МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

**Кафедра современных технологий программирования**

ЛАТОШ ЕВГЕНИЙ АЛЕКСАНДРОВИЧ

**Утилита для сбора информации о системе**

Курсовая работа

по дисциплине «Операционные системы и системное программирование»

студента 3 курса специальности

1-40 01 01 «Программное обеспечение информационных технологий»

дневной формы получения образования

|  |  |
| --- | --- |
|  | Научный руководитель  Урбан Ольга Ивановна,  старший преподаватель  кафедры современных технологий программирования |
|  |  |

Гродно, 2022

#### РЕЗЮМЕ

Тема курсовой работы

**«Утилита для сбора информации о системе»**

Работа содержит 32 страницы, 24 иллюстраций, 4 использованных источников литературы.

Ключевые слова: утилита, системная информация, WMI, OSD панель, операционная система.

Цель курсовой работы – разработка утилиты для сбора информации о системе.

Объектом исследования выступают утилиты для сбора информации о системе.

Предметом исследования настоящей работы являются функции получения системной информации.

В работе были использованы следующие методы: сравнительный анализ, моделирование, абстрагирование.

**Авторская характеристика работы.**

Программа написана на языке высокого уровня С#. В ней реализованы общие алгоритмы функционирования приложения. Основной задачей данной работы является создание законченного продукта, готового к использованию. Так же реализованные функции могут найти применение в создании новых приложений.

#### SUMMARY

Course work topic

**"Utility for collecting information about the system"**

The work contains 32 pages, 24 illustrations, 4 used literature sources.

Keywords: utility, system information, WMI, OSD panel, operating system.

The purpose of the course work is to develop a utility for collecting information about the system.

The object of the study are utilities for collecting information about the system.

The subject of this work is the functions of obtaining system information.

The following methods were used in the work: comparative analysis, modeling, abstraction.

**Author's characteristics of the work.**

The program is written in the high-level language C #. It implements the general algorithms for the operation of the application. The main task of this work is to create a finished product, ready to use. Also, the implemented functions can be used in the creation of new applications.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ

|  |  |  |
| --- | --- | --- |
| WMI | Windows Management Instrumentation | одна из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows. |
| CPU | Central Processing Unit | центральное обрабатывающее устройство (процессор). |
| GPU | Graphics Processing Unit | отдельное устройство персонального компьютера или игровой приставки, выполняющее графический рендеринг. |
| OSD | On-Screen Display | экранное меню, отображающее показатели системы, обычно такое меню располагается поверх остальных приложений. |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc121610089)

[ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc121610090)

[1.1 Основные аспекты 8](#_Toc121610091)

[1.2 Обзор существующих решений 9](#_Toc121610092)

[1.2.1 Приложение CPU-Z 9](#_Toc121610093)

[1.2.2 Приложение Belarc Advisor 9](#_Toc121610094)

[1.2.3 Приложение System Spec 10](#_Toc121610095)

[1.2.4 Приложение HWiNFO 10](#_Toc121610096)

[1.3 Вывод по главе 1 11](#_Toc121610097)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ УТИЛИТЫ ДЛЯ СБОРА ИНФОРМАЦИИ О СИСТЕМЕ 12](#_Toc121610098)

[2.1 Получение и вывод системной информации 12](#_Toc121610099)

[2.2 Получение текущих процессов 13](#_Toc121610100)

[2.3 Взаимодействие с процессом 14](#_Toc121610101)

[2.3.1 Завершение процесса 14](#_Toc121610102)

[2.3.2 Информация о потоках 14](#_Toc121610103)

[2.4 OSD панель 15](#_Toc121610104)

[2.5 Панель датчиков 16](#_Toc121610105)

[2.6 Увеличение нагрузки на процессор 17](#_Toc121610106)

[2.7 Вывод по главе 2 17](#_Toc121610107)

[ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ УТИЛИТЫ ДЛЯ СБОРА ИНФОРМАЦИИ О СИСТЕМЕ 18](#_Toc121610108)

[3.1 Получение и вывод системной информации 18](#_Toc121610109)

[3.2 Получение текущих процессов 20](#_Toc121610110)

[3.3 Взаимодействие с процессом 21](#_Toc121610111)

[3.3.1 Завершение процесса 21](#_Toc121610112)

[3.3.2 Информация о потоках 22](#_Toc121610113)

[3.4 OSD панель 22](#_Toc121610114)

[3.5 Панель датчиков 24](#_Toc121610115)

[3.6 Увеличение нагрузки на процессор 25](#_Toc121610116)

[3.7 Демонстрация работы реализованного приложения 25](#_Toc121610117)

[3.8 Вывод по главе 3 29](#_Toc121610118)

[ЗАКЛЮЧЕНИЕ 30](#_Toc121610119)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc121610120)

# **ВВЕДЕНИЕ**

Утилиты для сбора информации о системе - это программы, которые собирают все важные, но труднодоступные сведения об оборудовании вашей компьютерной системы. Такого рода данные очень полезны для тех, кто помогает вам решить проблему с вашим компьютером.

