

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ

_____ Б. В. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
на тему

«Утилита сбора информации о системе»

БГУИР КП 1-40 02 01 425 ПЗ

Студент:

Чеботарёв В.С.

Руководитель:

Ассистент кафедры ЭВМ
Басак Д. В.

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
1 ОБЗОР ЛИТЕРАТУРЫ.....	7
2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ.....	20
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ.....	21
4 ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ.....	26
5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	27
6 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ.....	32
ЗАКЛЮЧЕНИЕ.....	34
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	35
ПРИЛОЖЕНИЕ А.....	36
ПРИЛОЖЕНИЕ Б.....	37
ПРИЛОЖЕНИЕ В.....	38

ВВЕДЕНИЕ

Семимильными шагами идёт прогресс по планете Земля. Сегодня можно с абсолютной уверенностью сказать, что персональные компьютеры и ноутбуки, буквально тридцать лет назад представлявшие из себя огромного размера машины, которые были всего у нескольких тысяч людей со всего мира, стали самой настоящей обыденностью. Компьютеры внедряются во все отрасли человеческой деятельности. Они помогают автоматизировать процессы, на выполнение которых требуется много времени. У каждого в квартире присутствует как минимум по одному персональному компьютеру или ноутбуку. И можно с уверенностью предположить, что большинство продвинутых пользователей компьютера не единожды задавались вопросом об его системной информации для построения корректной работы. Эта информация позволяет узнать не только сведения об аппаратной части компьютера, но и программные составляющие, такие как, например, информация обо всех процессах компьютера или сетевых интерфейсах.

Данная программа предоставляет пользователю выбор следующих действий:

- увидеть полный список всех процессов на используемом ПК;
- увидеть аппаратные адреса устройства;
- увидеть информацию об установленном на компьютере центральном процессоре;
- увидеть информацию об оперативной памяти компьютера;
- информацию об операционной системе;
- имя пользователя компьютера.

В плане выбора языка программирования, на котором будет писаться приложение, сомнений не было. С++ является одним из лучших вариантов. Открыв любой тест производительности языков программирования в таблице лидеров, вы обязательно увидите его. В противовес этому часто ставится скорость написания кода, которая, например, у интерпретируемых языков на порядок выше. В этом есть доля истины — С#, Java и конечно же Python даже визуально занимают меньше места, с их помощью можно создавать сложные программы, затратив минимум времени. Однако, что лучше для конечного пользователя: время разработки приложения или его медленная работа? Ответ очевиден.

Еще одной причиной, стала универсальность данного языка, компиляторы С++ есть на каждой операционной системе, большинство программ легко переносится с платформы на платформу, со средой разработки и библиотеками точно не возникнет проблем. Язык имеет богатую классическую библиотеку, которая включает в себя разные контейнеры и алгоритмы, регулярные выражение, разные фреймворки и библиотеки которые позволяют создавать графическую часть приложения. Еще одним удобством стало то, что язык С++ позволяют писать системные утилиты без больших сложностей.

Исходя из этого можно с уверенностью сказать, что данный язык достаточно удобен и хорошо функционален для написания выбранной курсовой работы.

В плане выбора операционной системы, на которую будет писаться программа сомнений не было. Linux – это операционная система с открытым исходным кодом. Она достаточно открыта для пользователя и в случае потребностей можно легко изменить нужные для вас настройки. Так же на этой операционной системе нет такого большого количества качественных приложений, которые присутствуют на Windows. Изучив всю информацию по данному вопросу, я решил, что данную курсовую работу следует писать именно под операционную систему Linux.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор предметной области

1.1.1 Компьютер

Компьютер — это не одно устройство, а множество взаимосвязанных различных устройств (рисунок 1.1).



Рисунок 1.1 – Компьютер

Наиболее важные функциональные элементы компьютера:

Главная плата, процессор, оперативная память, жесткий диск, SD и DVD приводы, видеокарта, звуковая карта, сетевая карта, порты ввода/вывода (разъемы), блок питания.

Все данные обрабатываются материнской платой или системной платой во время работы компьютера. Материнская плата представляет собой сложную многослойную печатную плату, к которой подключены все остальные компоненты компьютера (рисунок 1.2).

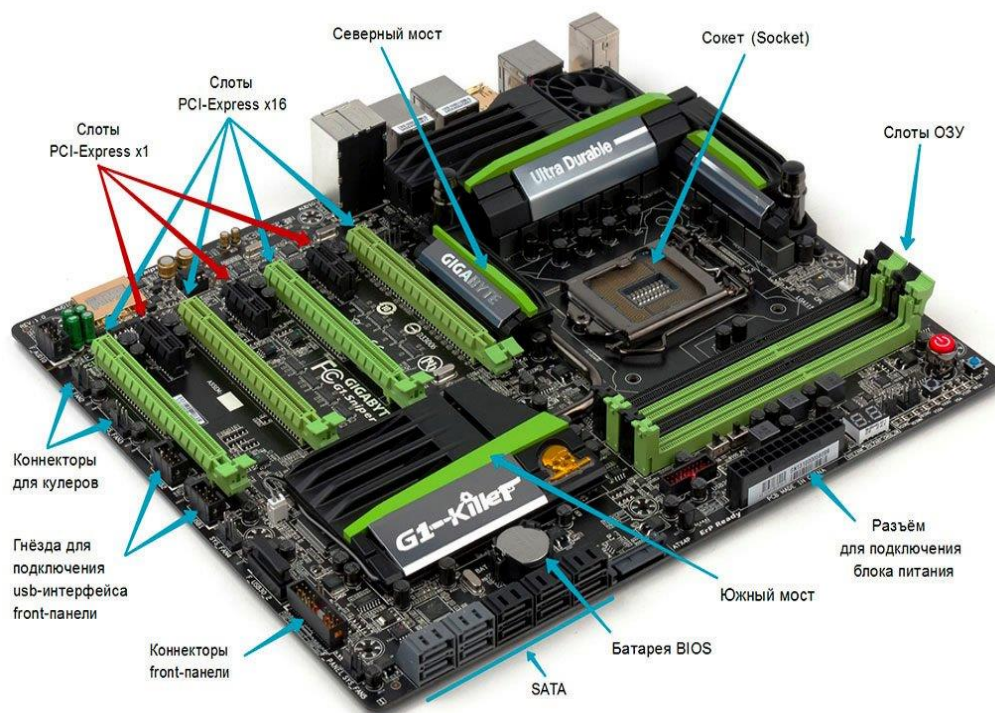


Рисунок 1.2 – Материнская плата

Скорость работы компьютера определяется главным процессором. Все пользовательские документы — файлы, папки, фотографии, изображения, музыка, фильмы и программы — хранятся на встроенном жестком диске компьютера. Компьютерная оперативная память — скорость работы компьютера зависит от того, насколько быстро он открывает приложения, реагирует на действия и выполняет задачи.



