МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ АКАДЕМИКА И.Г. ПЕТРОВСКОГО**»**

Институт естественно-научный

Факультет физико-математический

Кафедра информатики и прикладной математики

**КУРСОВАЯ РАБОТА**

**Разработка инструмента управления процессами и мониторинга системных ресурсов персонального компьютера**

Код, направление подготовки: 01.03.02 Прикладная математика и информатика  
Направленность (профиль) программы: Системное программирование и компьютерные технологии

Обучающийся  
Лемешев К.А.  
3 курса 1 группы   
очной формы обучения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(подпись)

Руководитель  
Рослякова Е.А.  
Ст. преподаватель кафедры ИПМ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(подпись)

Брянск 2024 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc162781737)

[ГЛАВА 1 УПРАВЛЕНИЕ ПРОЦЕССАМИ 5](#_Toc162781738)

[1.1 Анализ понятий «процесс», «поток» и «ресурс» 5](#_Toc162781739)

[1.2 Основы управления процессами компьютера: роль функций ОС. 8](#_Toc162781740)

[1.3 Мониторинг и оптимизация ОС WINDOWS 13](#_Toc162781741)

[ГЛАВА 2 РЕАЛИЗАЦИЯ ПРАКТИЧЕСКОГО ЗАДАНИЯ 18](#_Toc162781742)

[2.1 Управление процессами ОС WINDOWS 10 18](#_Toc162781743)

[2.2 Мониторинг системных ресурсов 26](#_Toc162781744)

[2.3 Тестирование программы 31](#_Toc162781745)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc162781746)

[ПРИЛОЖЕНИЕ 36](#_Toc162781747)

# ВВЕДЕНИЕ

Управление задачами и мониторинг системных ресурсов компьютера играет ключевую роль в представлении целей и значимости исследования пользователя. В современном мире, где компьютеры стали неотъемлемой частью повседневной жизни, важно обеспечить эффективное управление задачами и контролировать использование ресурсов системы.

В современном мире возникает необходимость мониторинга операционной системы т.к. компьютеры постоянно выходят из строя, либо работают ни на полном быстродействии. В связи с этим стали появляться новые профессии такие как: системный администратор. С развитием операционных систем требования к мониторингу постоянно менялись, но принцип мониторинга остался тем же – контроль за системой.

Сейчас профессия системного администратора очень распространена, в его функции входит следить за системой (приведение системы в порядок), настройка компьютера на максимальное быстродействие, и контроль за устройствами. Раньше это было не так просто сделать т.к. не было нужного программного обеспечения. Но благодаря современным технологиям и современным программам мониторинг системы стал максимально простым. Во всех современных операционных системах существуют стандартные программы мониторинга.

Актуальность выбранной темы подтверждается тем, что разработка такого инструмента позволит пользователям оптимизировать работу компьютерных систем, повысить производительность и эффективность использования ресурсов ПК. Контроль за задачами и мониторинг системных ресурсов поможет предотвратить перегрузку системы, улучшить отклик приложений и обеспечить стабильную работу компьютера.

Объект исследования: Процессы и системные ресурсы персонального компьютера

Процессы и системные ресурсы компьютера являются основными компонентами, которые определяют его производительность и функциональность. В персональном компьютере происходят различные процессы, такие как обработка данных, загрузка программ, взаимодействие с пользователем и другие. Системные ресурсы, такие как оперативная память, процессорное время, дисковое пространство и сетевые соединения, также играют важную роль в определении производительности системы.

Предмет исследования: методы, инструменты и технологии мониторинга и управления системными ресурсами и процессами персонального компьютера

Целью данного исследования является: создание удобного и эффективного программного инструмента, для контроля процессов, запущенных на компьютере, а также за использованием системных ресурсов.

Разработка такого инструмента позволит повысить эффективность работы ПК, улучшить процессы управления задачами и обеспечить стабильную и оптимальную работу компьютерной системы.

В ходе данной курсовой работы будут рассмотрены основные проблемы, с которыми сталкиваются пользователи при управлении задачами и мониторинге ресурсов ПК, представлены методы исследования, использованные при разработке инструмента, а также обсуждены практические аспекты применения разработанного решения в повседневной работе с компьютером.

Работа состоит из двух глав, введения, библиографического списка и приложений.

В первой главе подробно рассматриваются понятия и основные свойства работы диспетчера задач и ресурсов ПК.

Во второй главе на практике анализируется и строится инструмент для более точного контроля за процессами.

В ходе работы использовались учебники, учебные пособия, статьи из профильных журналов и электронные ресурсы, общий объем использованных ресурсов составил 10 единиц.

# ГЛАВА 1 УПРАВЛЕНИЕ ПРОЦЕССАМИ

# 1.1 Анализ понятий «процесс», «поток» и «ресурс»

Процессом является выполняемая программа, которая включает текущее значение счетчика команд, регистров и переменных.

Алан Шоу дал следующее определение процессу («Логическое проектирование операционных систем». – М.: Мир, 1981. – 360 с.), последовательный процесс (иногда называемый «задача») есть работа, производимая последовательным процессором при выполнении программы с ее данными

Это определение подчеркивает последовательный характер процесса, т.е. выполнение команд в определенном порядке. Термин «задача» понимается как синоним термина «процесс». Процесс – понятие динамическое. Программа – это текст, процесс – выполнение этого текста. На практике говорят: «программа вызывает функцию», «программа ждет ввода» и т.п., однако, строго говоря, правильнее было бы «процесс, выполняющий программу, вызывает…».

В многозадачных системах зачастую одна и та же программа может запускаться несколько раз (например, можно несколько раз открыть текстовый редактор Notepad для разных файлов или графический редактор). Это означает, что несколько процессов могут использовать одну и ту же программу, но с разными данными.

Поток – это последовательность исполняемых команд. В чем состоят принципиальные отличия в понятиях «процесс» и «поток»?

В операционных системах, где существуют и процессы, и потоки, процесс рассматривается операционной системой как заявка на потребление всех видов ресурсов, кроме одного – процессорного времени. Этот последний важнейший ресурс распределяется операционной системой между другими единицами работы – потоками, которые и получили свое название благодаря тому, что они представляют собой последовательности (потоки выполнения) команд.

Элементы, которые потоки используют совместно: адресное пространство, глобальные переменные, открытые файлы, дочерние процессы, необработанные аварийные сигналы, сигналы и их обработки, информация об использовании ресурсов.

Понятию «поток» соответствует последовательный переход процессора от одной команды программы к другой.

Элементы индивидуальные для каждого потока: элементы потока, счетчик команд, регистры, стек, состояние.

ОС распределяет процессорное время между потоками. Процессу ОС назначает адресное пространство и набор ресурсов, которые совместно используются всеми его потоками.

Для того чтобы процессы не могли вмешаться в распределение ресурсов, а также не могли повредить коды и данные друг друга, важнейшей задачей ОС является изоляцияодного процесса от другого. Для этого операционная система обеспечивает каждый процесс отдельным виртуальным адресным пространством, так что ни один процесс не может получить прямого доступа к командам и данным другого процесса.

Другим основополагающим понятием, тесно связанным с управлением процессами, является понятие ресурса.

Говоря упрощенно, ресурс – это нечто дефицитное в вычислительной системе. К важнейшим ресурсам любой системы относятся процессор (точнее сказать, процессорное время), основная память, периферийные устройства, файлы. В зависимости от конкретной ОС, к дефицитным ресурсам могут относиться места в таблице процессов или в таблице открытых файлов, буферы кэша, блоки в файле подкачки и другие системные структуры данных.

Если два процесса никак не связаны логикой своей работы, то никакой синхронизации или обмена данными между ними нет, однако такие процессы все же могут оказывать косвенное влияние друг на друга вследствие конкуренции за ресурсы. Многозадачная ОС управляет доступом процессов к ресурсам. В некоторых случаях система временно закрепляет ресурс за одним процессом, отказывая другим процессам в доступе или заставляя их ждать освобождения ресурса. В других случаях оказывается возможным совместный доступ нескольких процессов к одному ресурсу.

Под ресурсом понимается любой аппаратный или программный объект, который может понадобиться для работы процессов и доступ, к которому может при этом вызвать конкуренцию процессов.

Основные задачи управления процессами. Под управлением процессами понимаются процедуры ОС, обеспечивающие запуск системных и прикладных программ, их выполнение и завершение.

В однозадачных ОС управление процессами решает следующие задачи: загрузка программы в память, подготовка ее к запуску и запуск на выполнение; выполнение системных вызовов процесса; обработка ошибок, возникших в ходе выполнения; нормальное завершение процесса; прекращение процесса в случае ошибки или вмешательства пользователя.

Все эти задачи решаются сравнительно просто. В многозадачном режиме добавляются значительно более серьезные задачи: эффективная реализация параллельного выполнения процессов на единственном процессоре, переключение процессора между процессами; выбор очередного процесса для выполнения с учетом заданных приоритетов процессов и статистики использования процессора; исключение возможности несанкционированного вмешательства одного процесса в выполнение другого; предотвращение или устранение тупиковых ситуаций, возникающих при конкуренции процессов за системные ресурсы; обеспечение синхронизации процессов и обмена данными между ними.