Есть и другие полезные применения этих инструментов, такие как предоставление данных о типе оперативной памяти, которая у вас есть, чтобы вы могли купить правильное оборудование на замену, создание списка оборудования при продаже компьютера, отслеживание температуры ваших важных компонентов, и многое другое. Данная информация позволяет не только узнать характеристики вашего оборудования и системы, но и помогает определить был ли подвержен ваш компьютер воздействию вредоносного программного обеспечения.

Всегда полезно перепроверить характеристики вашего рабочего стола перед устранением неполадок и обновлением. Совместимость может быть большой проблемой для настольных компьютеров.

Также данные утилиты достаточно полезны при приобретении нового компьютера. После покупки нового устройства не лишним будет протестировать его производительность или узнать, совпадают ли ваши результаты с данными в различных обзорах.

Цель курсовой работы – разработать утилиту для сбора информации о системе. Данное приложение должно предоставлять пользователю как общую информацию о компьютере, так и более детальную информацию об отдельных участках системы. Предусмотрена возможность вывода информации о процессах и их потоках, работающих в данный момент в система, а также способность завершить выбранный процесс. Предусмотрены вспомогательные окна для отображения температуры датчиков CPU, загруженности CPU и GPU, значения свободной памяти. Также предусмотрена возможность увеличения нагрузки на CPU.

Предусмотрено решение следующих задач:

1) Реализовать получение и вывод системной информации

2) Реализовать вывод текущих процессов

3) Реализовать возможность завершения процесса, а также получение информации о его потоках

4) Реализовать OSD панель

5) Реализовать панель датчиков

6) Реализовать возможность увеличения нагрузки на процессор

# **ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Основные аспекты**

Иногда вам может потребоваться узнать немного больше о вашей машине: подробности об оборудовании, программном обеспечении и т. д.

В настоящее время существует огромное количество программ, помогающих пользователю получить, обобщить и проанализировать информацию о компьютере.

При схожих целях подобные утилиты зачастую довольно сильно различаются по своей реализации, удобству интерфейса, набору инструментов диагностики и функциональности в целом. Среди подобных программ встречаются как узкоспециализированные, предназначенные для детального рассмотрения одной из подсистем компьютера, так и позволяющие провести диагностику системы в целом и всех ее подсистем в отдельности.

Конечно, значительную часть информации, предоставляемой подобными утилитами, можно при желании почерпнуть и из штатных средств Windows, но нужно отметить более удобный и дружественный интерфейс таких утилит и их несомненно большую информативность. Зачастую в состав утилит диагностики и мониторинга разработчики включают тестовые модули, позволяющие на основе несложных, а главное — недлительных синтетических тестов составить более полное представление о компьютерной системе и принять продуманное решение, касающееся способов увеличения ее производительности. Да и просто сбор систематизированной подробной информации о системе способен порой открыть пользователю глаза на причины тех или иных проблем, возникающих при работе с ПК.

## **1.2 Обзор существующих решений**

Чтобы выявить основные аспекты, связанные с реализацией утилиты для сбора информации о системе, необходимо ознакомиться с рынком и установить на какие приложения существует спрос на данный момент и каким требованиям они должны отвечать. Рассмотрим разные приложения, которые могут использоваться чаще всего и имеют богатый функционал, позволяющий достичь нужного результата.

### **1.2.1 Приложение CPU-Z**

Легендарный CPU-Z, это бесплатное приложение системного профилировщика (системного монитора) для Microsoft Windows (для всех версий, начиная с 95), которое определяет центральный процессор, оперативную память, набор микросхем материнской платы и другие аппаратные функции современного персонального компьютера и представляет информацию в одном окне. Он собирает информацию о некоторых основных устройствах вашей системы:

* Имя и номер процессора, кодовое имя, процесс, пакет, уровни кэша.
* Материнская плата и чипсет.
* Тип памяти, размер, тайминги и спецификации модуля (SPD).
* Измерение в реальном времени внутренней частоты каждого ядра, частоты памяти.

### **1.2.2 Приложение Belarc Advisor**

Belarc Advisor создает подробный профиль вашего установленного программного и аппаратного обеспечения, инвентаризации сети, отсутствующих исправлений Microsoft, состояния антивируса, контрольных показателей безопасности и отображает результаты в вашем веб-браузере. Вся информация о вашем профиле ПК хранится на вашем ПК в тайне и не отправляется ни на какой веб-сервер. Работает на Windows 10, 2016, 8.1, 2012 R2, 8, 2012, 7, 2008 R2, Vista, 2008, 2003 SP2, XP SP3. Поддерживаются как 32-битные, так и 64-битные версии Windows. Отлично работает на Windows 10 от Microsoft и macOS от Apple.

### **1.2.3 Приложение System Spec**

System Spec — это бесплатная системная информационная утилита, которая создает спецификации аппаратного и программного обеспечения вашей системы. С помощью System Spec вы можете просмотреть, сохранить и распечатать полную спецификацию вашего ПК. Эта автономная системная информационная утилита также может выполнять различные функции Windows. Дополнительная расширенная информация включает процессор, диски, приложения, дисплей, память, сеть, Интернет, приводы CD/DVD и многое другое.

### **1.2.4 Приложение HWiNFO**

HWiNFO — это профессиональные средства информации и диагностики оборудования, поддерживающие новейшие компоненты, отраслевые технологии и стандарты. Оба инструмента предназначены для сбора и предоставления максимально возможного объема информации об оборудовании компьютеров, что делает их подходящими для пользователей, ищущих обновления драйверов, производителей компьютеров, системных интеграторов и технических экспертов. Полученная информация представлена в логичной и понятной форме и может быть экспортирована в различные типы отчетов.