Рисунок 1.3 – Процессор

В спецификации компьютера обычно указывается тип и частота процессора, и объем оперативной памяти. Эти характеристики компьютера являются наиболее важными, так как они определяют скорость работы компьютера.

Процессор является центральным процессором компьютера, который выполняет необходимую работу вычислительного программного обеспечения.

ОЗУ является внутренней памятью и отличается от внешней памяти, например, жесткого диска или компакт-диска. Внешняя память сохраняет информацию даже после выключения компьютера (рисунок 1.4).

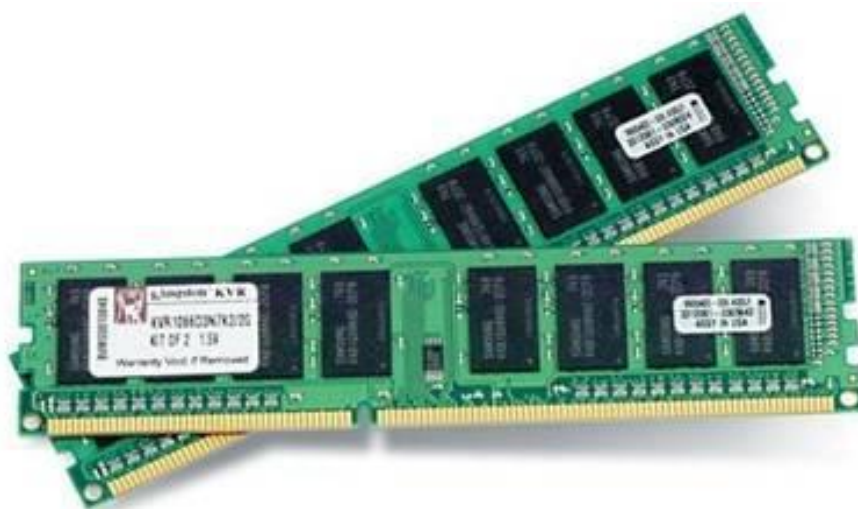


Рисунок 1.4 – ОЗУ

Жесткий диск или жесткий диск является основным устройством хранения данных на вашем компьютере. Большинство компьютеров имеют встроенный жесткий диск, на котором устанавливаются приложения, создаются файлы и хранятся различные данные (рисунок 1.5).



Рисунок 1.5 – Жёсткий диск

1.2 Анализ аналогов программного средства

Часто пользователи задаются вопросом — какой же сервис по просмотру информации выбрать? На самом деле ответить на этот вопрос так просто нельзя, кому-то более нравится интерфейс в одном приложении, другой готов купить подписку чтоб пользоваться приложением без рекламы, третьему большое количество информации о системе в приложении. Сейчас уже появилось достаточное количество приложений по просмотру информации о системе и каждый найдёт для себя то которое подойдёт именно ему. В данном разделе мы рассмотрим возможности утилит для диагностики оборудования компьютера. Они позволят узнать конфигурацию системы и характеристики установленных компонентов, а также в реальном времени проследить за важными показателями, например, загруженностью процессора и задействованной оперативной памятью. Их можно применять в процессе поиска оптимальных драйверов, ведь далеко не каждый пользователь знает обозначение своего оборудования. Все представленные программы можно разделить на две категории — узкоспециализированные утилиты для анализа точных параметров конкретного устройства и приложения широкого спектра, позволяющие продиагностировать не только основные компоненты ПК, но и подключенное оборудование вроде принтеров и контейнеров информации. Разберём наиболее популярные из них.

1.2.1 Sпessу

Sпessу – бесплатная для некоммерческого использования проприетарная утилита, которая предоставляет пользователям мощный и простой в использовании инструмент для отображения детальной системной информации, а также о каждом аппаратном обеспечении персональных компьютеров. Утилита была создана британской частной фирмой Piriform и написана на C++. Интерфейс и частичный функционал изображены на рисунках 1.6 и 1.7.

Программа обеспечивает:

- просмотр параметров центрального процессора, материнской платы, модулей оперативной памяти, HDD и SSD дисков, оптических дисководов, звуковой, графической и сетевой карт;
- определение рабочей температуры дисков, плат и процессора;
- отображение сведений о сетевых и периферийных устройствах;
- поддержку справочной системы;
- сохранение отчета в текстовом файле;
- распространяется бесплатно;
- отсутствует реклама.

Недостатки приложения:

- ограниченный функционал по сравнению с платными аналогами;
- редкая поддержка.