# 1.2 Основы управления процессами компьютера: роль функций ОС.

В операционной системе компьютера происходит управление всеми ресурсами и процессами, необходимыми для работы компьютера. Она является связующим звеном между аппаратным обеспечением и пользовательскими программами. Операционная система выполняет множество функций, таких как управление процессами, управление задачами и управление оборудованием компьютера. Рассмотрим основные понятия и свойства операционной системы, а также ее роль в работе компьютера.

Операционная система (ОС) – это программное обеспечение, которое управляет ресурсами компьютера и предоставляет пользователю интерфейс для взаимодействия с компьютером.

ОС выполняет ряд важных функций, включая управление процессами, управление задачами и управление оборудованием компьютера.

Операционная система является основным компонентом компьютерной системы и обеспечивает эффективное использование ресурсов компьютера, таких как процессор, память, диски и периферийные устройства.

ОС также обеспечивает защиту данных и программ от несанкционированного доступа, а также обеспечивает механизмы для управления файлами и сетевыми соединениями.

Существует множество различных операционных систем, включая Windows, macOS, Linux и многие другие. Каждая ОС имеет свои особенности и функции, но все они выполняют общую задачу – управление компьютером и обеспечение работы приложений и пользовательского взаимодействия.

Управление процессами – это одна из основных функций операционной системы. С точки зрения операционной системы, процесс – это программа, которая выполняется на компьютере. ОС отвечает за создание, запуск, приостановку, возобновление и завершение процессов.

ОС управляет процессами, чтобы обеспечить эффективное использование ресурсов компьютера. Когда процесс запускается, ОС выделяет ему необходимые ресурсы, такие как память и процессорное время. Она также контролирует доступ процессов к общим ресурсам, чтобы избежать конфликтов и обеспечить безопасность данных.

ОС может приостанавливать процессы, когда они неактивны или не нуждаются в ресурсах, чтобы освободить место для других процессов. Она также может возобновлять приостановленные процессы, когда они снова становятся активными. Когда процесс завершается, ОС освобождает занимаемые им ресурсы и удаляет его из системы.

Управление процессами также включает планирование выполнения процессов. ОС определяет, какие процессы должны выполняться в данный момент и в каком порядке. Она использует различные алгоритмы планирования, чтобы обеспечить справедливое распределение ресурсов и оптимальную производительность системы.

Управление задачами – это одна из основных функций операционной системы, которая отвечает за управление выполнением задач или программ на компьютере. Задача представляет собой набор инструкций, которые должны быть выполнены компьютером.

Операционная система отвечает за создание, запуск, приостановку, возобновление и завершение задач. Она также обеспечивает механизмы для взаимодействия между задачами и управления их приоритетами.

Операционная система предоставляет пользователю или программисту возможность создавать новые задачи. Пользователь может запустить программу или открыть новый документ, и ОС создаст соответствующую задачу для выполнения этой работы.

После создания задачи, операционная система запускает ее для выполнения. Она выделяет необходимые ресурсы, такие как память и процессорное время, и передает управление задаче.

Операционная система может приостановить выполнение задачи, когда она не нуждается в ресурсах или когда другая задача требует больше ресурсов. Приостановленная задача сохраняет свое состояние, чтобы можно было возобновить ее позже.

Возобновление задач

Когда задача приостановлена, операционная система может возобновить ее выполнение, когда она снова становится активной. Задача восстанавливает свое сохраненное состояние и продолжает выполнение с того места, где она была приостановлена.

Когда задача завершает свое выполнение, операционная система освобождает занимаемые ею ресурсы и удаляет ее из системы. Завершение задачи может быть вызвано ее завершением по окончании работы или пользовательским запросом на завершение.

Управление приоритетами задач

Операционная система также управляет приоритетами задач. Она может назначать каждой задаче определенный приоритет, который определяет ее важность и порядок выполнения. Задачи с более высоким приоритетом получают больше ресурсов и выполняются быстрее, чем задачи с более низким приоритетом.

Управление оборудованием компьютера

Операционная система играет важную роль в управлении оборудованием компьютера. Она обеспечивает взаимодействие между программным обеспечением и аппаратными компонентами, такими как процессор, память, жесткий диск, периферийные устройства и т.д.

Для работы с различными устройствами операционная система использует драйверы. Драйверы – это программы, которые обеспечивают коммуникацию между операционной системой и конкретным устройством. Они позволяют операционной системе распознавать и управлять устройствами, обрабатывать ввод и вывод данных, а также обеспечивать совместимость с различными моделями и производителями устройств.

Операционная система также управляет распределением ресурсов компьютера между различными программами и задачами. Она контролирует доступ к процессору, памяти, диску, сети и другим ресурсам, чтобы обеспечить эффективное использование ресурсов и предотвратить конфликты и перегрузки.

Планирование задач включает в себя следующее. Она определяет порядок выполнения задач, устанавливает приоритеты, распределяет ресурсы и контролирует выполнение задач. Это позволяет операционной системе эффективно использовать ресурсы и обеспечить отзывчивость системы.

Обработка ошибок и исключительных ситуаций

Операционная система также обрабатывает ошибки и исключительные ситуации, которые могут возникнуть во время работы компьютера. Она может обнаруживать и исправлять ошибки, предоставлять сообщения об ошибках и предлагать решения проблем. Операционная система также может обеспечивать защиту от вредоносного программного обеспечения и других угроз безопасности.

Функции операционной системы

1. Управление ресурсами

Операционная система управляет ресурсами компьютера, такими как процессор, память, диски, сетевые устройства и периферийные устройства. Она распределяет ресурсы между запущенными задачами, чтобы обеспечить их эффективное использование. Операционная система также контролирует доступ к ресурсам и предотвращает конфликты при их использовании.

1. Управление процессами

Операционная система управляет выполнением процессов, которые являются основными единицами работы в компьютере. Она планирует и распределяет процессорное время между процессами, обеспечивая справедливое и эффективное выполнение задач. Операционная система также отслеживает состояние процессов, обеспечивает их взаимодействие и контролирует их выполнение.

1. Управление задачами

Операционная система управляет выполнением задач, которые представляют собой более крупные и сложные единицы работы. Она позволяет запускать и завершать задачи, управлять их приоритетами и ресурсами, а также обеспечивать их взаимодействие. Операционная система также может предоставлять средства для планирования и отслеживания выполнения задач.

1. Управление оборудованием компьютера

Операционная система управляет оборудованием компьютера, таким как диски, сетевые устройства, принтеры и другие периферийные устройства. Она обеспечивает доступ к оборудованию, управляет его конфигурацией и настройками, а также обеспечивает взаимодействие с ним. Операционная система также может предоставлять драйверы и другие программные компоненты для поддержки работы с конкретным оборудованием.

1. Обработка ошибок и исключительных ситуаций

Операционная система также обрабатывает ошибки и исключительные ситуации, которые могут возникнуть во время работы компьютера. Она может обнаруживать и исправлять ошибки, предоставлять сообщения об ошибках и предлагать решения проблем. Операционная система также может обеспечивать защиту от вредоносного программного обеспечения и других угроз безопасности.

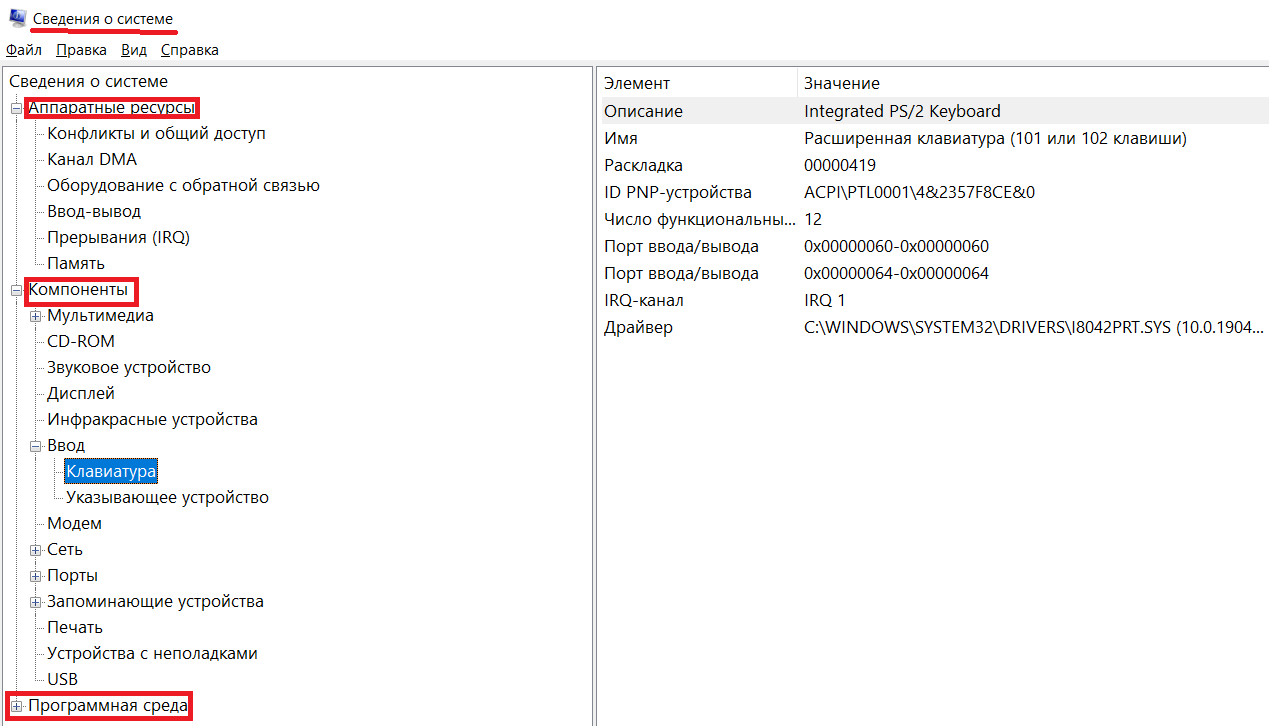
# 1.3 Мониторинг и оптимизация ОС WINDOWS

Мониторинг – система сбора или регистрации, хранения и анализа небольшого количества ключевых (явных или косвенных) признаков или параметров описания данного объекта для вынесения суждения о поведении или состоянии данного объекта в целом.