* Подробная информация об оборудовании — от краткого обзора до глубины всех аппаратных компонентов. Всегда актуальная поддержка новейших технологий и стандартов.
* Мониторинг системы в режиме реального времени — точный мониторинг всех компонентов системы для определения фактического состояния и прогнозирования отказов. Настраиваемый интерфейс с множеством опций.
* Обширная отчетность — несколько типов отчетов, регистрация состояния и взаимодействие с другими инструментами или надстройками.

## **1.3 Вывод по главе 1**

Утилиты для сбора информации о системе — это мощный инструмент для диагностики, который может пригодится во множестве ситуаций при работе с компьютером. Данный инструмент поможет в отслеживании состояния системы и как следствие позволит увеличить срок работоспособности оборудования.

Приложения, рассмотренные в данной главе, позволили выявить основные аспекты, связанные с реализацией приложения. А именно:

1. Реализовать получение информации о CPU
2. Реализовать получение информации о материнской плате
3. Реализовать получение информации о GPU
4. Реализовать получение информации о разделах дисков

# **ГЛАВА 2 ПРОЕКТИРОВАНИЕ УТИЛИТЫ ДЛЯ СБОРА ИНФОРМАЦИИ О СИСТЕМЕ**

## **Получение и вывод системной информации**

Основной задачей утилиты является получение и вывод на экран информации о системе. Информация о системе будет добываться из разных источников: WMI, реестр windows, классы CultureInfo и Environment. Кратко опишу каждый из перечисленных источников:

**CultureInfo** – это класс языка программирования C# пространства имен System.Globalization. Он предоставляет сведения об определенном языке и региональных параметрах. Эти сведения включают имена языков и региональных параметров, систему письма, используемый календарь, порядок сортировки строк и форматы дат и чисел.

**Environment** - это класс языка программирования C# пространства имен System. Предоставляет сведения о текущей среде и платформе, а также необходимые для управления ими средства.

**Реестр windows** - иерархически построенная база данных параметров и настроек в большинстве операционных систем Microsoft Windows. Реестр содержит информацию и настройки для аппаратного обеспечения, программного обеспечения, профилей пользователей, предустановки.

**WMI** - инструментарий управления Windows. WMI — это одна из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows.

После того как мы получили системную информацию, следует ее вывести на экран. Для этой задачи будут реализованы несколько классов называемых контроллерами. Данные классы будут заниматься группировкой полученной информации в логические блоки, которые в последствии будут выводится на экран в зависимости от раздела, выбранного пользователем.

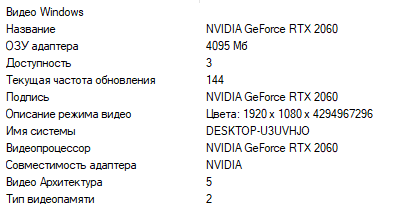


Рис. 2.1 ‒ Графическое представление логического блока

## **Получение текущих процессов**

В приложении предусмотрена возможность вывода текущих процессов. В языке C# получить информацию о процессах можно при помощи метода GetProcesses() класса Process. После чего данная информация передается в контроллер процессов, который формирует таблицу с нужной информацией. В последствии данная таблица выводится на экран.

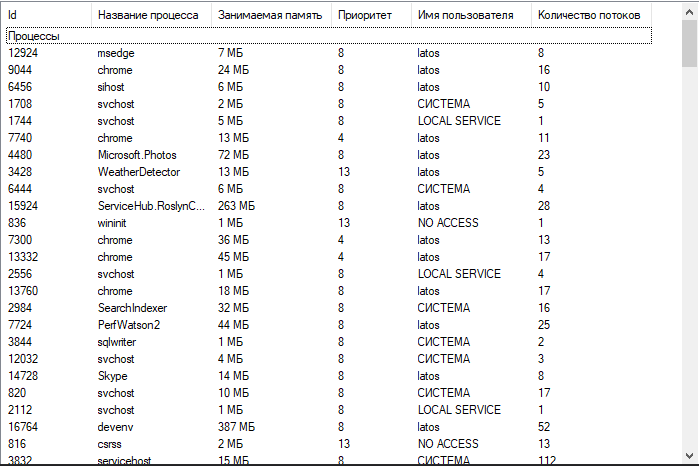


Рис. 2.2 ‒ Таблица процессов

## **Взаимодействие с процессом**

### **2.3.1 Завершение процесса**

Один из основных способов взаимодействия с процессом – это возможность принудительно завершить работу процесса.

Так как на предыдущем этапа была получена информация о процессе, при помощи ее можно завершить работу процесса используя метод Kill().

Чтобы завершить работу процесса, нужно кликнуть правой кнопкой мыши по выбранному процессу, после чего всплывет контекстное меню, в котором следует выбрать пункт “Снять задачу”.