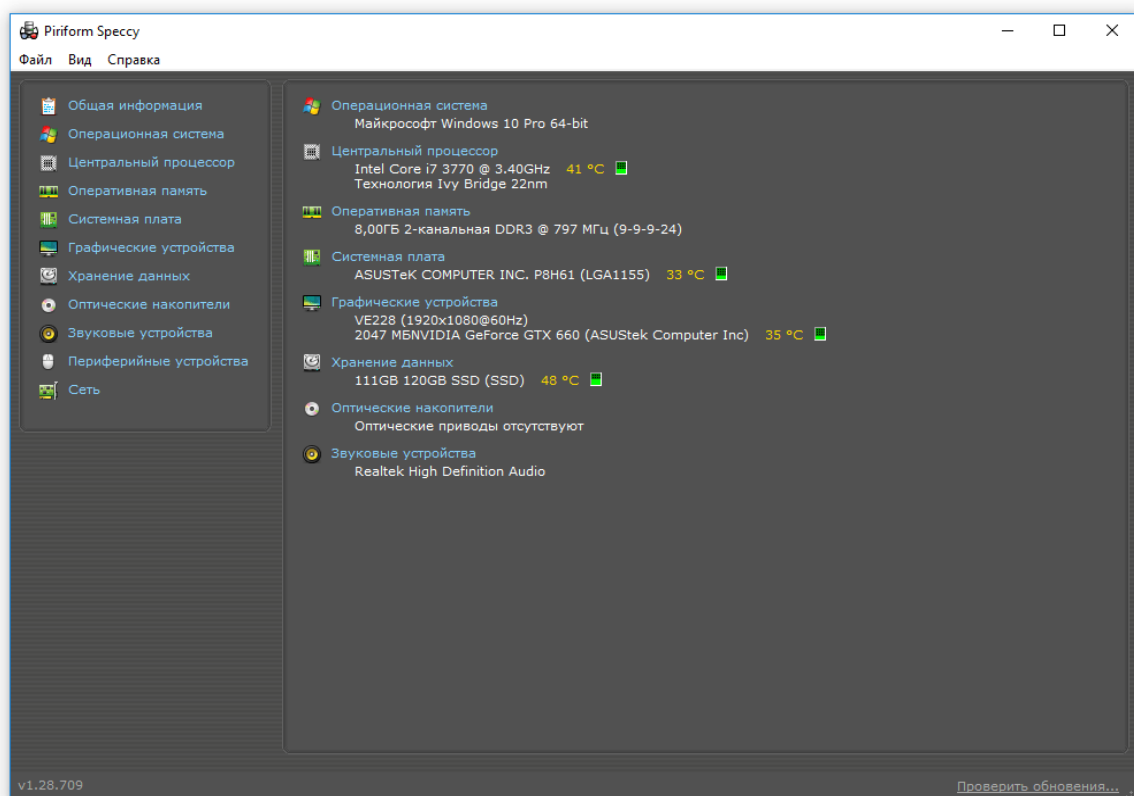


Рисунок 1.6 — Скриншот программы Спессу

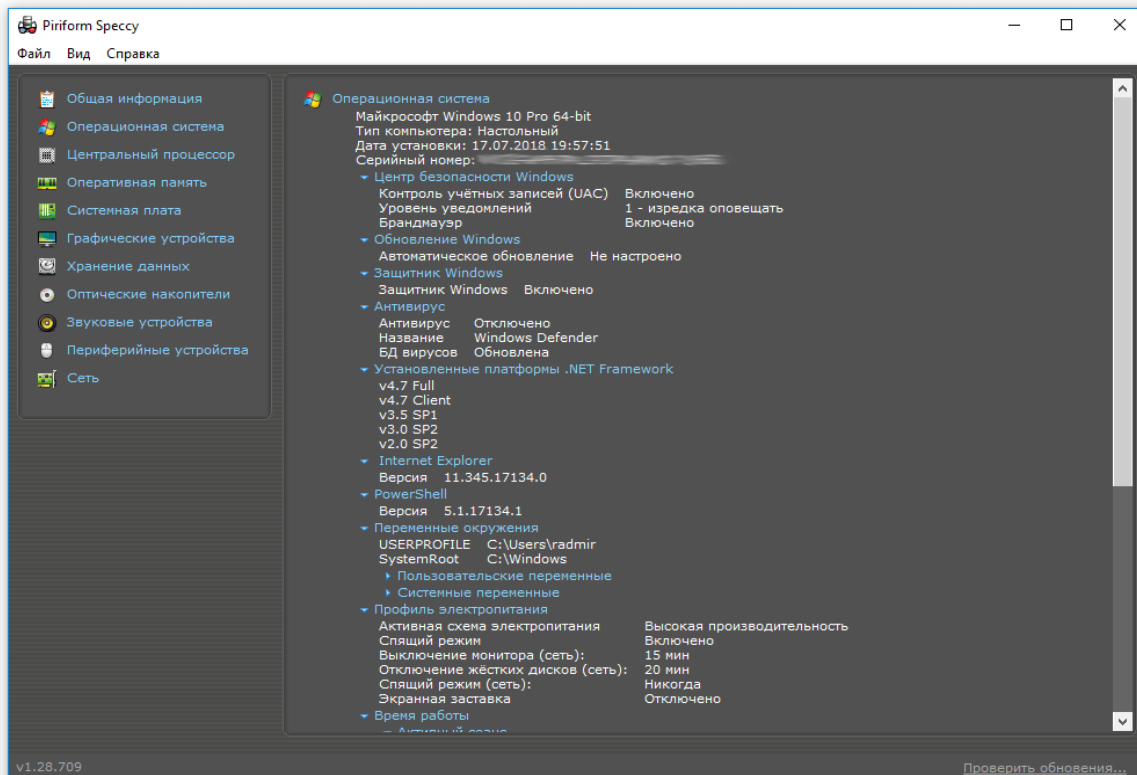


Рисунок 1.7 — Скриншот программы Спессу

1.2.2 AIDA64

AIDA64 – платная профессиональная программа позволяющая получать информацию об аппаратной и программной конфигурации компьютера, а также проводить тестирование отдельных компонентов компьютера или всей системы в целом. Интерфейс и частичный функционал изображены на рисунках 1.8 и 1.9.

Программа анализирует конфигурацию компьютера и выдаёт подробную информацию о:

- установленных в системе устройствах;
- их характеристиках;
- поддерживаемых ими наборах команд и режимах работы;
- установленном программном обеспечении;
- конфигурации операционной системы;
- установленных драйверах;
- автоматически загружаемых программах;
- возможность тестирования компонентов;
- большое количество информации о компонентах.

Недостатки приложения:

- ограниченный функционал бесплатной версии;
- реклама в бесплатной версии.

