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

Прескурел Я.Ю.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Результаты выполнения лабораторной работы 6](#_Toc146631500)

[Выводы 8](#_Toc146631501)

[Список использованных источников 9](#_Toc146631502)

[Приложение А (обязательное) Листинг кода 10](#_Toc146631503)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является создание утилиты для установки и удаления программ с использование информации из реестра Windows.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) представляет собой набор функций и интерфейсов, предоставляемых операционной системой Windows для разработки приложений. Этот мощный набор инструментов обеспечивает доступ к различным функциональным возможностям Windows, включая создание и управление окнами, обработку сообщений, работу с файлами и реестром, а также многие другие операции. Win32 API играет ключевую роль в разработке приложений для Windows и обеспечивает высокую степень контроля над поведением приложений.

Реестр Windows и журналы событий играют важную роль в управлении и мониторинге операционной системы Microsoft Windows. Реестр – это централизованное хранилище системных настроек, параметров и данных о приложениях. Он используется для хранения информации о конфигурации операционной системы, установленных приложениях, драйверах и других системных ресурсах. Реестр предоставляет доступ к этим данным и позволяет как системным администраторам, так и приложениям управлять системными параметрами.

Журналы событий Windows служат для регистрации и мониторинга событий и ошибок в операционной системе. Они предоставляют информацию о работе системы, приложениях и службах, а также могут содержать данные о безопасности и аудите. Журналы событий помогают администраторам и разработчикам выявлять проблемы, мониторить производительность и обнаруживать события, связанные с безопасностью.

Для управления реестром Windows и анализа журналов событий доступны различные инструменты и API. Они позволяют выполнять операции чтения, записи и анализа данных в реестре, а также мониторить и фильтровать события в журналах. Для обеспечения безопасности и надежности системы важно правильно использовать эти средства управления, чтобы предотвратить несанкционированный доступ и обнаруживать проблемы в работе операционной системы.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

1. Работа с реестром Windows: при написании кода использовались функции из Windows API (Application Programming Interface) для работы с реестром, такие как RegOpenKeyEx, RegCreateKey, RegSetValueEx, RegQueryValueEx, и RegCloseKey. Эти функции позволяют создавать, читать и записывать данные в реестр.

2. Процессы и потоки: при написании кода использовались функции для запуска инсталлятора и деинсталлятора программ с использованием ShellExecuteEx и WaitForSingleObject. Это связано с управлением процессами и ожиданием их завершения.

3. Ввод и вывод через командную строку: при написании кода были прописаны взаимодействия с пользователем, предлагая меню с опциями и принимая выбор пользователя.

4. Обработка ошибок и проверка результата операций: в коде используются условные операторы для проверки результатов операций, такие как открытие или создание ключей в реестре, выполнение инсталлятора или деинсталлятора.

5. Многозадачность: при написании кода использовались функции ожидания завершения процесса, что позволяет управлять многозадачным выполнением операций.

6. Работа со строками: при написании кода были использованы операции с C++ строками, включая ввод, вывод и манипуляции.

7. Интерфейс командной строки: программа предоставляет текстовый интерфейс, который позволяет пользователю выбирать опции и взаимодействовать с программой через команды.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было реализовано простое консольное приложение, представляющее собой утилиту для установки и удаления программ с использованием информации из реестра Windows. При использовании утилиты пользователь может взаимодействовать с меню, результат работы с программой представлены на рисунках 3.1 – 3.4.