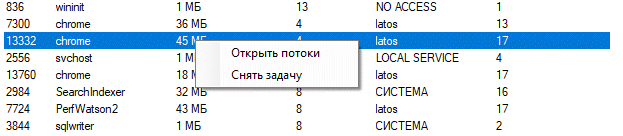


Рис. 2.3 ‒ Демонстрация контекстного меню

### **2.3.2 Информация о потоках**

Также предусмотрена возможность отобразить более подробную информацию о потоках выбранного процесса. Отобразить данную информацию можно если в контекстном меню выбрать пункт “Открыть потоки”, после чего всплывает диалоговое окно с информацией о потоках. В данном окне находится таблица, в которой представлены Id и приоритет для каждого потока.

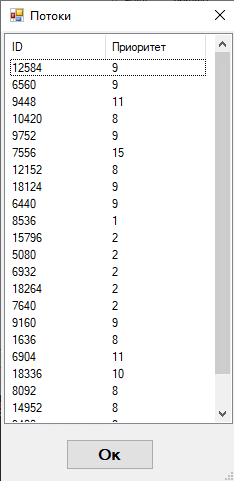


Рис. 2.4 ‒ Демонстрация таблицы потоков

## **OSD панель**

В приложении предусмотрена возможность вывести OSD панель. Данная панель отображает загруженность CPU и GPU в процентах, а также свободную оперативную память в мегабайтах.

Данную информацию можно получить при помощи класса PerformanceCounter. Информация обновляется каждую секунду и выводится на OSD панель. Открыть данную панель можно вызвав контекстное меню и выбрав пункт “OSD панель”. Также данную панель можно закрыть, вызвав контекстное меню и выбрав пункт “Закрыть”.

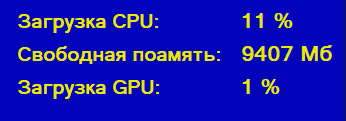


Рис. 2.5 ‒ Демонстрация OSD панели

## **Панель датчиков**

В приложении предусмотрена возможность вывести панель датчиков. Данная панель отображает температуру, предоставляемую датчиками центрального процессора.

Данную информацию можно получить, используя библиотеку OpenHardwareMonitor. В зависимости от количества датчиков формируется размер окна, в котором каждую секунду обновляется значение датчиков. Открыть данную панель можно вызвав контекстное меню и выбрав пункт “Панель датчиков”. Также данную панель можно закрыть, вызвав контекстное меню и выбрав пункт “Закрыть”.

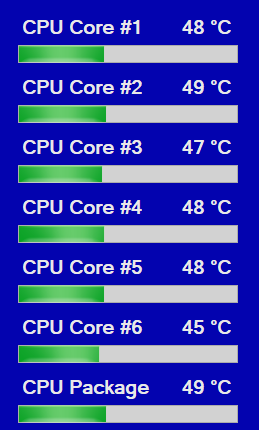


Рис. 2.6 ‒ Демонстрация панели датчиков

## **Увеличение нагрузки на процессор**

В приложении предусмотрена возможность увеличения нагрузки на процессор. На главном окне находится кнопка “Тест”, которая открывает окно с тремя кнопками. При нажатии на кнопку “Старт” запускаются несколько потоков с активным ожиданием, которые нагружают CPU до 100%. При нажатии на кнопки “Стоп” или “Закрыть” запущенные патоки завершаются.

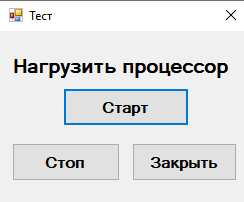


Рис. 2.7 ‒ Окно увеличения нагрузки

## **2.7 Вывод по главе 2**

Во второй главе были описаны способы получения системной информации. Также были спроектированы основные возможности приложения и выбраны способы их реализации. После того, как были спроектированы основные возможности приложения можно приступать к их реализации программным кодом.

В качестве среды разработки приложения будет использована Visual Studio 2019 Community. Используемый язык программирования – С#.

# **ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ УТИЛИТЫ ДЛЯ СБОРА ИНФОРМАЦИИ О СИСТЕМЕ**

## **Получение и вывод системной информации**

Информация о системе будет добываться из разных источников: WMI, реестр windows, классы CultureInfo и Environment.

Получение информации через классы Environment и CultureInfo является достаточно простым, достаточно вызвать свойство, которое вам надо. Поэтому я сосредоточусь на описании методов получения информации из оставшихся двух источников.

Получение информации из реестра windows реализовано в классе RegistryAccess.

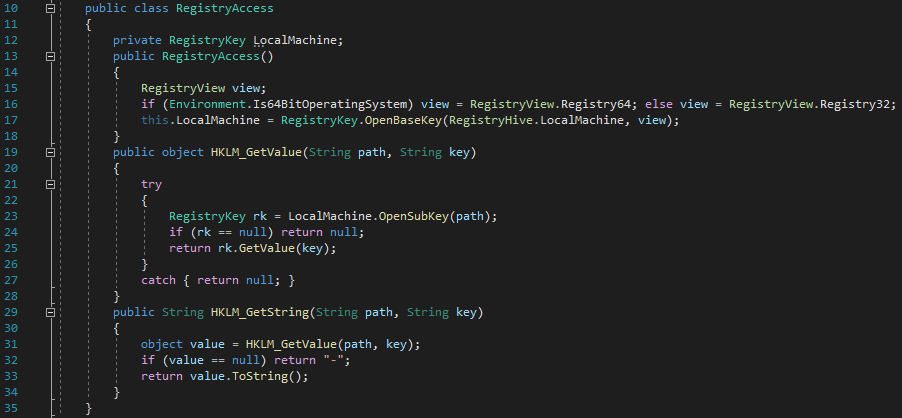


Рис. 3.1 ‒ Класс реализующий получение информации из реестра

В конструкторе класса, в строке 16 в зависимости от разрядности системы определяется представление реестра. В строке 17 происходит открытие раздела реестра “LocalMachine”.