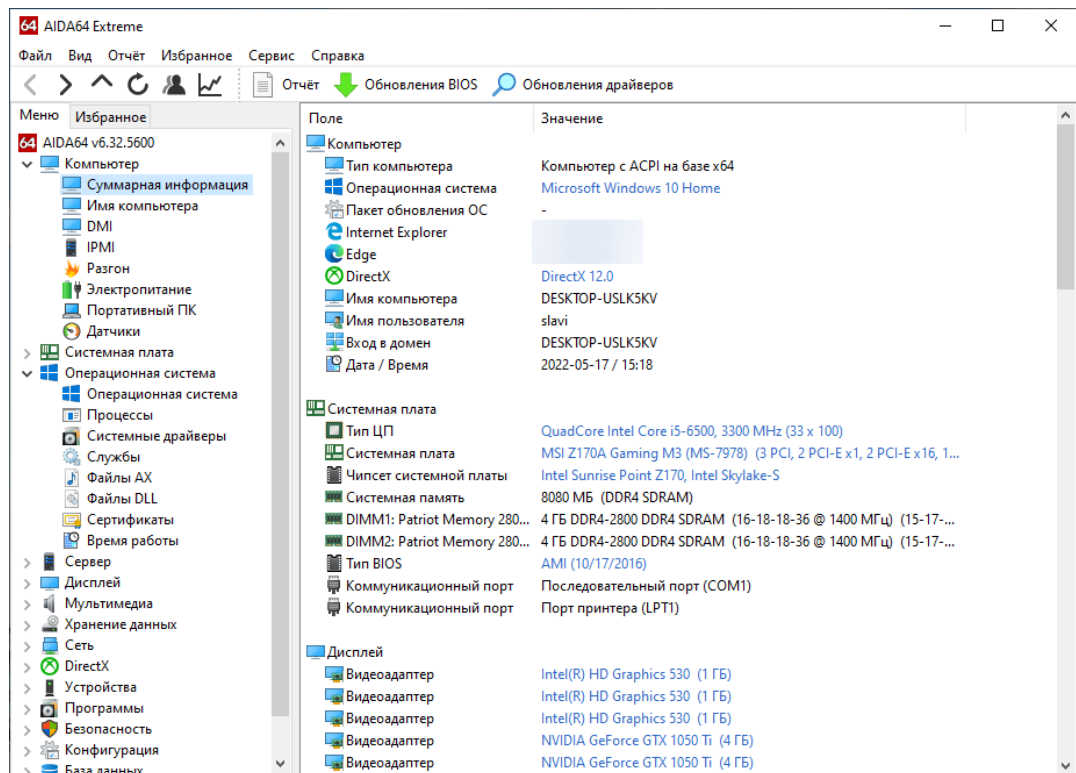


Рисунок 1.8 — Скриншот программы AIDA64

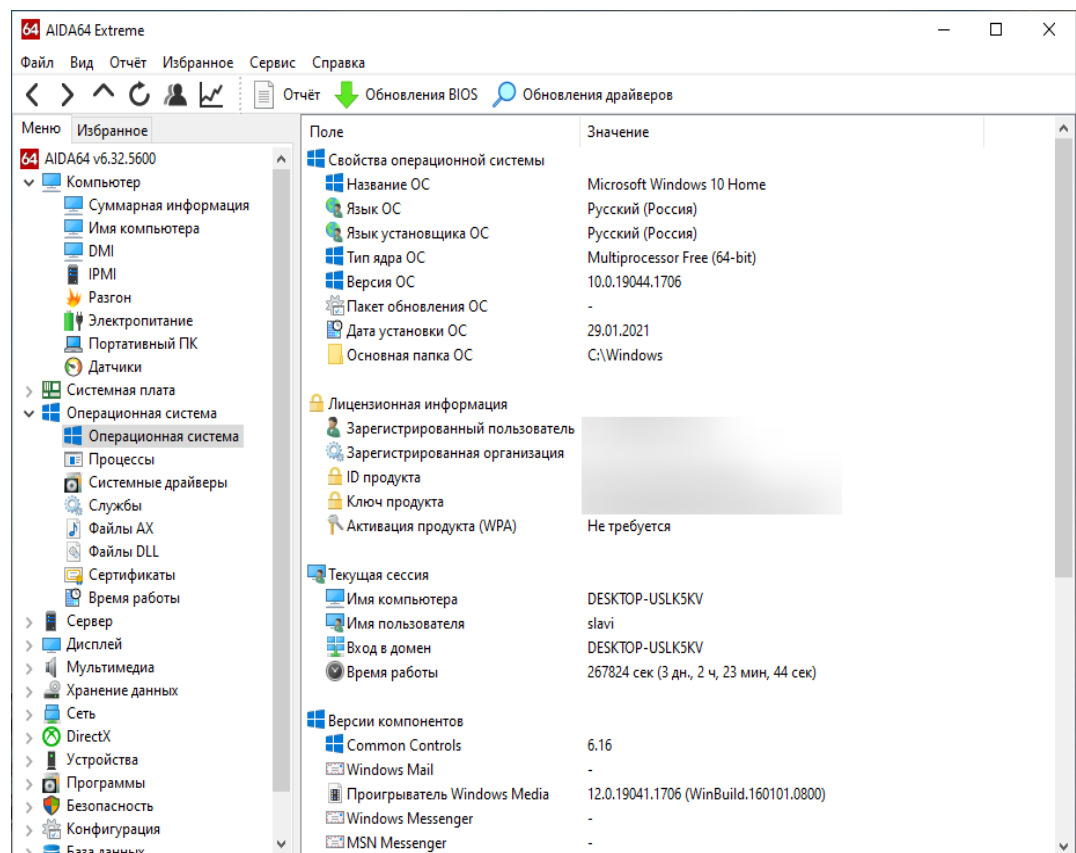


Рисунок 1.9 — Скриншот программы AIDA64

1.2.3 SiSoftware Sandra

SiSoftware Sandra — это системный анализатор для 32-х и 64-х битных версий Windows, включающий в себя тестовые и информационный модули. Sandra объединяет возможности для сравнения производительности как на высоком, так и на низком уровне. Интерфейс и частичный функционал изображены на рисунках 1.10 и 1.11.

Возможности приложения:

- информация о компонентах компьютера;
- тесты производительности процессора, видеоадаптера, накопителей, сети, контроллера памяти;
- тест стабильности системы;
- индекс производительности системы;
- отображение результатов теста в виде графика;
- сравнение результатов с другим оборудованием;
- сохранение отчётов в файл;
- бесплатное распространение;
- русскоязычная локализация;
- модуль «Совет дня» с подсказками о работе;
- наличие онлайн-поддержки пользователей;
- простой удобный интерфейс.

Недостатки приложения:

- многие функции доступны только в платной версии продукта;
- реклама в бесплатной версии продукта.