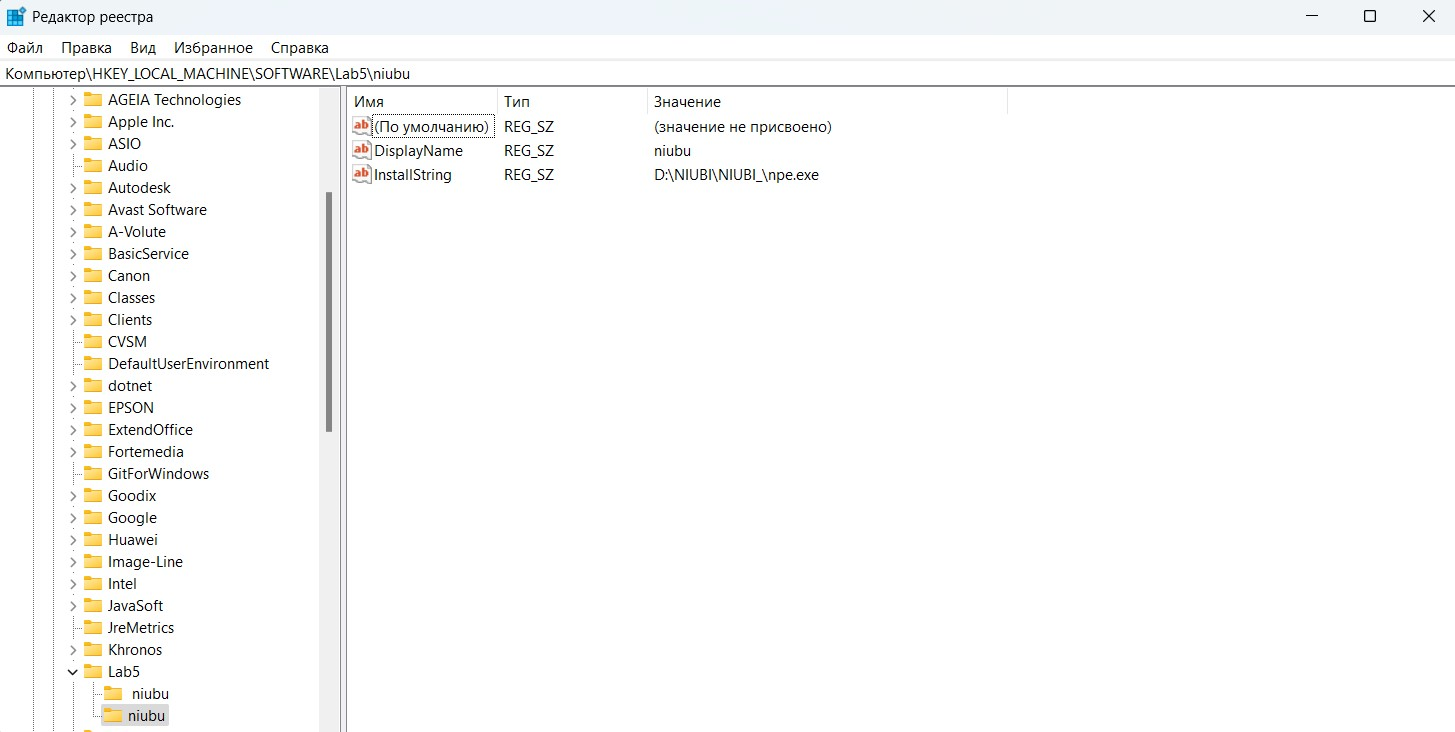


Рисунок 3.1 – Результат регистрации новой программы и установки пути до инсталлятора