В этом смысле мониторинг реализуется программным модулем Msinfo32(в папке C:\Program Files\Common Files\microsoft shared\MSInfo).

После запуска программы открывается окно, которое называется «Сведения о системе». В нем приведен список категорий, а в правой части программы – подробные сведения о каждой из них. К этим категориям относятся:

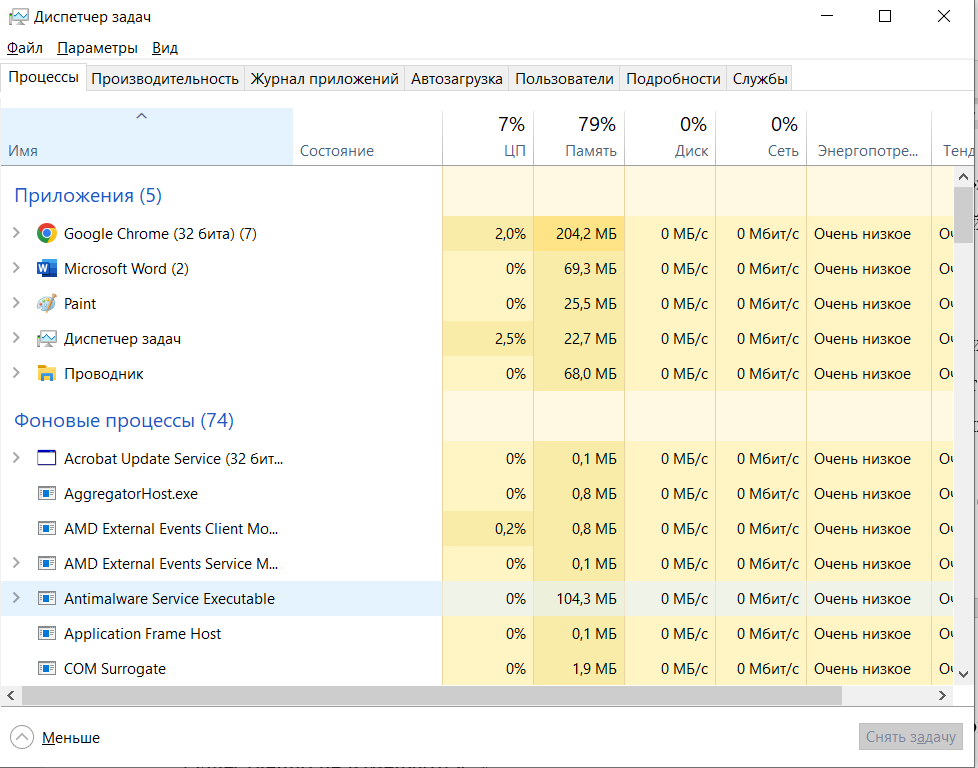
1. Сведения о системе. Содержит общие сведения о компьютере и операционной системе, такие как имя компьютера и его изготовитель, тип используемой BIOS, а также объем установленной памяти.
2. Аппаратные ресурсы. Содержит дополнительные сведения об оборудовании компьютера, которые включает в себя: конфликты и общий доступ, канал DMA, оборудование с обратной связью, ввод-вывод, прерывания, память.
3. Компоненты. Содержит перечень установленных устройств: дисководов, звуковых устройств, модемов и других компонентов.
4. Программная среда. Содержит сведения о драйверах, сетевых подключениях и другую информацию, связанную с программами. (Рис.1)

Рисунок – 1 Сведения о системе

Мониторинг – процесс наблюдения и регистрации данных о каком-либо объекте на неразрывно примыкающих друг к другу интервалах времени, в течение которых значения данных существенно не изменяются.

Мониторинг, в дугом смысле, реализует программный модуль Taskmgr.exe (в папке C:\Windows\System32) – «Диспетчер задач». Диспетчер задач отображает приложения, процессы и службы, которые в текущий момент запущены на компьютере. С его помощью можно контролировать производительность компьютера или завершать работу приложений, которые не отвечают.

После запуска программы открывается окно «Диспетчер задач Windows», содержащее вкладки: процессы, производительность, журнал приложений, автозагрузка, пользователи, подробности, службы. (Рис.2)

Рисунок 2 Диспетчер задач

+-

Различают мониторинг параметров и мониторинг состояния объекта.

Мониторинг параметров – наблюдение за какими-либо параметрами. Результат мониторинга параметров представляет собой совокупность измеренных значений параметров, получаемых на неразрывно примыкающих друг к другу интервалах времени, в течение которых значения параметров существенно не изменяются.

Мониторинг состояния – наблюдение за состоянием объекта для определения и предсказания момента перехода в предельное состояние.

Результат мониторинга состояния объекта представляет собой совокупность диагнозов составляющих его субъектов, получаемых на неразрывно примыкающих друг к другу интервалах времени, в течение которых состояние объекта существенно не изменяется. Принципиальным отличием мониторинга состояния от мониторинга параметров является наличие интерпретатора измеренных параметров в терминах состояния – экспертной системы поддержки принятия решений о состоянии объекта и дальнейшем управлении.

Мониторинг состояния реализует программный модуль Dxdiag.exe (в папке C:\Windows\System32) – «Средство диагностики DirectX».

Отображает сведения о компонентах и драйверах интерфейса прикладного программирования приложений (API) Microsoft DirectX в системе. Позволяет проверить работу звуковой и графической подсистем.

DirectX – это набор [API,](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) разработанных для решения задач, связанных с программированием под [Microsoft Windows.](http://ru.wikipedia.org/wiki/Microsoft_Windows)

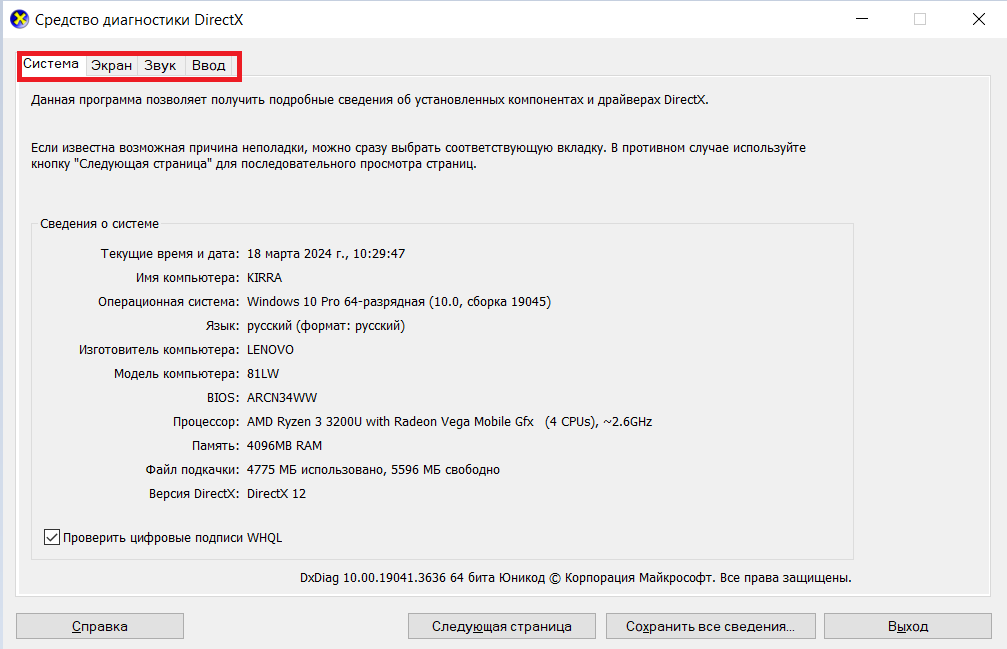
При выполнении Dxdiag.exe открывается окно со следующими вкладками: система, экран, звук, ввод. Все вкладки содержат средства диагностики. (Рис.3)

Рисунок 3 Средство диагностики DirectX

Мониторинг состояния реализует программный модуль: **Msconfig.exe** (в папке

C:\Windows\System32.)