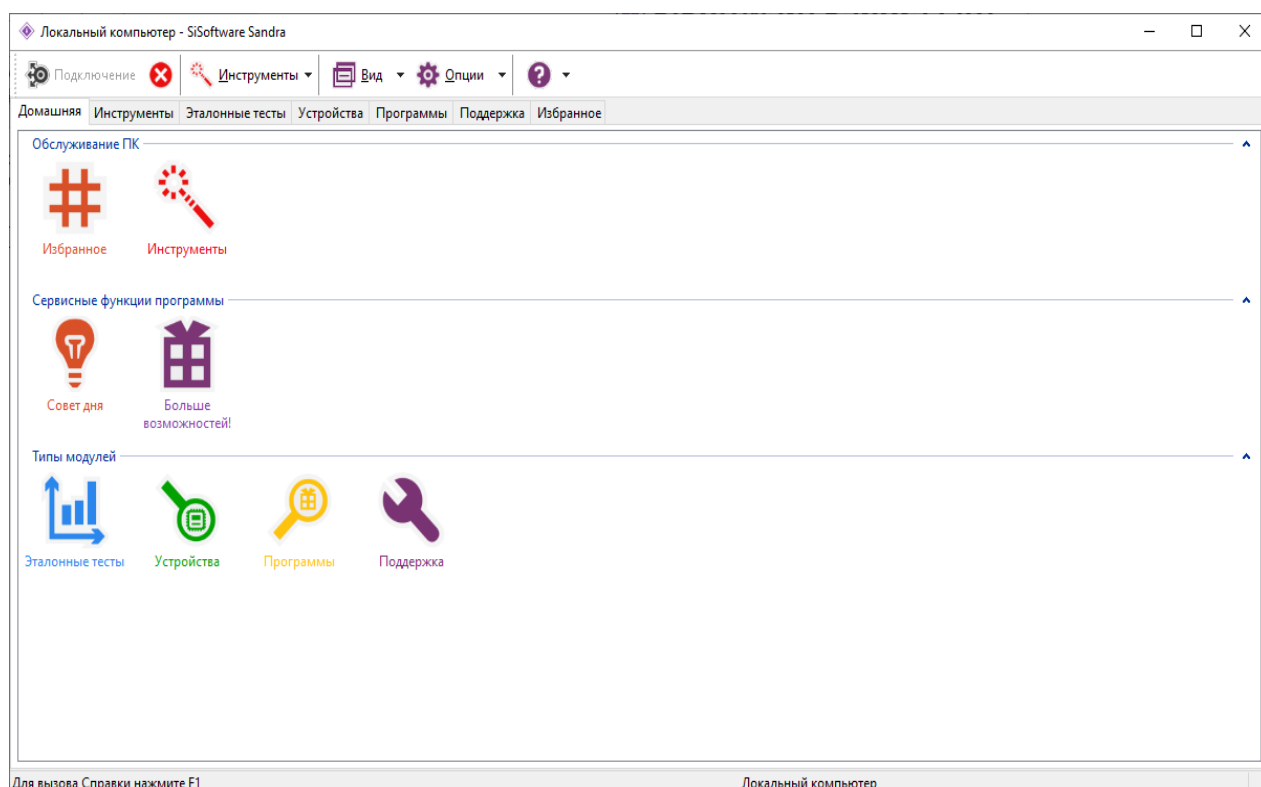


Рисунок 1.10 — Скриншот программы SiSoftware Sandra

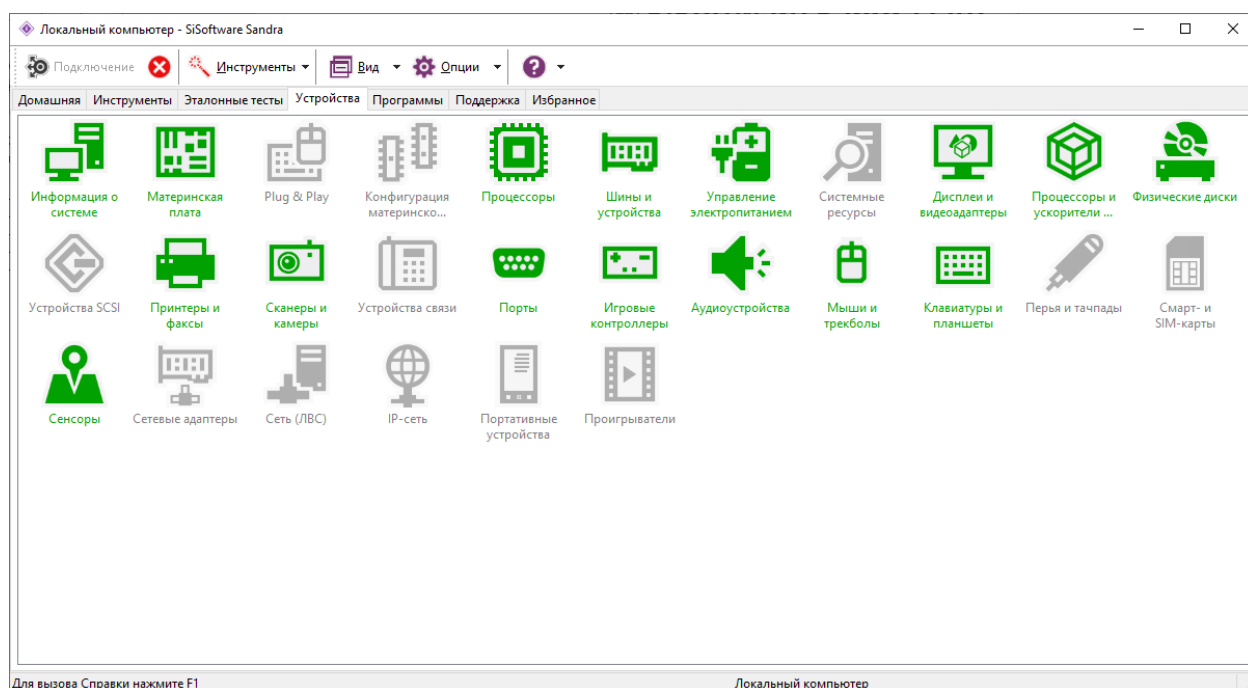


Рисунок 1.11 — Скриншот программы SiSoftware Sandra

1.2.4 I-NEX

I-Nex — это приложение, которое собирает информацию для аппаратных компонентов, доступных в вашей системе, и отображает ее с помощью пользовательского интерфейса, похожего на популярный инструмент Windows CPU-Z. Интерфейс и частичный функционал изображены на рисунках 1.12 и 1.13.

Возможности приложения:

- достаточно простой и минималистичный интерфейс;
- возможность сохранить данные в файл;
- бесплатное распространение;
- русскоязычная локализация;
- отсутствует реклама;
- может генерировать расширенный отчет, для которого вы можете выбрать, что включать, и при необходимости отправить отчет в службу, такую как Pastebin (и другие);
- возможность сделать снимок экрана окна I-Nex непосредственно из приложения;
- информация об оборудовании представлена таким образом, что ее легче понять.

Недостатки приложения:

- редко выходящие обновления;
- ограниченный функционал по сравнению с платными аналогами.