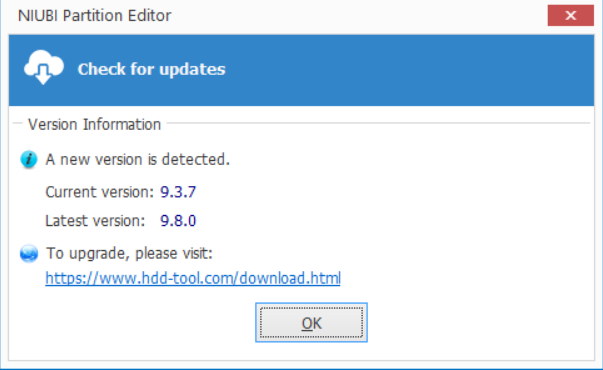


Рисунок 3.2 – Результат работы инсталлятора программы

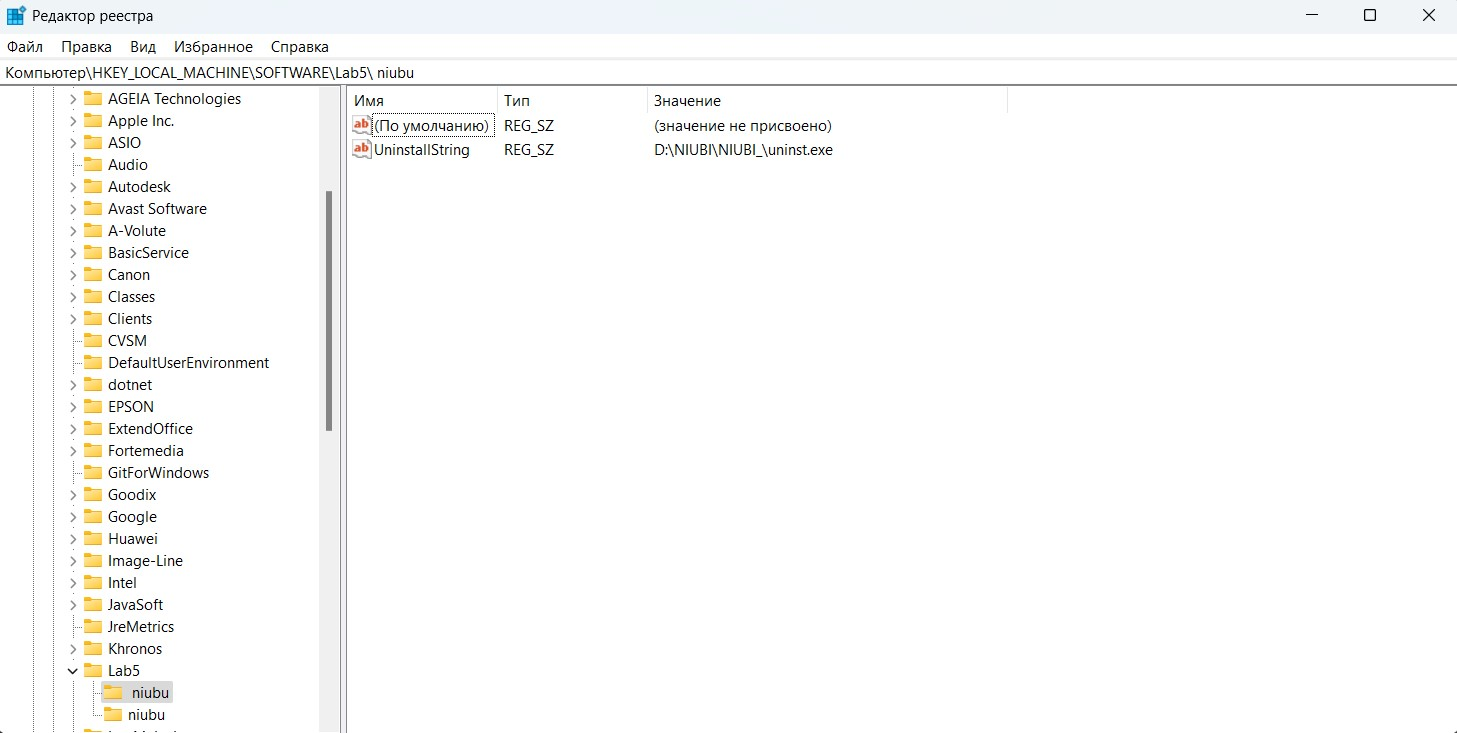


Рисунок 3.3 – Результат добавления пути к деинсталлятору

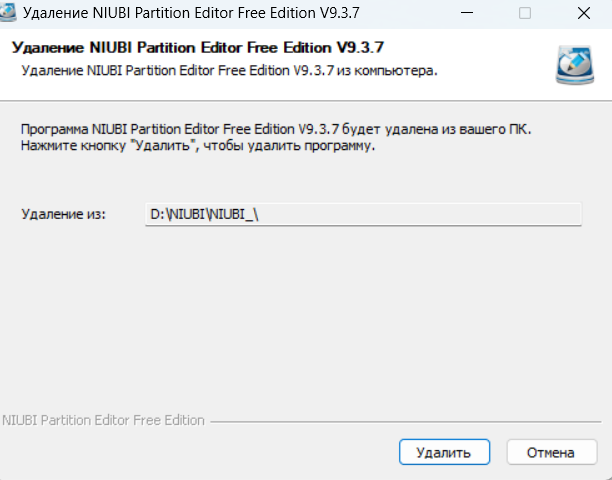


Рисунок 3.4 – Результат работы деинсталлятора программы

При последующих взаимодействиях с утилитой, пользователь без лишних взаимодействий с программой может использовать пути к инсталлятору и деинсталлятору для установки и удаления программы соответственно, если они будут располагаться на тех же путях в системе.