Вкладка содержит средства, позволяющие изменять конфигурацию ОС путем отключения (включения) системных компонентов и программных модулей, оптимизировать работу ОС, а также автоматизировать устранение неполадок при настройке ОС. (Рис.4)

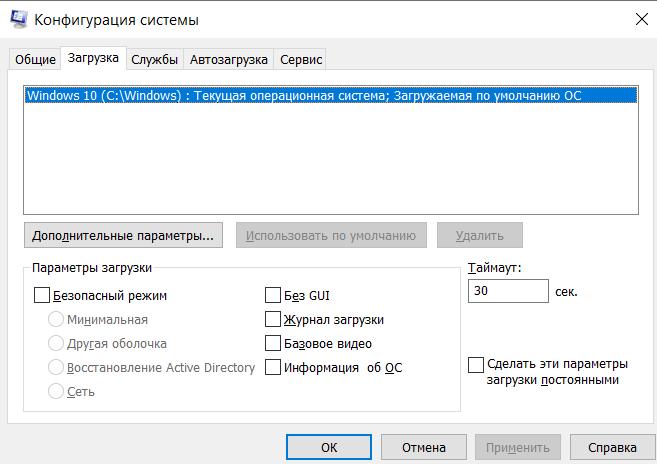


Рисунок 4 Конфигурация системы и изменение конфигурации ОС

# ГЛАВА 2 РЕАЛИЗАЦИЯ ПРАКТИЧЕСКОГО ЗАДАНИЯ

# 2.1 Управление процессами ОС WINDOWS 10

В данной главе разработаем графический инструмент для отслеживания системных процессов, их сортировки и запуска новых системных задач.

Создание графических приложений на C# можно использовать бесплатную среду разработки – Visual Studio, которая облегчает проектирование приложений.

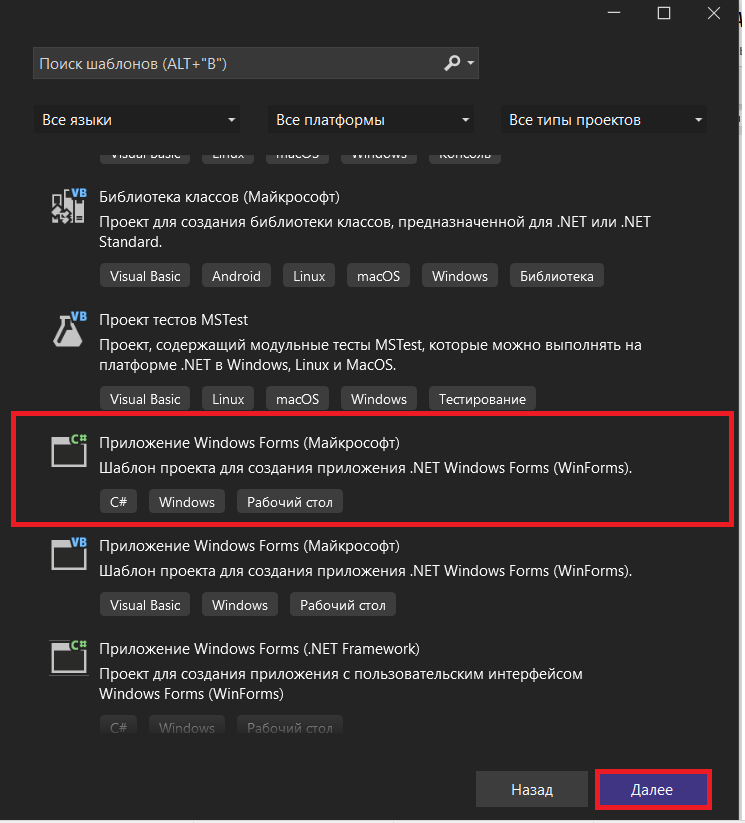
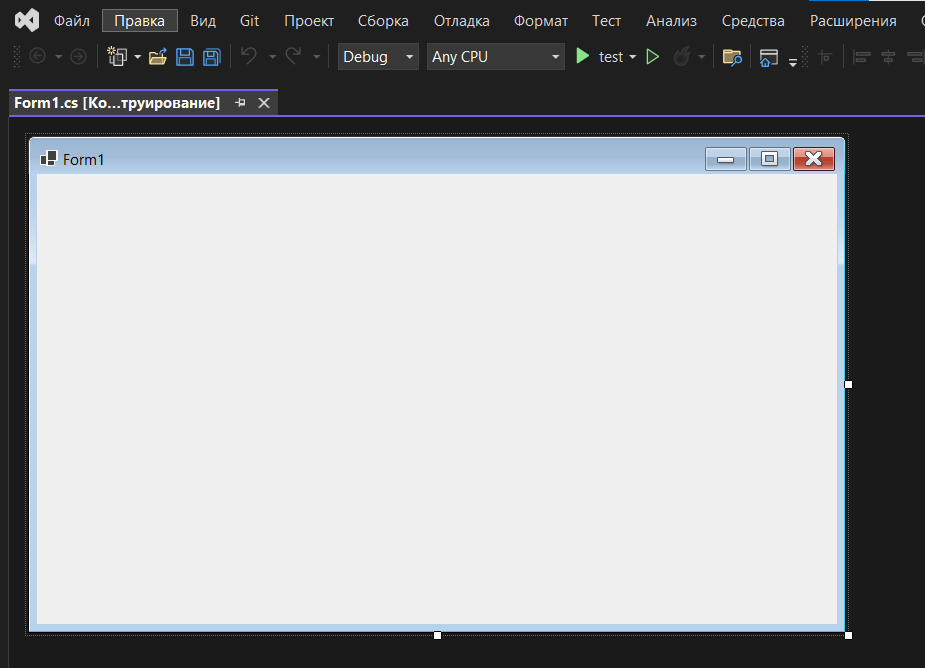
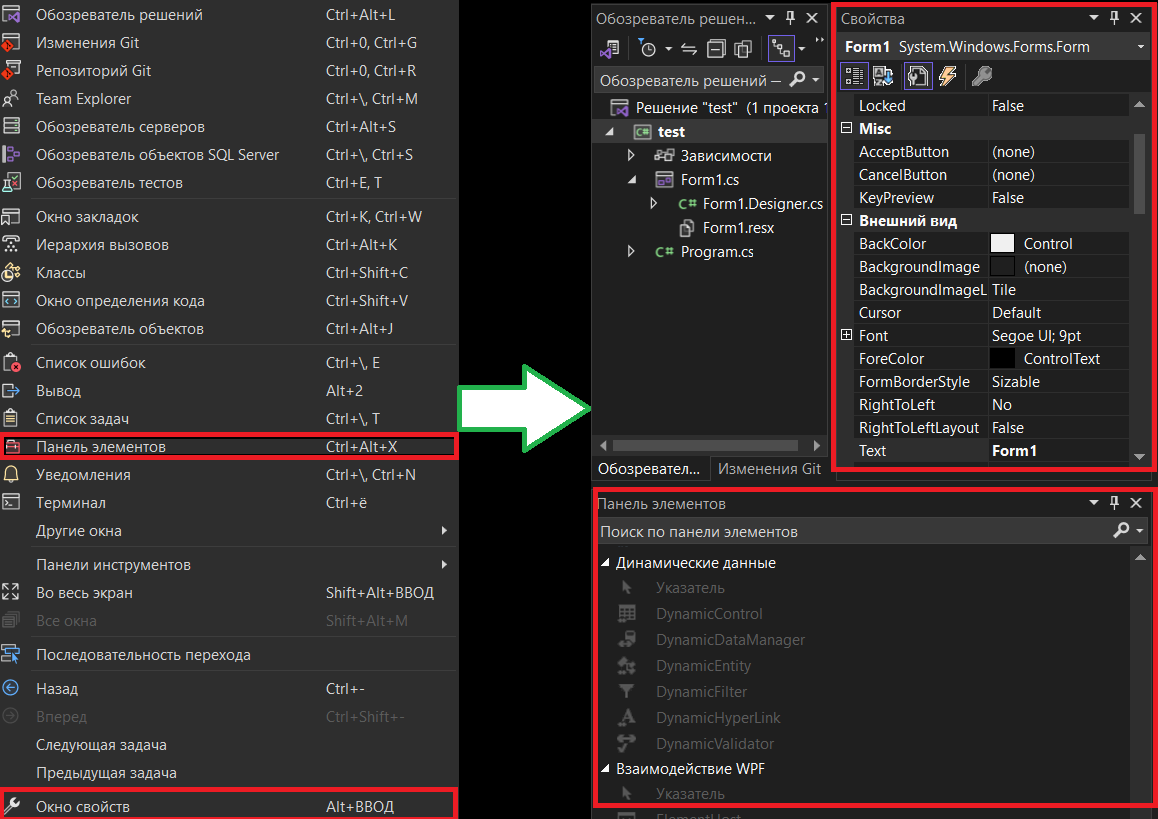
Чтобы разработать инструмент управления процессами операционной системы, будем использовать Windows Forms для наглядной работы. Чтобы добавить в Visual Studio поддержку проектов для Windows Forms и C#, при создании проекта выберем: (Рис.5)

Рисунок 5 Выбор шаблона для проекта

После откроется рабочая форма: (Рис.6)