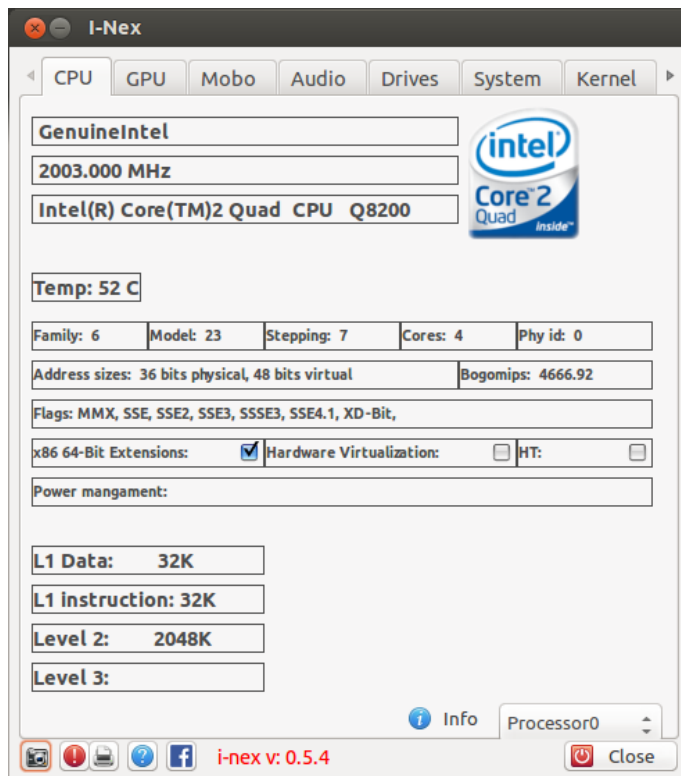


Рисунок 1.12 — Скриншот программы I-Nex

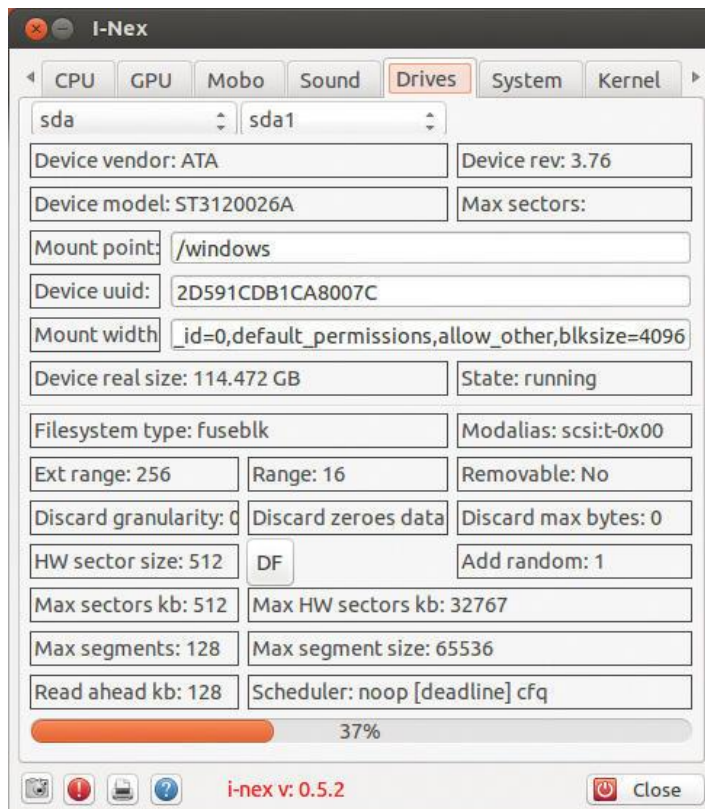


Рисунок 1.13 — Скриншот программы I-Nex

1.3 Постановка задачи

После рассмотрения аналогов можно сказать, что все они обладают большим количеством функций, которые невозможно реализовать в курсовом проекте за данный период времени. Поэтому были выбраны несколько ключевых возможностей, которые будут выполнены в рамках одного семестра.

Будет разработан программный продукт, предоставляющий пользователям возможность просмотра информации о персональном компьютере. Созданное приложение должно иметь простое меню с удобным интерфейсом. Приложение должно быть интуитивно понятно и удобно в использовании.

В качестве языка программирования выбран C++, по причине быстрой производительности, требуемой для обработки большого количества объектов, поддержки объектно-ориентированного подхода к программированию и наличия опыта в использования данного языка. Приложение будет реализовано для операционной системы Linux.

2. СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

2.1 Структура приложения

После определения требований к функционалу разрабатываемого приложения его следует разбить на функциональные блоки. Такой подход упростит понимание проекта, позволит устранить проблемы в архитектуре, обеспечит гибкость и масштабируемость программного продукта в будущем путем добавления новых блоков.

Работа программы разбита на три основных блока: взаимодействие пользователя с приложением, пользовательский интерфейс, обработка пользовательского нажатия и представление информации.

2.1.1 Блок пользовательского интерфейса

Модуль пользовательского интерфейса предназначен для представления и визуализации данных, обработки команд пользователя. Является основным в работе приложения, так как соединяет все остальные блоки между собой через ряд функций и классов для реализации интерфейса. Выводятся все основные функции вывода информации на экран, такие как вывод всей информации, вывод информации по отдельности.

2.1.2 Блок логики приложения

Если от интерфейса пришел запрос на получение каких-либо данных, то он обрабатывается через слой логики, который связывает данные с интерфейсом, отделяя их друг от друга.

2.1.3 Блок управления приложением

Модуль управления приложением является центральным модулем приложения. Он отвечает за все функции приложения, пересылку данных, корректное взаимодействие между всеми остальными компонентами системы.

Структурная схема представлена в приложении А.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описывается функционирование и структура разрабатываемого приложения.

Диаграмма классов представлена в приложении Б.

3.1 Описание работы классов

3.1.1 Пространство имён Util

Функции пространства предоставляют возможность собирать всю информацию от каждого класса по отдельности, а именно: информацию о памяти, имя хоста, список сетевых интерфейсов и их информацию, информацию об операционной системе, список процессов и информацию о них, информацию о процессоре.

Методы:

- void print_all — выводит информацию о всём компьютере;
- void print_hostname — выводит имя пользователя;
- void print_memory — выводит информацию о памяти;
- void print_networklist — выводит информацию о сетях;
- void print_os — выводит информацию об операционной системе;
- void print_processlist — выводит информацию о процессах;
- void print_processor — выводит информацию о процессоре;
- void print_menu — выводит меню выбора.

3.1.2 Класс Hostname

Данный класс возвращает информацию об имени хоста компьютера по средствам библиотеки unistd.h

Поля:

- std::string hostname — имя пользователя.

Методы:

- HostName — конструктор для объекта имени хоста, который находит имя хоста системы и сохраняет его в атрибуте;
- std::string get_hostname — возвращает имя хоста системы;
- std::string to_string — возвращает строковое представление имени хоста.

3.1.3 Класс Memory

Данный класс возвращает информацию, связанную с текущим состоянием памяти и ее размер по средствам библиотеки `sysinfo.h`

Поля:

- `unsigned long totalram` — общий максимальный объем оперативной памяти;
- `unsigned long freeram` — общий свободный объем оперативной памяти;
- `unsigned int mem_unit` — размер блока памяти в байтах.

Методы:

- `Memory` — конструктор для объекта имени хоста, который находит имя хоста системы и сохраняет его в атрибуте;
- `unsigned long get_totalram` — возвращает общий максимальный объем оперативной памяти, принадлежащей компьютеру;
- `unsigned long get_freeram` — возвращает общий свободный объем оперативной памяти, принадлежащей компьютеру;
- `std::string to_string` — возвращает строковое представление информации о памяти.

3.1.4 Класс OperatingSystem

Данный класс возвращает информацию, связанную с операционной системой компьютера по средствам библиотеки `sysinfo.h`

Поля:

- `std::string system_name` — системное имя;
- `std::string os_release` — название выпуска операционной системы;
- `std::string os_version` — версия операционной системы;
- `long uptime` — общее время, в течение которого операционная система была активна.

Методы:

- `OperatingSystem` — конструктор для объекта операционной системы, который вызывает системную функцию и извлекает информацию об операционной системе;
- `std::string get_system_name` — возвращает системное имя, идентифицированное операционной системой;
- `std::string get_os_release` — возвращает название выпуска операционной системы;
- `std::string get_os_version` — возвращает версию операционной системы;

- `long get_uptime` — возвращает общее время, в течение которого операционная система была активна;
- `std::string to_string` — возвращает строковое представление информации об операционной системе.