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы было создано простое консольное приложение, являющееся утилитой для установки и удаления программ с использованием информации из реестра.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.
2. [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/debug/system-error-codes--0-499 – Дата доступа 21.10.2023
3. [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecutea – Дата доступа: 23.10.2023

**ПРИЛОЖЕНИЕ А**

## (обязательное)

## Листинг кода

**Lab5.cpp**

#include <Windows.h>

#include <iostream>

#include <string>

using namespace std;

bool SetProgramRegistry(const string& programName, const string& installerPath)

{

HKEY hParentKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Lab5", 0, KEY\_WRITE, &hParentKey) == ERROR\_SUCCESS)

{

HKEY hAppKey;

if (RegCreateKey(hParentKey, programName.c\_str(), &hAppKey) == ERROR\_SUCCESS)

{

RegSetValueEx(hAppKey, "DisplayName", 0, REG\_SZ, (const BYTE\*)programName.c\_str(), programName.size());

RegSetValueEx(hAppKey, "InstallString", 0, REG\_SZ, (const BYTE\*)installerPath.c\_str(), installerPath.size());

RegCloseKey(hAppKey);

RegCloseKey(hParentKey);

return true;

}

}

return false;

}

bool RunInstaller(const string& installerPath)

{

SHELLEXECUTEINFO ShExecInfo = { 0 };

ShExecInfo.cbSize = sizeof(SHELLEXECUTEINFO);

ShExecInfo.fMask = SEE\_MASK\_NOCLOSEPROCESS;

ShExecInfo.lpFile = installerPath.c\_str();

ShExecInfo.nShow = SW\_SHOWNORMAL;

if (ShellExecuteEx(&ShExecInfo) == TRUE)

{

WaitForSingleObject(ShExecInfo.hProcess, INFINITE);

CloseHandle(ShExecInfo.hProcess);

return true;

}

return false;

}

bool SetUninstallRegistry(const string& programName, const string& uninstallerPath)

{

HKEY hParentKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Lab5", 0, KEY\_WRITE, &hParentKey) == ERROR\_SUCCESS)

{

HKEY hAppKey;

if (RegCreateKey(hParentKey, programName.c\_str(), &hAppKey) == ERROR\_SUCCESS)

{

RegSetValueEx(hAppKey, "UninstallString", 0, REG\_SZ, (const BYTE\*)uninstallerPath.c\_str(), uninstallerPath.size());

RegCloseKey(hAppKey);

RegCloseKey(hParentKey);

return true;

}

}

return false;

}

bool RunUninstaller(const string& programName)

{

HKEY hParentKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Lab5", 0, KEY\_READ, &hParentKey) == ERROR\_SUCCESS)

{

HKEY hAppKey;

if (RegOpenKeyEx(hParentKey, programName.c\_str(), 0, KEY\_READ, &hAppKey) == ERROR\_SUCCESS)

{

DWORD dataValueSize = 1024;

char uninstallerPath[1024];

if (RegQueryValueEx(hAppKey, "UninstallString", 0, NULL, (LPBYTE)uninstallerPath, &dataValueSize) == ERROR\_SUCCESS)

{

SHELLEXECUTEINFO ShExecInfo = { 0 };

ShExecInfo.cbSize = sizeof(SHELLEXECUTEINFO);

ShExecInfo.fMask = SEE\_MASK\_NOCLOSEPROCESS;

ShExecInfo.lpFile = uninstallerPath;

ShExecInfo.nShow = SW\_SHOWNORMAL;

if (ShellExecuteEx(&ShExecInfo) == TRUE)

{

WaitForSingleObject(ShExecInfo.hProcess, INFINITE);

