(key);
                        RegCloseKey(key);
                }
        }
        break;

        case CREATE_REG_KEY:
        {
                printf(INPUT_KEY_MSG);
                scanf("%s", keyName);
                createKeyByName(HKEY_CURRENT_USER, keyName);
        }
```

```
                    break;


                    case CLOSE_PROGRAMM:
                    {
                          isNotClosing = false;
                    }
                    break;


             }
       }


       return 0;
}
```

Скриншоты работы программы




Файл Правка Вид Избранное Справка
Компьютер\HKEY_CURRENT_USER\TestKey
Компьютер
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
AppEvents
Console
Control Panel
Environment
EUDC
Keyboard Layout
Microsoft
Network
NewTest
Printers
SOFTWARE
System
TestKey
Uninstall
Volatile Environment
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
Имя
Тип
Значение
(По умолчанию)
REG_SZ
(значение не присвоено)
Value
REG_SZ
TestValue