Рисунок 6 Рабочее пространство для начала работы

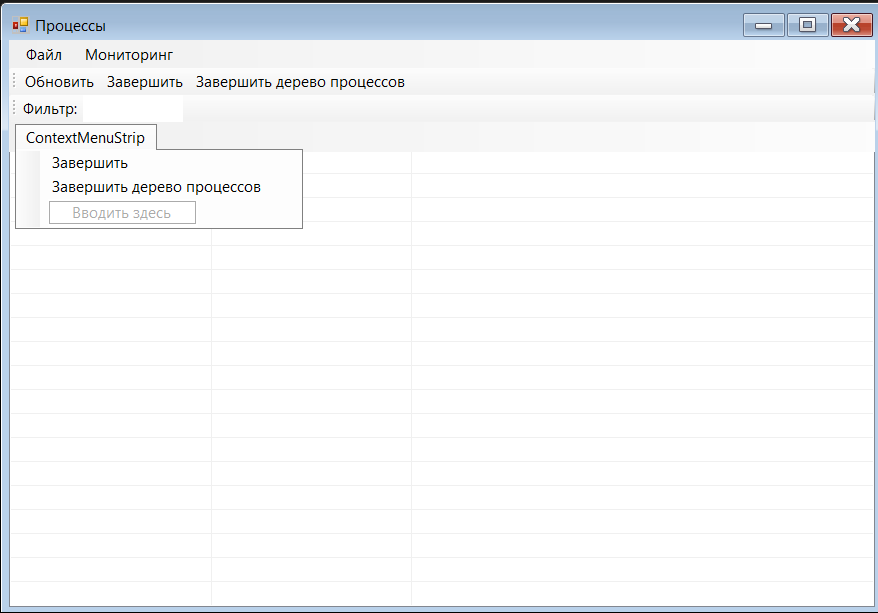
Для более понятного и простого использования конструктора нужно открыть вспомогательные окна «Свойства» и «Панель элементов». Свойства и панель элементов в Windows Forms – это основные инструменты разработки графического пользовательского интерфейса. Свойства позволяют настраивать параметры элементов управления, такие как цвет, размер, текст и другие. Панель элементов представляет собой контейнер, в котором можно размещать различные элементы управления для создания интерактивных форм и приложений. (Рис.7)

Рисунок 7 Рабочее пространство Windows Forms

В панели элементов выберем и добавим на главную форму Form1.cs некоторые формы:

1. toolStrip1 и toolStrip2 – будут выполнять функции кнопок, которые могут быть вложенными и выполнять некоторые действия.
2. contextMenuStrip1 – форма, которая при нажатии будет выполнять функции завершения процесса и завершения всего дерева процессов.
3. Listview – отвечает за основной фон, на котором будут отображаться все процессы.
4. menuStrip1 – рабочая панель для добавление системной задачи и мониторинга процессов.

После расстановки всех форм и настройки их свойств, итоговая работа будет выглядеть примерно так: (Рис.8)

Рисунок 8 Итоговый вид окна управления процессами

Перейдем к работе с кодом. Подключим пространство имен для взаимодействия с системными процессами - [System.Diagnostics,](https://learn.microsoft.com/ru-ru/dotnet/api/system.diagnostics?view=net-8.0) [System.Management](https://learn.microsoft.com/ru-ru/dotnet/api/system.management?view=dotnet-plat-ext-8.0).

Создадим необходимые поля:

Листинг 1

private List<Process> processes = null; //Главный список хранения процессов

private ListViewItemComparer comparer = null; //Сортировка списка

Необходимо создать два метода. Первый метод отвечает за заполнение списка.

fЛистинг 2

private void GetProcesses() //метод заполнения списка и его заполнения

{

processes.Clear(); //очистка списка

processes = Process.GetProcesses().ToList<Process>(); //заполнение списка заново и получаем все системные процессы GetProcesses()}

Во втором методе мы будет создавать колонки и заполнять ListView “контентом”.

Листинг 3

private void RefreshProcessesList() //метод заполнения ListView контента

{

listView1.Items.Clear(); //Очистка

double memSize = 0; //В этой переменной будет при переборе всех процессов храниться память, которую занимает процессор

foreach (Process p in processes)

{ //Перебор всех процессов

memSize = 0;

PerformanceCounter pc = new PerformanceCounter();

pc.CategoryName = "Process";

pc.CounterName = "Working Set - Private";

pc.InstanceName = p.ProcessName;

memSize = (double)pc.NextValue() / (1000 \* 1000);

string[] row = new string[] //Передаем массив колонок

{p.ProcessName.ToString(), Math.Round(memSize,1).ToString()

};

listView1.Items.Add(new ListViewItem(row));

pc.Close();

pc.Dispose();

}

Text = "Запущено процессов: " + processes.Count.ToString();

Создадим метод для завершения дерева процессов.

Листинг 4

private void KillProcessesAndChildren(int pid) //Завершение дерева процесса

{

if (pid == 0) //Проверяем id процесса

return;

ManagementObjectSearcher searcher = new ManagementObjectSearcher( //С помощью System.Management рекурсивно по id заверашаем нужный процесс

"Select + From Win32\_Process Where ParentProcessID=" + pid);

ManagementObjectCollection objectsCollection = searcher.Get();

foreach (ManagementObject obj in objectsCollection)

{

KillProcessesAndChildren(Convert.ToInt32(obj["ProcessId"]));

}

try {

Process p = Process.GetProcessById(pid);

p.Kill();

p.WaitForExit();

}

catch (ArgumentException) { }

}

Теперь перейдем к вызову наших методов в основной форме программы Form1\_Load.- заполним список процессами.

Листинг 5

private void Form1\_Load(object sender, EventArgs e)

{

processes = new List<Process>();

GetProcesses(); //Заполняет список с процессами

RefreshProcessesList(); //Заполнит ListVie

comparer = new ListViewItemComparer();

comparer.ColumnIndex = 0;

Вторым шагом будем обрабатывать кнопку обновления процессов.

Листинг 6

private void toolStripButton1\_Click(object sender, EventArgs e) //Кнопка обновления процессов

{

GetProcesses(); RefreshProcessesList()}

Последним шагом будем запускать новую системную задачу помощью оператора обработки исключений.

Листинг 7

private void запуститьЗадачуToolStripMenuItem\_Click(object sender, EventArgs e)

{

string path = Interaction.InputBox("Введите имя программы", "Запуск новой задачи");

try

{

Process.Start(path);

}

catch (Exception) { }}

Функция вызова следующей работы, которая описана ниже.

Листинг 8

private void мониторингToolStripMenuItem\_Click(object sender, EventArgs e)

{

f2 = new Form2();

f2.Show();

}}

В итоге получим данный инструмент управления процессами (Рис.9)

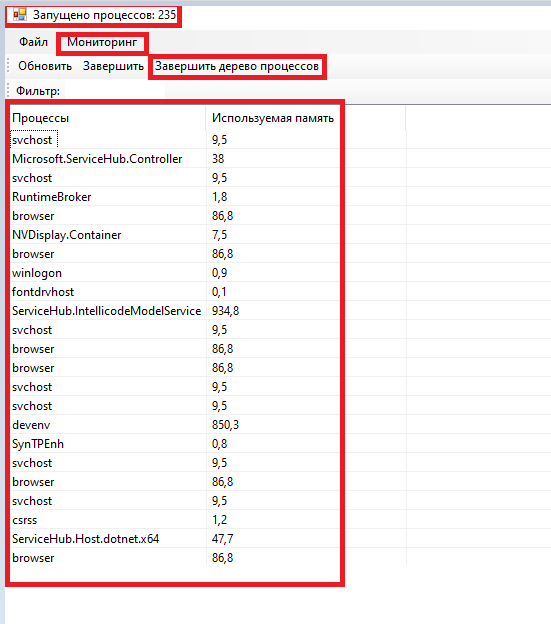


Рисунок 9 Разработка инструмент управления процессами

В итоге получаем готовый ресурс для регулировки и отслеживания процессов компьютера.

# 2.2 Мониторинг системных ресурсов

В данном блоке создадим ресурс, который будет отображать графики загруженности процессора и оперативной памяти компьютера.

Будем использовать пространство имен MetroFramework.Forms.

MetroFramework.Forms – это библиотека .NET, которая предоставляет набор элементов управления пользовательского интерфейса для создания приложений с современным и привлекательным дизайном. Она позволяет разработчикам быстро и легко создавать приложения с пользовательским интерфейсом, соответствующим рекомендациям Microsoft по дизайну Windows.

Чтобы использовать данную библиотеку путем using MetroFramework.Forms нужно добавить ссылку на сборку с расширением .dll в проект.

Первым действием это создание проекта Windows Forms [(Рис.5)](#_2.1_Управление_процессами) и Создание рабочей области [(Рис.6)](#_2.1_Управление_процессами).