В методе public object HKLM\_GetValue(String path, String key) реализуется получение значения из реестра. Данный метод принимает два входных параметра (путь к директорию и название параметра).

В строке 23 происходит получение доступа к нужному директорию. Если данного директория не существует в строке 24 метод возвращает null. В противном случае метод возвращает значение параметра.

В методе public String HKLM\_GetString(String path, String key) реализуется получение строки значения из реестра. Данный метод принимает два входных параметра (путь к директорию и название параметра).

В строке 31 происходит вызов метода, описанного выше, который возвращает значение нужного параметра. Если параметра не существует или значение параметра пустое, метод повращает прочерк. В противном случае метод возвращает строку выбранного параметра.

Далее рассмотрим получение информации через WMI. Для начала нужно определить область управления и объектный запрос.



Рис. 3.2 ‒ Объявление области управления и объектного запроса

Далее следует определить ManagementObjectSearcher (коллекцию объектов управления).



Рис. 3.3 ‒ Демонстрация получения объекта управления

В строке 20 определяется объект ManagementObjectSearcher в конструктор, которого первым параметром передается область управления, а вторым объектный запрос.

В строке 21 метод Get() возвращает коллекцию объектов управления, из которой можно получить нужный объект.

Получение информации из полученного объекта, реализовано в методе private object GetInformation(String str). Данный метод принимает один параметр (название свойства).



Рис. 3.4 ‒ Демонстрация получения значения

В строке 106 возвращается значение, переданного названия свойства.

Далее классы контроллеры группируют полученную информацию, пример группировки информации демонстрирует метод public InformationBlock GetBasicInformation() класса VIdeoInformationController.

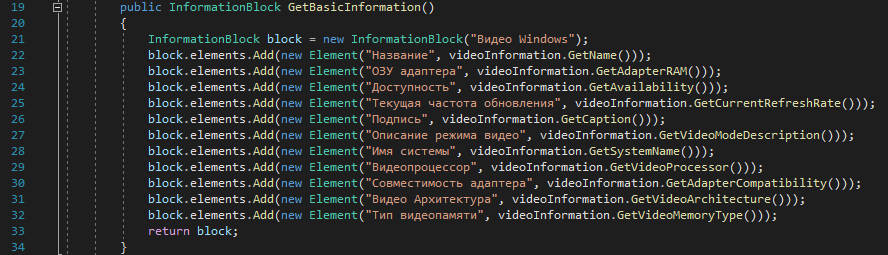


Рис. 3.5 ‒ Демонстрация группировки информации

В строке 21 объявляется логический блок, в конструктор которого передается его название.

В строках 22-32 в данный блок добавляется полученная информация.

В строке 33 метод возвращает сформированный блок.

## **Получение текущих процессов**

Получение информации о процессах реализована в методе public void UpdateProcessesInfo() класса Processes.

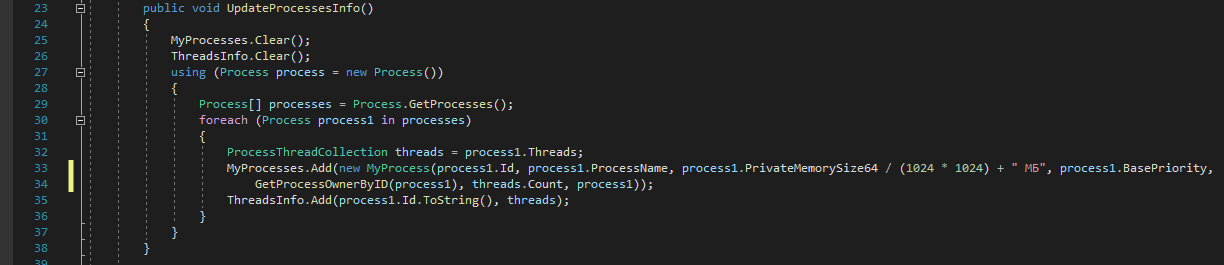


Рис. 3.6 ‒ Метод реализующий получение информации о процессах

В строках 25-26 происходит очистка коллекций с информацией.

В строке 29 происходит получение информации о процессах.

В строке 32 происходит получение информации о потоках каждого процесса.

В строках 33-35 реализовано заполнение коллекций необходимой информацией.

## **Взаимодействие с процессом**

### **3.3.1 Завершение процесса**

Завершение процесса реализовано в методе public void KillProcess(String id) класса ProcessesController.

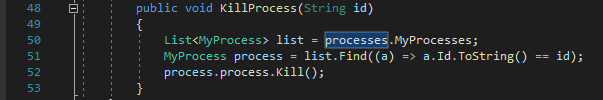


Рис. 3.7 ‒ Метод реализующий завершение процесса

Данный метод принимает один параметр (Id процесса).