3.1.5 Класс Processor

Данный класс возвращает информацию об атрибутах с информацией о системном процессоре по средствам библиотеки `sysinfo.h`

Поля:

- `std::string vendor` — имя производителя процессора;
- `std::string model_name` — модель процессора;
- `double clock_speed` — тактовая частота процессора в МГц;
- `float loads[3]` — средние нагрузки за 1, 5 и 15 минут.

Методы:

- `Processor` — конструктор класса процессора, инициализирует атрибуты из системного файла, проанализировав его;
- `double get_clock_speed` — возвращает тактовую частоту в текущее время;
- `float get_cpu_load` — возвращает загрузку процессора, индекс соответствует параметру усреднения по времени: индекс в диапазоне `[0, 2]`;
- `string get_vendor` — возвращает имя производителя, связанного с процессором;
- `std::string get_model_name` — возвращает название модели процессора;
- `std::string to_string` — возвращает строковое представление информации о процессоре.

3.1.6 Класс NetworkList

Данный класс возвращает информацию об интерфейсах, используемых компьютером по средствам библиотеки `dirent.h`

Поля:

- `std::vector<Network> network_list` — список интерфейсов.

Методы:

- `NetworkList` — конструктор класса обращается к каталогу всех интерфейсов и создает список интерфейсов;
- `std::vector<Network> get_network_list` — возвращает вектор списка сетей;
- `std::string to_string` — возвращает строковое представление всех интерфейсов.

3.1.7 Класс Network

Данный класс возвращает информацию из файлов с интерфейсами.

Поля:

- `std::string name` — имя интерфейса;
- `std::string mac_address` — физический адрес, связанный с этим интерфейсом.

Методы:

- `Network` — конструктор для сетевого объекта, который читает из файла интерфейса и заполняет параметр атрибутов;
- `std::string get_name` — возвращает имя этого интерфейса;
- `std::string get_mac_address` — возвращает физический адрес, связанный с этим интерфейсом;
- `std::string to_string` — возвращает строковое представление сетевого интерфейса.

3.1.8 Класс ProcessList

Данный класс возвращает список всех процессов на момент, когда приложение было собрано по средствам библиотеки `dirent.h`.

Поля:

- `std::vector<Process> process_list` — список процессов.

Методы:

- `ProcessList` — конструктор для списка процессов, который открывает каталог процесса и заполняет список объектов процесса на основе содержимого;
- `std::vector<Process> get_process_list` — возвращает вектор всех процессов;
- `std::string to_string` — возвращает строковое представление всех интерфейсов.

3.1.9 Класс Process

Данный класс возвращает данные, связанные с процессами.

Поля:

- `unsigned long pid` — `pid` процесса;
- `unsigned long ppid` — родительский `pid` процесса;
- `std::string name` — имя процесса;
- `std::string owner` — `uid` владельца процесса;
- `std::string state` — состояние процесса.

Методы:

- `Process` — конструктор для процесса, учитывая числовой идентификатор процесса, заполняет параметр атрибутов класса;
- `unsigned long get_pid` — возвращает `pid` процесса;
- `unsigned long get_ppid` — возвращает родительский `pid` процесса;
- `std::string get_name` — возвращает имя процесса;
- `std::string get_owner` — возвращает `uid` владельца процесса;
- `std::string get_state` — возвращает состояние процесса;
- `std::string to_string` — возвращает строковое представление информации о процессе.

4. ПРИНЦИПИАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе рассмотрены описания алгоритмов, используемых в программе.

4.1 Создание и инициализация объектов информационного класса с последующим выводом

Шаг 1. Вызов конструктора базового класса.

Шаг 2. Обращение к необходимому устройству в конструкторе базового класса.

Шаг 3. Установка соединения с выбранным устройством в конструкторе базового класса.

Шаг 4. Последовательная инициализация полей в базовом классе.

Шаг 5. Вывод информации на экран исходя из полей базового класса.

4.2 Алгоритм работы приложения

Шаг 1. Запуск приложения.

Шаг 2. Циклическое создание, а также инициализация объектов класса.

Шаг 3. Циклическое отображение полученной информации в консоли.

Шаг 4. Завершение работы приложения при выборе соответствующего пункта меню.

4.3 Алгоритм считывания информации о процессоре в конструкторе класса Processor

Шаг 1. Начало.

Шаг 2. Открываем для чтения файл CPU_INFO_FILE.

Шаг 3. Сохраняем список ключей, которые будем искать в файле.

Шаг 4. Создаём буфер для хранения информации.

Шаг 5. Начало цикла по файлу до тех пор, пока файл не закончится.

Шаг 6. Проверяем, соответствует ли ключ строки искомой информации.

Шаг 7. Если да, то будем использовать оператор присваивания копии из `std::string` для заполнения данных.

Шаг 8. После завершения цикла создаём объект структуры `sysinfo` для получения загруженности системы за одну, пять и пятнадцать минут.

Шаг 9. Записываем информацию в поле.

Шаг 10. Закрываем файл CPU_INFO_FILE.