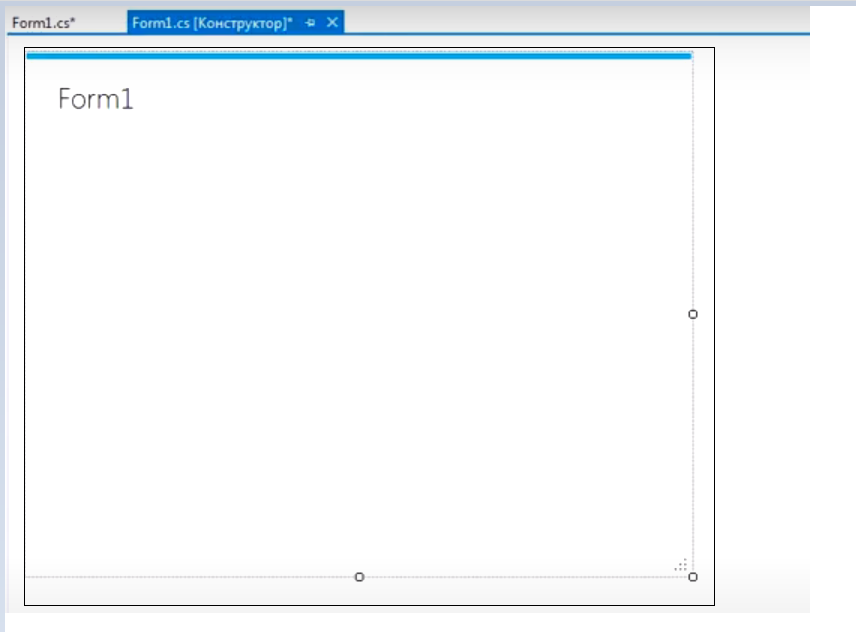
Подключим нашу библиотеку к проекту с мониторингом, нужно нажать на форму правой кнопкой мыши и выбрать перейти к коду, после чего подключить пространство имен и получим: (Рис.10)

Рисунок 10 Работа с библиотекой MetroForms

Так же для данного приложения нужно использовать свойства и панель элементов. С помощью панели элементов добавим следующие вещи для удобного отображения функционала:

1. Label – текстовые поля, чтобы подписывать графики и ProgressBar.
2. metroProgressBar – будут отображать полосы загруженности ЦП и ОЗУ.
3. performanceCPU и performanceRAM – отображение производительности CPU и RAM.
4. Chart – является элементом, который отвечает за график отображения процессов.

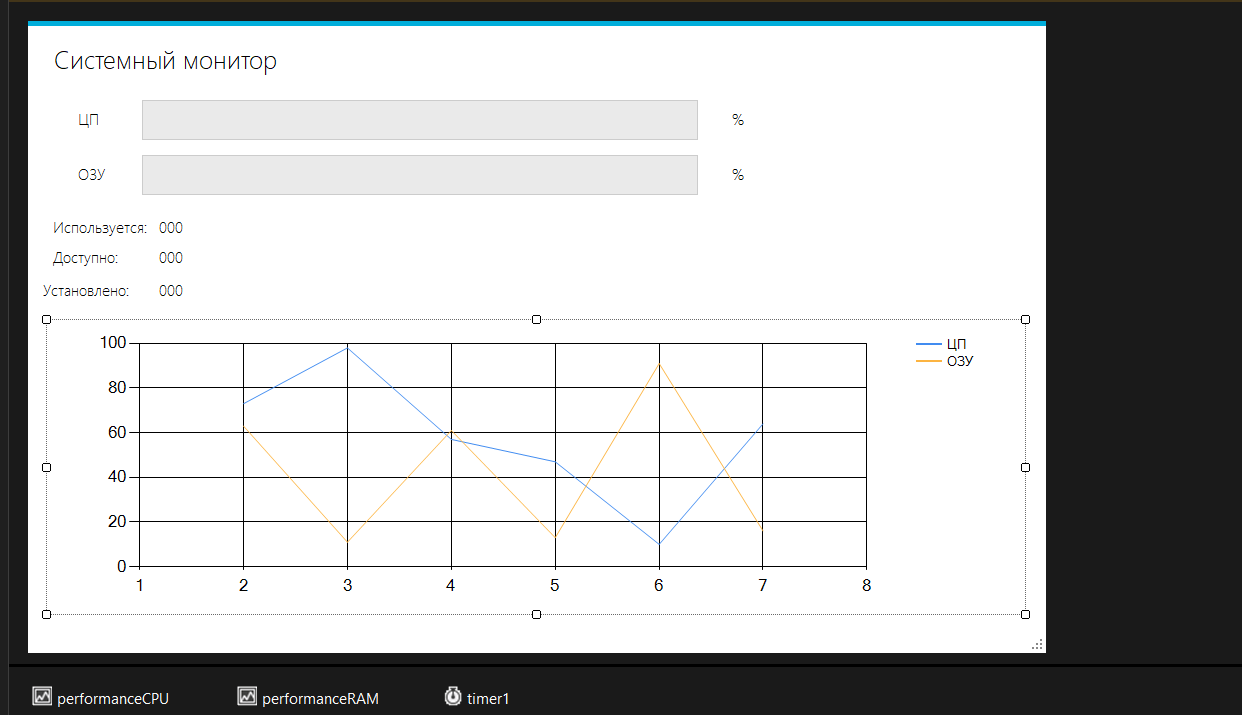
После добавления элементов и правильно заданных полей Label получим: (Рис.11)

Рисунок 11 Системный мониторинг загрузки CPU и RAM

Перейдем к коду. Для начала кликнем два раза по заголовку и у нас создастся событие Form2\_Load. В нем будем получать общий объем памяти, будем инициализировать некоторые свойства и зададим текст с помощью Label.

Создадим 2 поля через private float – cpu и ram, а также через символьный тип данных ulong поле installedMemory.

Листинг 9

private float cpu;

private float ram;

private ulong installedMemory;

Объявим пространство имен System.Runtime.InteropServices - позволяет различным программам иметь общий доступ к памяти для более эффективного распределения ресурсов, что способствует повышению быстродействия компьютера.

Реализуем класс и метод с помощью которых будем получать общий объем памяти. Класс нужен для того, чтобы получать и возвращать получаемые значения метода.

Листинг 10

private class MEMORYSTATUSEX

{

public uint dwLength;

public uint dwMemoryLength;

public ulong ullTotalPhys;

public ulong ullAvailPhys;

public ulong ullTotalPageFile;

public ulong ullAvailPageFile;

public ulong ullTotalVirtual;

public ulong ullAvailVirtual;

public ulong ullAvailExtendedVirtual;

public MEMORYSTATUSEX()

{

this.dwLength = (uint)Marshal.SizeOf(typeof(MEMORYSTATUSEX));

}}

Функция [MEMORYSTATUSEX](https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/ns-sysinfoapi-memorystatusex) хранит информацию о текущем состоянии как физической, так и виртуальной памяти, включая расширенную память.

Опишем сигнатуру метода kernel32.dll - динамически подключаемая библиотека, являющаяся ядром всех версий ОС Microsoft Windows. Она предоставляет приложениям многие базовые API Win32, такие как управление памятью, операции ввода-вывода, создание процессов и потоков и функции синхронизации. Метод должен принимать один параметр по ссылке - MEMORYSTATUSEX

Листинг 11

[DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]

static extern bool [GlobalMemoryStatusEx](https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-globalmemorystatusex)([In, Out] MEMORYSTATUSEX lpBuffer);

Теперь вернемся к событию Form2\_Load проинициализируем поле installedMemory, также всю информацию будет переводить в гигабайты и с помощью таймера обновлять информацию в metroProgressBar за 1 секунду или 1000 миллисекунд.

Листинг 12

private void Form2\_Load(object sender, EventArgs e)

{

MEMORYSTATUSEX mEMORYSTATUSEX = new MEMORYSTATUSEX();

if(GlobalMemoryStatusEx(mEMORYSTATUSEX))

{

installedMemory = mEMORYSTATUSEX.ullTotalPhys;

}

metroLabel10.Text = Convert.ToString(installedMemory / 1000000000) + "Гб";

timer1.Interval = 1000;

timer1.Start();}

Последним действием создадим обработчик события timer1\_tick. (Рис.12)

Листинг 13

private void timer1\_Tick(object sender, EventArgs e)

{

cpu = performanceCPU.NextValue(); //Считывает данные с perfomancecouner

ram = performanceRAM.NextValue(); //Считывает данные с perfomancecouner

metroProgressBar1.Value = (int)cpu; //Присвоим значения в metroProgressBar1

metroProgressBar2.Value = (int)ram; //Присвоим значения в metroProgressBar2

metroLabel2.Text = Convert.ToString(Math.Round(cpu, 1)) + "%"; //Конвертируем в текст и переводим с точностью до 1 знака после запятой

metroLabel3.Text = Convert.ToString(Math.Round(ram, 1)) + "%";

metroLabel6.Text = Convert.ToString(Math.Round((ram / 100 \* installedMemory) / 1000000000, 1)) + " Гб";

metroLabel8.Text = Convert.ToString(Math.Round((installedMemory - ram / 100 \* installedMemory) / 1000000000, 1)) +" Гб";

chart1.Series["ЦП"].Points.AddY(cpu); //Обращаемся к линиям графика как к ключу словаря через ЦП

chart1.Series["ОЗУ"].Points.AddY(ram);} //Обращаемся к линиям графика как к ключу словаря через ОЗУ

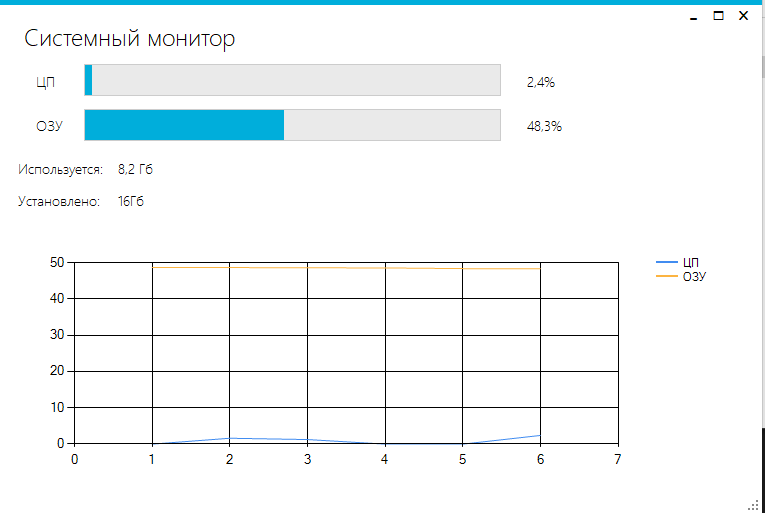


Рисунок 12 Итоговое отображение процессов

В итоге был разработан процесс отслеживания загруженности системных параметров компьютера.