В строке 50 происходит получение списка всех процессов.

В строке 51 происходит поиск нужного процесса по Id.

В строке 52 вызывается метод Kill(), который завершает нужный процесс.

### **3.3.2 Информация о потоках**

Получение информации о потоках реализовано в методе public List<MyThread> GetThreads(String id) класса ProcessesController.

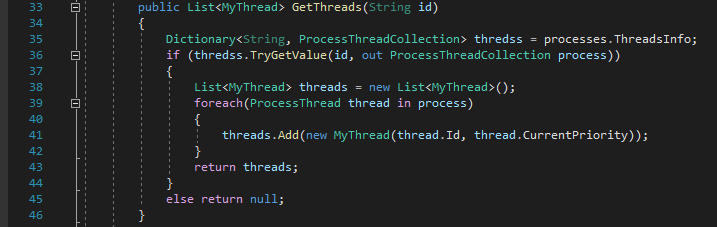


Рис. 3.8 ‒ Метод реализующий получение информации о потоках

Данный метод принимает один параметр (Id процесса).

В строке 35 происходит получение коллекции, в которой, каждая коллекция потоков соответствует определенному Id процесса.

В строках 36, 45 происходит попытка получить коллекцию потоков по Id процесса, если такого процесса не существует, возвращается значение null.

В строке 38 объявляется коллекция с необходимой информацией.

В строках 39-42 происходит перебор каждого потока.

В строке 41 формируется новая коллекция. После формирования коллекции в строке 43 метод ее возвращает.

## **OSD панель**

Для получения значение загруженности CPU и количество свободной оперативной памяти используется класс PerformanceCounter. В данный класс передается соответствующие названия категории и названия счетчика.

Получение загруженности GPU происходит также при помощи класса PerformanceCounter. Но в данном случает нет простого названия категории, поэтому придется вычислить ее самостоятельно.

Получение загруженности GPU реализовано в методе public async Task<float> getValue() класса GPUUsage.

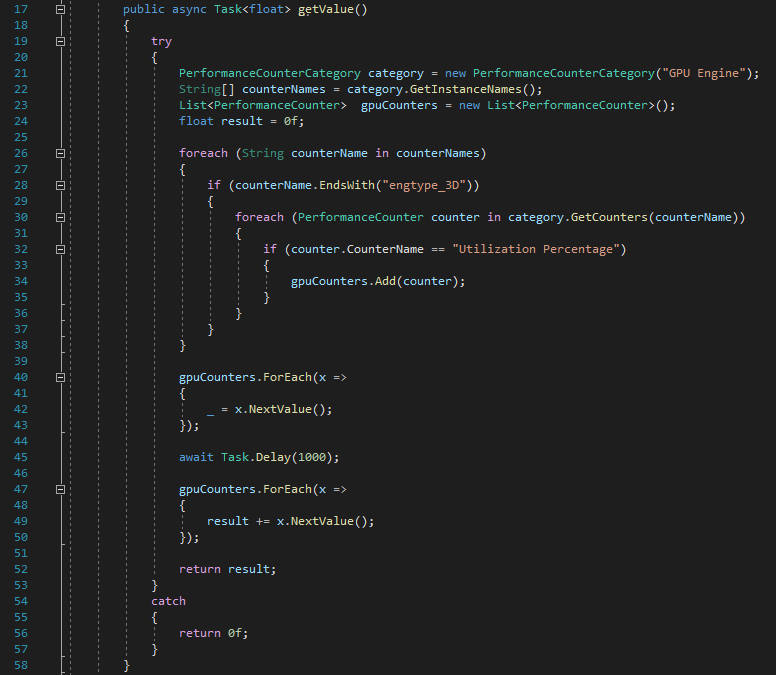


Рис. 3.9 ‒ Метод реализующий получение загруженности GPU

В строке 21 объявляется нужная категория. Далее в троке 22 происходит получение названий имен всех объектов, связанных с данной категорией.

В строке 23 объявляется список счетчиков.

В строках 26-38 происходит перебор всех названий объектов, связаных с категорией. Далее в строке 28 выбираются объекты, названия которых заканчиваются на "engtype\_3D". Далее в строках 30-36 происходит получение счетчиков с нужным названием и их перебор. После чего в строке 32 выбираются счетчики, которые считают процент загруженности. В строке 34 счетчик добавляется в список.

В строках 40-43 у всех счетчиков пропускается текущее значение.

В строке 45 процесс останавливается на секунду.

В строках 47-50 значения всех счетчиков суммируется и возвращается в строке 52.

## **Панель датчиков**

Получение температуры процессора реализовано при помощи библиотеки OpenHardwareMonitor.

Получение температуры датчиков процессора реализовано в методе public Dictionary<String, float?> GetCpuTemp() класса CpuTemperature.