CloseHandle(ShExecInfo.hProcess);

return true;

}

else

{

cout << "Failed to run the uninstaller." << endl;

return false;

}

}

RegCloseKey(hAppKey);

}

RegCloseKey(hParentKey);

}

return false;

}

void ListInstalledPrograms()

{

HKEY hParentKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Lab5", 0, KEY\_READ, &hParentKey) == ERROR\_SUCCESS)

{

DWORD subkeyIndex = 0;

char subkeyName[255];

while (RegEnumKey(hParentKey, subkeyIndex, subkeyName, sizeof(subkeyName)) == ERROR\_SUCCESS)

{

HKEY hAppKey;

if (RegOpenKeyEx(hParentKey, subkeyName, 0, KEY\_READ, &hAppKey) == ERROR\_SUCCESS)

{

cout << "Program: " << subkeyName << endl;

RegCloseKey(hAppKey);

}

subkeyIndex++;

}

RegCloseKey(hParentKey);

}

}

int main()

{

while (true)

{

int choice;

string programName;

string installerPath;

string uninstallerPath;

cout << "1. List installed programs\n2. Register new program and set InstallerPath\n3. Set UninstallerPath\n4. Install program\n5. Uninstall program\n6. Exit\nEnter your choice: ";

cin >> choice;

if (choice == 1)

{

ListInstalledPrograms();

}

else if (choice == 2)

{

cout << "Enter the program name: ";

cin.ignore();

getline(cin, programName);

cout << "Enter the installer path: ";

getline(cin, installerPath);

if (SetProgramRegistry(programName, installerPath))

{

cout << "Program information added to the registry." << endl;

}

else

{

cout << "Failed to add program information to the registry." << endl;

}

}

else if (choice == 3)

{

cout << "Enter the program name to set UninstallString:";

cin.ignore();

getline(cin, programName);

cout << "Enter the uninstaller path: ";

getline(cin, uninstallerPath);

if (SetUninstallRegistry(programName, uninstallerPath))

{

cout << "Uninstall information added to the registry." << endl;

}

else

{

cout << "Failed to add uninstall information to the registry." << endl;

}

}

else if (choice == 4)

{

cout << "Enter the program name to install: ";

cin.ignore();

getline(cin, programName);

HKEY hInstallKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Lab5", 0, KEY\_READ, &hInstallKey) == ERROR\_SUCCESS)

{

HKEY hAppKey;

if (RegOpenKeyEx(hInstallKey, programName.c\_str(), 0, KEY\_READ, &hAppKey) == ERROR\_SUCCESS)

{

DWORD valueSize = 1024;

char installPath[1024];

if (RegQueryValueEx(hAppKey, "InstallString", 0, NULL, (LPBYTE)installPath, &valueSize) == ERROR\_SUCCESS)

{

if (RunInstaller(installPath))

{

cout << "Program installed successfully!" << endl;

}

else

{

cout << "Failed to run the installer." << endl;

}

}

RegCloseKey(hAppKey);

}

RegCloseKey(hInstallKey);

}

}

else if (choice == 5)

{

cout << "Enter the program name to uninstall: ";

cin.ignore();

getline(cin, programName);

HKEY hUninstallKey;

if (RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, "SOFTWARE\\Lab5", 0, KEY\_READ, &hUninstallKey) == ERROR\_SUCCESS)

{

HKEY hAppKey;

if (RegOpenKeyEx(hUninstallKey, programName.c\_str(), 0, KEY\_READ, &hAppKey) == ERROR\_SUCCESS)

{

DWORD valueSize = 1024;

char uninstallPath[1024];

if (RegQueryValueEx(hAppKey, "UninstallString", 0, NULL, (LPBYTE)uninstallPath, &valueSize) == ERROR\_SUCCESS)

{

if (RunUninstaller(programName))

{

cout << "Program uninstalled successfully!" << endl;

}

else

{

cout << "Failed to run the uninstaller." << endl;

}

}

RegCloseKey(hAppKey);

}

RegCloseKey(hUninstallKey);

}

}

else if (choice == 6)

{

break;

}

}

return 0;

}