# 2.3 Тестирование программы

Разработав приложение, приступим к тестированию. В первом пункте разбирали создание собственного инструмента управления процессами, рассмотрим все возможности, которые удалось реализовать: 1) количество запущенных процессов, 2) запуск системных задач, 3) обновление и завершение процессов, 4) фильтрация по названию, 5) сортировка по имени процесса и по количеству занятой памяти. (Рис.13)

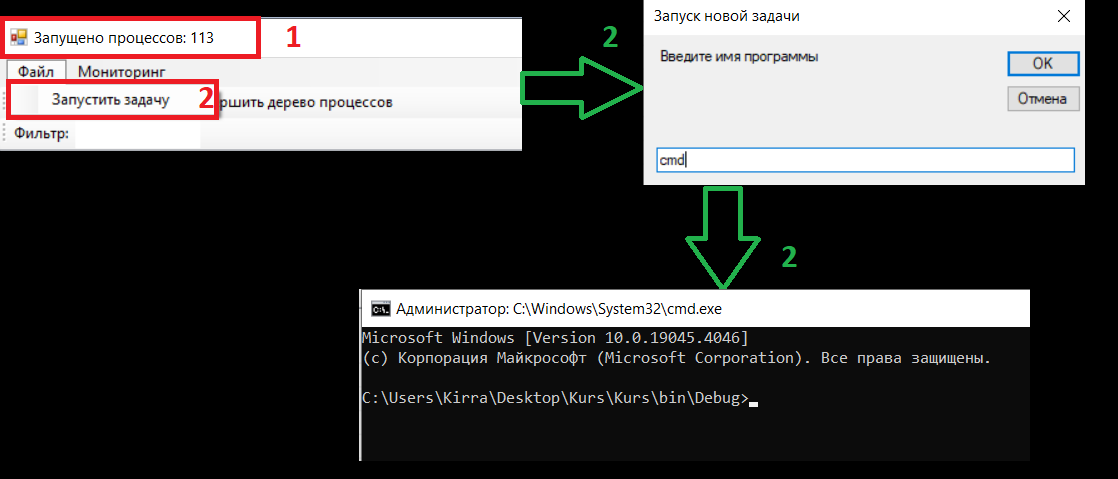


Рисунок 13 Запущенные процессы и системные задачи

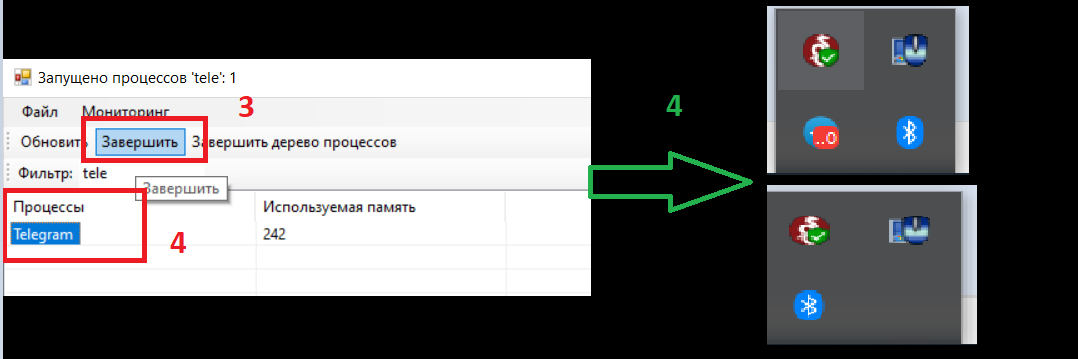
С помощью фильтрации по названию можно быстро искать запущенный процесс и завершать его: (Рис.14)

Рисунок 14 Завершение и фильтрация процессов

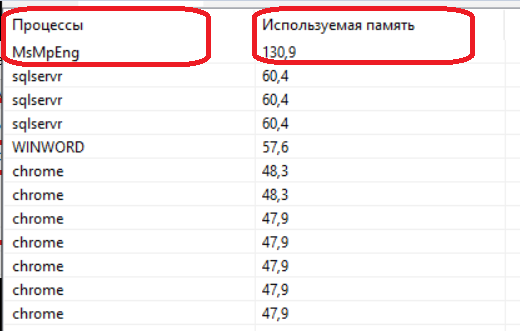
Для сортировки процессов по названию и памяти достаточно нажать на «процессы» и «используемая память» (Рис.15)

Рисунок 15 Сортировка процессов

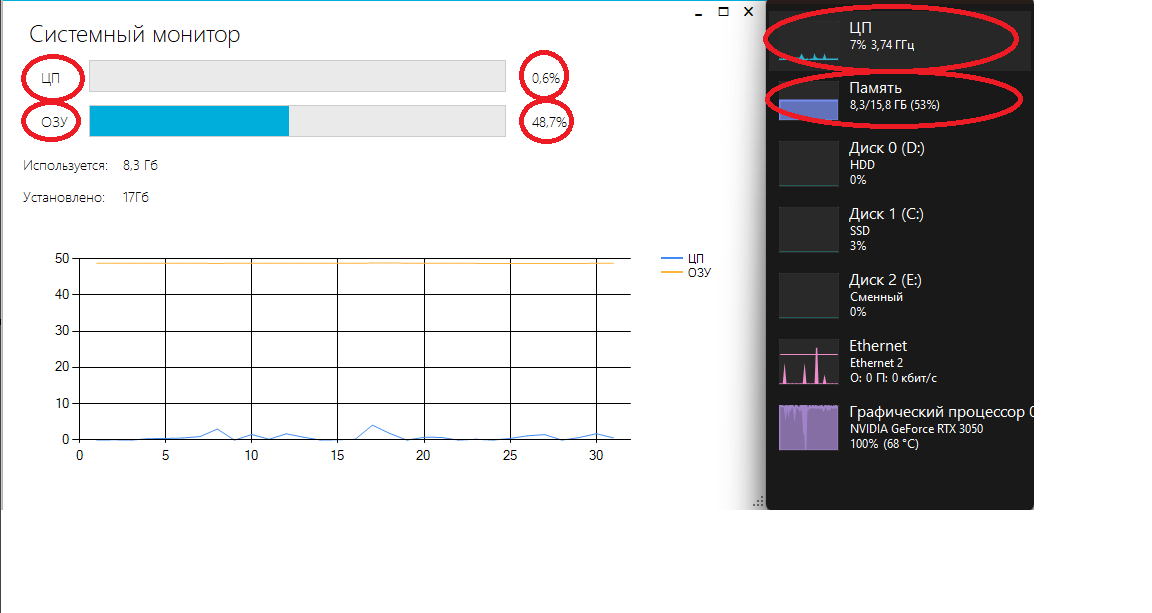
Второе встроенное приложение отображает процентные данные загруженности процессора и оперативной памяти, а также реализован наглядный график: (Рис.16)

Рисунок 16 Реализация мониторинга системных параметров

В ходе выполнения задания в данной главе был разработан инструмент для управления системными процессами. Данный инструмент позволяет пользователю контролировать использование системных ресурсов, определять наиболее ресурсоемкие процессы и оптимизировать их работу.

Разработанный инструмент включает в себя модуль мониторинга, который предоставляет информацию о загрузке процессора, использовании оперативной памяти. Это позволяет пользователю оперативно реагировать на изменения в работе системы и принимать меры для предотвращения перегрузки ресурсов.

Также инструмент содержит модуль управления процессами, который позволяет пользователю останавливать, запускать и перезапускать процессы. Это дает возможность оптимизировать работу системы, снижая нагрузку на процессор и оперативную память.