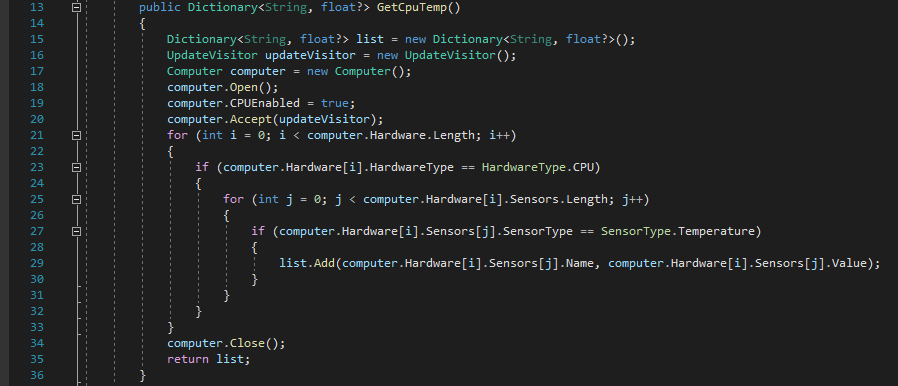


Рис. 3.10 ‒ Метод реализующий получение температуры CPU

В строке 15 происходит объявление списка результатов.

В строках 16-20 происходит настройка объектов библиотеки.

В строках 21-33 происходит перебор всех устройств и в строке 23 выбирается только процессор. Дале в троках 25-31 происходит перебор всех датчиков процессора и в строке 27 выбираются датчики температуры. После чего в троке 29 в список добавляется название датчика и его значение.

В строке 34 закрывается объект библиотеки и в строке 35 возвращается результат.

## **Увеличение нагрузки на процессор**

Алгоритм увеличения нагрузки на процессор реализован в событии нажатии на кнопку private void button2\_Click(object sender, EventArgs e) формы Tests.

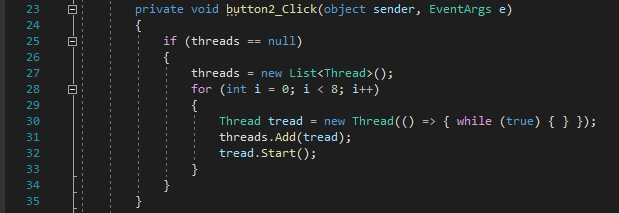


Рис. 3.11 ‒ Метод реализующий увеличение нагрузки на процессор

В строке 25 происходит проверка на то что нагрузки нету. Далее в троке 27 объявляется список потоков.

В строках 28-33 создается 8 потоков с активным ожиданием. Далее в строке 31 поток добавляется в список и в строке 32 потоки запускаются.

## **Демонстрация работы реализованного приложения**

Для запуска приложения ему требуется предоставить права администратора.

После запуска приложения пользователя встречает главное окно, в котором слева находится список категорий, а справа таблица с информацией. Также на форме присутствуют две кнопки, для обновления таблицы и перехода на окно увеличения нагрузки.

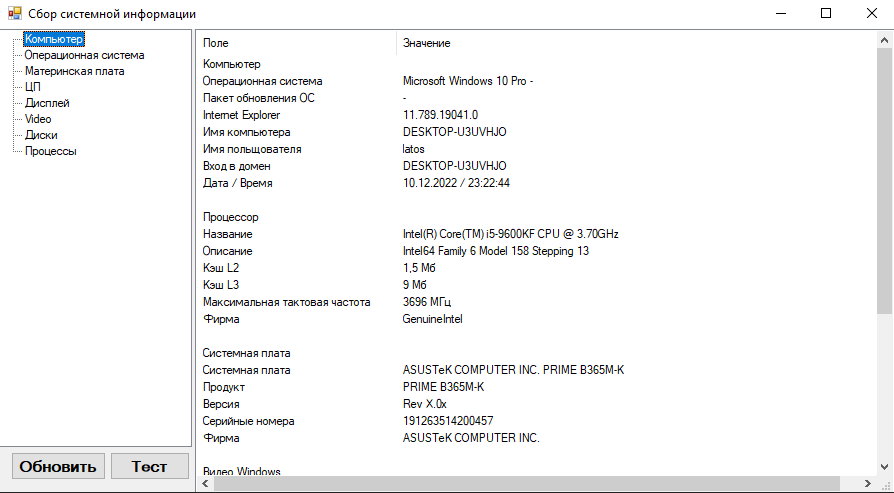


Рис. 3.12 ‒ Главное окно

После нажатия на кнопку “Тест” открывается окно увеличения нагрузки. На данном окне находятся три кнопки: “Старт”, “Стоп”, “Закрыть”. При нажатии на кнопку “Старт” начинается увеличение нагрузки на CPU. При нажатии на кнопку “Стоп” увеличение нагрузки останавливается, а при нажатии на кнопку “Закрыть” увеличение нагрузки останавливается и окно закрывается.

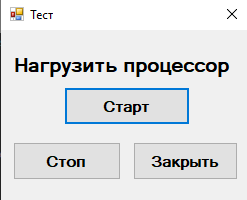


Рис. 3.13 ‒ Окно увеличения нагрузки

Если на главном окне открыть контекстное меню и выбрать пункт “OSD панель”, откроется OSD панель в которой отображается загруженность CPU и GPU в процентах, а также свободная оперативная память в мегабайтах. Данная информация обновляется каждую секунду.

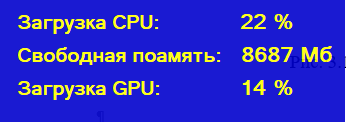


Рис. 3.14 ‒ OSD панель

Если на главном окне открыть контекстное меню и выбрать пункт “Панель датчиков”, откроется панель датчиков в которой отображается температура, предоставляемая датчиками центрального процессора. Данная информация обновляется каждую секунду.

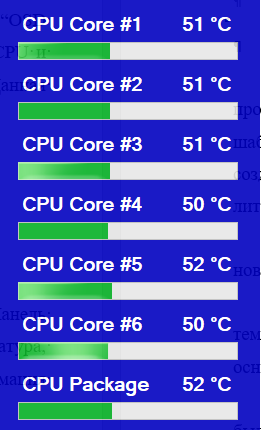


Рис. 3.15 ‒ Панель датчиков

Если на главном окне выбрать категорию “Процессы” откроется таблица процессов, с которыми можно взаимодействовать через контекстное меню.

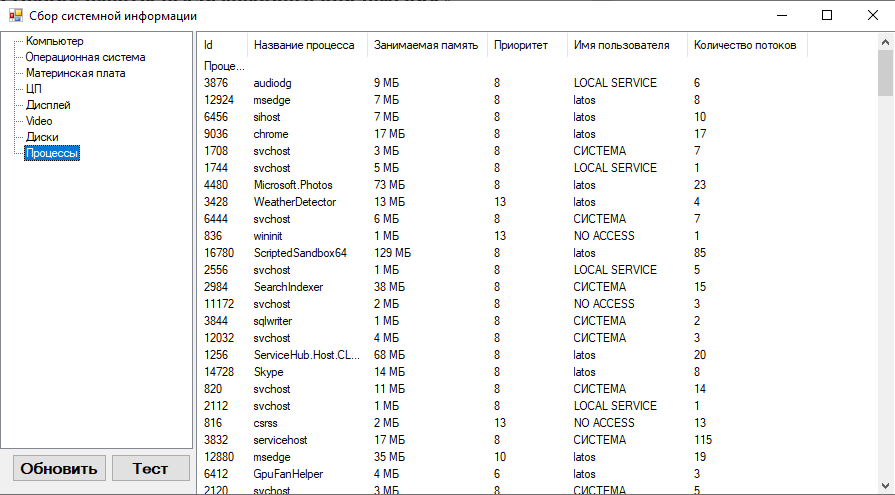


Рис. 3.16 ‒ Главное окно с таблицей процессов

Если на таблице процессов открыть контекстное меню и выбрать пункт “Открыть потоки”. Откроется диалоговое окно с информацией о потоках. На данном окне присутствует таблица потоков и кнопка закрытия окна.

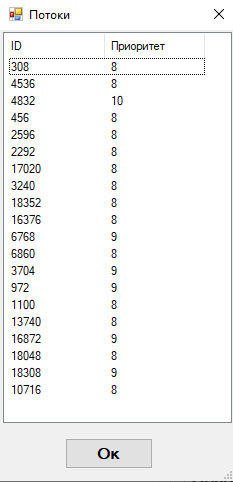


Рис. 3.17 ‒ Окно с таблицей потоков

## **Вывод по главе 3**

Таким образом мы разобрались с функционалом приложения и его реализацией. В данной главе была описана реализация основных методов данного проекта.

В будущем, проект может представлять, как практическую, так и академическую пользу. Поскольку в случае его удачной реализации, его можно будет развивать дальше, а конечная версия, написанная в максимально простом виде, может в дальнейшем использоваться как шаблон, которым смогут воспользоваться все желающие для того, чтобы ознакомиться с получением системной информации на конкретном примере.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы были изучены различные способы получения системной информации. На этапе разработки программы были получены практические навыки создания утилиты. Благодаря использованию дополнительной учебной литературы, особых трудностей при написании работы не возникло.

Дальнейшую модификацию программы вижу в возможности дополнения ее новыми функциями и возможностями.

В первой главе курсовой работы была доказана актуальность выбранной темы. Также были рассмотрены существующие решения и были выявлены основные аспекты, связанные с реализацией приложения.

Во второй главе были рассмотрены различные способы получения системной информации. А также были спроектированы основные методы, использованные в курсовой работе.

В третьей главе была описана программная реализация основных методов, использованных в курсовой работе.

В заключении хочется отметить, что данное приложение можно будет развивать дальше, добавляя новы возможности. Кроме того, реализованные методы могут быть применены не только в данном проекте, их можно использовать в различных приложениях. Также данный проект может использоваться в качестве примера для ознакомления со способами получения системной информации.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Павловская Т. А. C# программирование на языке высокого уровня / Т. А. Павловская – СПб.: Питер Пресс, 2009. — 432 с.
2. codeguru.com [Электронный ресурс]: Working with the Windows Registry in C#. -URL: <https://www.codeguru.com/csharp/windows-registry-c-sharp/> (дата обращения: 20.09.2022)
3. learn.microsoft.com [Электронный ресурс]: Удаленное подключение к WMI с помощью C#. -URL: <https://learn.microsoft.com/ru-ru/windows/win32/wmisdk/connecting-to-wmi-remotely-with-c-> (дата обращения: 23.09.2022)
4. instructables.com [Электронный ресурс]: Using Open Hardware Monitor to Get CPU Temperature in C#. -URL: <https://www.instructables.com/Using-Open-Hardware-Monitor-to-Get-CPU-Temperature/> (дата обращения: 15.10.2021)