В целом, разработанный инструмент является полезным инструментом для управления системными ресурсами персонального компьютера и может быть использован как профессионалами в области информационных технологий, так и обычными пользователями для оптимизации работы своей системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Леонтьев, В. Windows 10. Новейший самоучитель для компьютеров / В. Леонтьев. - 4-е изд. - Москва: Эксмо, 2022. - 384 c. - Текст: непосредственный.
2. Шитов, В. Windows 10: самый простой и понятный самоучитель / В. Шитов. - 1-е изд. - Москва: Эксмо, 2023. - 464 c. - Текст: непосредственный.
3. Иванько, А. Операционные системы. Практикум / А. Иванько, М. Иванько, А. Курносова. - 1-е изд. - Санкт-Петербург: Лань, 2022. - 132 c. - Текст: непосредственный.
4. Арпачи-Дюрсо, Андреа Операционные системы. Три простых элемента / Андреа Арпачи-Дюрсо. - 3-е изд. - Москва: ДМК Пресс, 2021. - 730 c. - Текст: непосредственный.
5. Гарькушев, А. Основы обеспечения безопасности операционных систем / А. Гарькушев, И. Карпова. - 1-е изд. - Вологда: Инфра-Инженерия, 2024. - 124 c. - Текст: непосредственный.
6. Матвеев, М. Windows 10? Легко! Настройки и лайфхаки. Пошаговые инструкции по применению / М. Матвеев. - 4-е изд. - Санкт-Петербург: Наука и техника, 2020. - 288 c. - Текст: непосредственный.
7. Хорев, П. Объектно-ориентированное программирование с примерами на C#. Учебное пособие / П. Хорев. - 4-е изд. - Москва: Форум, 2020. - 200 c. - Текст: непосредственный.
8. Евдокимов, П. C# на примерах. Практика, практика и только практика / П. Евдокимов. - 2-е изд. - Санкт-Петербург: Наука и техника, 2022. - 320 c. - Текст: непосредственный.
9. Тюкачев, Н. А. C#. Основы программирования. Учебное пособие / Н. А. Тюкачев. - 3-е изд. - Санкт-Петербург: Лань, 2021. - 272 c. - Текст: непосредственный.
10. Албахари, Джозеф C# 6.0. Карманный справочник / Джозеф Албахари. - 4-е изд. - Москва: Лань, 2019. - 213 c. - Текст: непосредственный.
11. Шакин, В. Базовые средства программирования на Visual Basic в среде Visual Studio. NET практикум. Учебное пособие. / В. Шакин. - 4-е изд. - Москва: Инфра-М, 2019. - 287 c. - Текст: непосредственный.
12. Андриевская, Н. Основы визуального программирования в среде VISUAL STUDIO на базе C# / Н. Андриевская, Т. Мартыненко, В. Турупалов. - 2-е изд. - Вологда: Инфра-Инженерия, 2023. - 232 c. - Текст: непосредственный.
13. Шилдт, Герберт C++ для начинающих / Герберт Шилдт. - 1-е изд. - Санкт-Петербург: Питер, 2024. - 608 c. - Текст: непосредственный.
14. Руководство по программированию в Windows Forms. - Текст: электронный // METANIT.COM: [сайт]. - URL: https://metanit.com/sharp/windowsforms/ (дата обращения: 31.03.2024).
15. C# Windows Учебное пособие по применению форм с примером. - Текст: электронный // GURU99: [сайт]. - URL: https://www.guru99.com/ru/c-sharp-windows-forms-application.html (дата обращения: 31.03.2024).
16. Полное руководство по языку программирования С# 12 и платформе .NET 8. - Текст: электронный // METANIT.COM: [сайт]. - URL: https://metanit.com/sharp/tutorial/ (дата обращения: 31.03.2024).
17. Процессы и потоки в Windows / - Текст: электронный // Win32 и не только: [сайт]. - URL: https://av239.wordpress.com (дата обращения: 23.03.2024).
18. Основные системные процессы Windows 10 и их назначение / - Текст: электронный // white-window: [сайт]. - URL: https://www.white-windows.ru/osnovnye-sistemnye-protsessy-windows-10-i-ih-naznachenie/ (дата обращения: 23.03.2024).
19. Используем монитор ресурсов Windows. - Текст: электронный // remontka.pro: [сайт]. - URL: https://remontka.pro/windows-resource-monitor/ (дата обращения: 23.03.2024).
20. Как работать с диспетчером задач в windows. - Текст: электронный // DNS: [сайт]. - URL: https://goo.su/97CIeG (дата обращения: 23.03.2024).

# ПРИЛОЖЕНИЕ

К пункту 2.1:

using System.Windows.Forms;

using System.Diagnostics; //!!!

using System.Management; //!!!

using Microsoft.VisualBasic; //!!!

namespace Kurs

{public partial class Form1 : Form

{

private List<Process> processes = null; //Главный список хранения процессов

private ListViewItemComparer comparer = null;

public Form1()

{

InitializeComponent();}

Form2 f2;

private void GetProcesses() //метод заполнения списка и его заполнения{

processes.Clear(); //очистка списка

processes = Process.GetProcesses().ToList<Process>(); //заполнение списка заново и получаем все системные процессы GetProcesses()

}

private void RefreshProcessesList() //метод заполнения ListView контента

{listView1.Items.Clear(); //Очистка

double memSize = 0; //В этой переменной будет при переборе всех процессов храниться память, которую занимает процессор

foreach (Process p in processes)

{ //Перебор всех процессов

memSize = 0;

PerformanceCounter pc = new PerformanceCounter();

pc.CategoryName = "Process";

pc.CounterName = "Working Set - Private";

pc.InstanceName = p.ProcessName;

memSize = (double)pc.NextValue() / (1000 \* 1000);

string[] row = new string[]

{p.ProcessName.ToString(), Math.Round(memSize,1).ToString()

};listView1.Items.Add(new ListViewItem(row));

pc.Close();

pc.Dispose();}

Text = "Запущено процессов: " + processes.Count.ToString();}

private void RefreshProcessesList(List<Process> processes, string keyword)

{try{

listView1.Items.Clear();

double memSize = 0;

foreach (Process p in processes){

if (p != null){

memSize = 0;

PerformanceCounter pc = new PerformanceCounter();

pc.CategoryName = "Process";

pc.CounterName = "Working Set - Private";

pc.InstanceName = p.ProcessName;

memSize = (double)pc.NextValue() / (1000 \* 1000);

string[] row = new string[]{

p.ProcessName.ToString(), Math.Round(memSize,1).ToString()};

listView1.Items.Add(new ListViewItem(row));

pc.Close();

pc.Dispose();}}

Text = $"Запущено процессов '{keyword}': " + processes.Count.ToString();}

catch (Exception) { }}

private void KillProcesses(Process processes){

processes.Kill();

processes.WaitForExit();

}

private void KillProcessesAndChildren(int pid)

{if (pid == 0)

return;

ManagementObjectSearcher searcher = new ManagementObjectSearcher(

"Select + From Win32\_Process Where ParentProcessID=" + pid);

ManagementObjectCollection objectsCollection = searcher.Get();

foreach (ManagementObject obj in objectsCollection){

KillProcessesAndChildren(Convert.ToInt32(obj["ProcessId"]));}

try {

Process p = Process.GetProcessById(pid);

p.Kill();

p.WaitForExit();}

catch (ArgumentException) { }}

private int GetParentProccesId(Process p){

int parentId = 0;

try{

private void Form1\_Load(object sender, EventArgs e){

processes = new List<Process>();

GetProcesses();

RefreshProcessesList();

comparer = new ListViewItemComparer();

comparer.ColumnIndex = 0;}

private void toolStripButton1\_Click(object sender, EventArgs e){

GetProcesses(); RefreshProcessesList();}

private void toolStripButton2\_Click(object sender, EventArgs e){

try{

if (listView1.SelectedItems[0] != null){

Process processToKill = processes.Where((x) => x.ProcessName ==

listView1.SelectedItems[0].SubItems[0].Text).ToList()[0];

KillProcesses(processToKill);

GetProcesses(); RefreshProcessesList(); }}

catch (Exception) { }

К пункту 2.2:

using System.Windows.Forms;

using MetroFramework.Forms;

using System.Runtime.InteropServices;

namespace Kurs

{

public partial class Form2 : MetroForm{

private float cpu;

private float ram;

private ulong installedMemory;

public Form2(){

InitializeComponent();}

private void Form2\_Load(object sender, EventArgs e){

MEMORYSTATUSEX mEMORYSTATUSEX = new MEMORYSTATUSEX();

if(GlobalMemoryStatusEx(mEMORYSTATUSEX)){

installedMemory = mEMORYSTATUSEX.ullTotalPhys;}

metroLabel10.Text = Convert.ToString(installedMemory / 1000000000) + "Гб";

timer1.Interval = 1000;

timer1.Start();}

\[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]

private class MEMORYSTATUSEX

{public MEMORYSTATUSEX(){

this.dwLength = (uint)Marshal.SizeOf(typeof(MEMORYSTATUSEX));}}

[return: MarshalAs(UnmanagedType.Bool)]

[DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]

static extern bool GlobalMemoryStatusEx([In, Out] MEMORYSTATUSEX lpBuffer);

private void timer1\_Tick(object sender, EventArgs e)

{cpu = performanceCPU.NextValue(); //Считывает данные с perfomancecouner

ram = performanceRAM.NextValue(); //Считывает данные с perfomancecouner

metroProgressBar1.Value = (int)cpu; //Присвоим значения в metroProgressBar1

metroProgressBar2.Value = (int)ram; //Присвоим значения в metroProgressBar2

metroLabel2.Text = Convert.ToString(Math.Round(cpu, 1)) + "%"; //Конвертируем в текст и переводим с точностью до 1 знака после запятой

metroLabel3.Text = Convert.ToString(Math.Round(ram, 1)) + "%";

metroLabel6.Text = Convert.ToString(Math.Round((ram / 100 \* installedMemory) / 1000000000, 1)) + " Гб";

// metroLabel8.Text = Convert.ToString(Math.Round((installedMemory - ram / 100 \* installedMemory) / 1000000000, 1)) +" Гб";

chart1.Series["ЦП"].Points.AddY(cpu); //Обращаемся к линиям графика как к ключу словаря через ЦП

chart1.Series["ОЗУ"].Points.AddY(ram); //Обращаемся к линиям графика как к ключу словаря через ОЗУ