Шаг 11. Конец.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

5.1 Системные требования

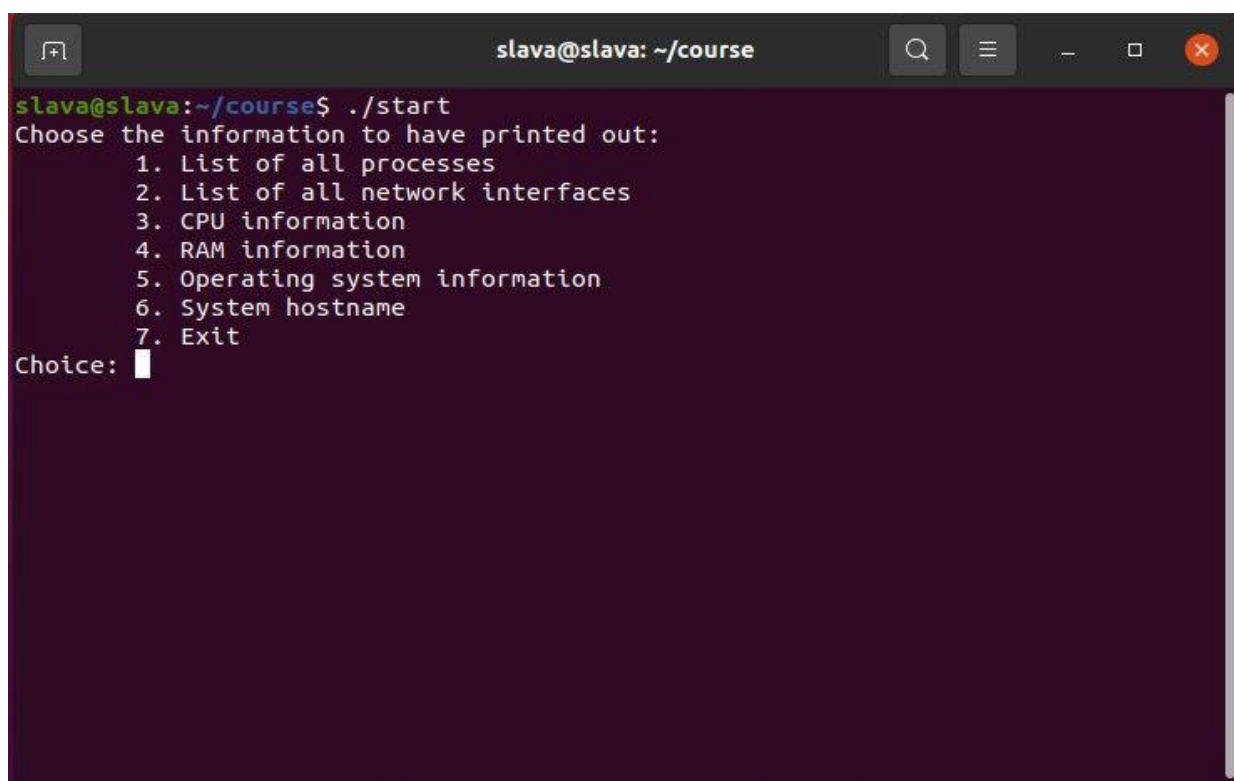
Для запуска данной программы необходим персональный компьютер с установленной любой UNIX-системой подобной Linux, в частности практически любым дистрибутивом Linux.

5.2 Использование приложения

Разработанное программное средство просто в использовании. Ниже будут даны описания последовательности действий, необходимых для успешной работы с приложением.

Для запуска приложения, необходимо в консоли указать полный путь к исполняющему файлу “start” и запустить его.

При запуске программы пользователю представляется стартовое меню для выбора и последующего вывода нужной ему информации о системе, которое можно увидеть на скриншоте ниже. После вывода информации программа сама предложит повторный выбор пользователю, для завершения работы программы необходимо выбрать соответствующий пункт в меню (рисунок 5.1).



```
slava@slava: ~/course
slava@slava:~/course$ ./start
Choose the information to have printed out:
 1. List of all processes
 2. List of all network interfaces
 3. CPU information
 4. RAM information
 5. Operating system information
 6. System hostname
 7. Exit
Choice: █
```

Рисунок 5.1 — Запуск программ

Выбрав первый пункт меню, можно посмотреть информацию о процессах: их имя, UID владельца процесса, PID родителя и статус процесса (рисунок 5.2).

```
slava@slava: ~/course
Choice: 1
-
Full process list description:
*****
Process information for PID 1
  Name is 'systemd'
  Owner UID is 0 0 0 0
  Parent PID is 0
  The process is in state S (sleeping)
Process information for PID 2
  Name is 'kthreadd'
  Owner UID is 0 0 0 0
  Parent PID is 0
  The process is in state S (sleeping)
Process information for PID 3
  Name is 'rcu_gp'
  Owner UID is 0 0 0 0
  Parent PID is 2
  The process is in state I (idle)
Process information for PID 4
  Name is 'rcu_par_gp'
  Owner UID is 0 0 0 0
  Parent PID is 2
```

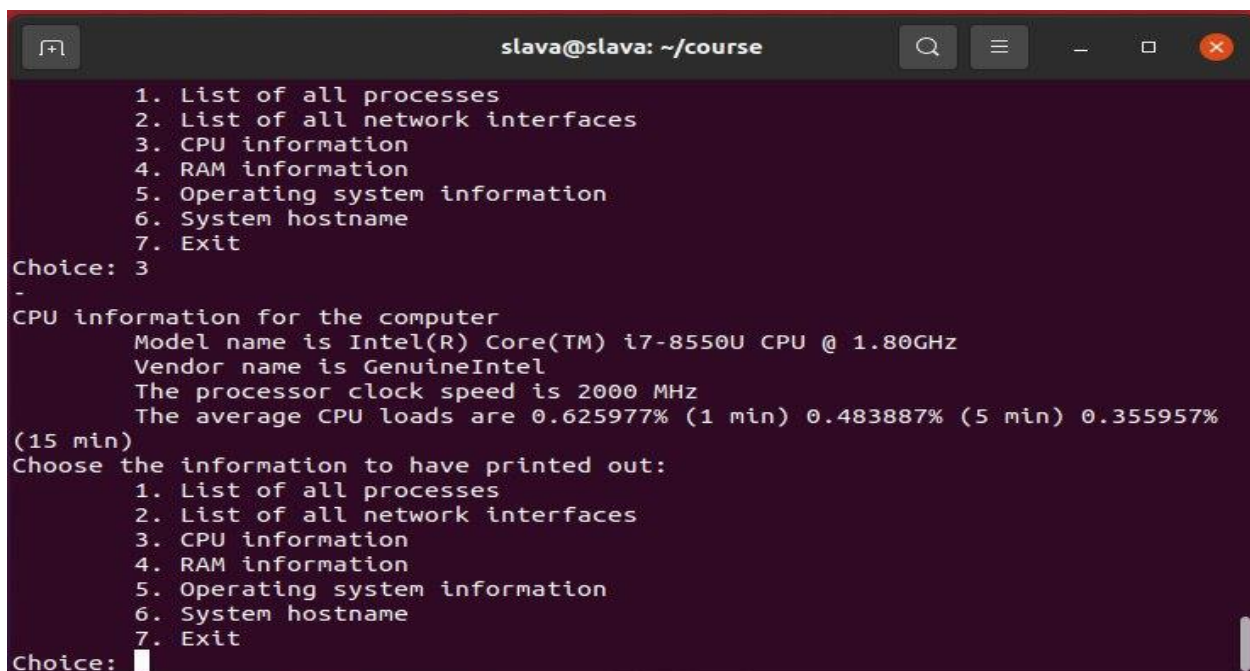
Рисунок 5.2 — Список процессов

После выбора второго пункта меню пользователю будет выведен список всех сетевых интерфейсов. В нём будет выведена информация об их имя и MAC-адрес (рисунок 5.3).

```
slava@slava: ~/course
6. System hostname
7. Exit
Choice: 2
-
Full network interface description:
*****
Network information for interface lo
  Mac address is 00:00:00:00:00:00
Network information for interface wlp3s0
  Mac address is 80:2b:f9:df:66:11
Network information for interface enp2s0
  Mac address is 3c:2c:30:b3:7a:5c
*****
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: █
```

Рисунок 5.3 — Список сетевых интерфейсов

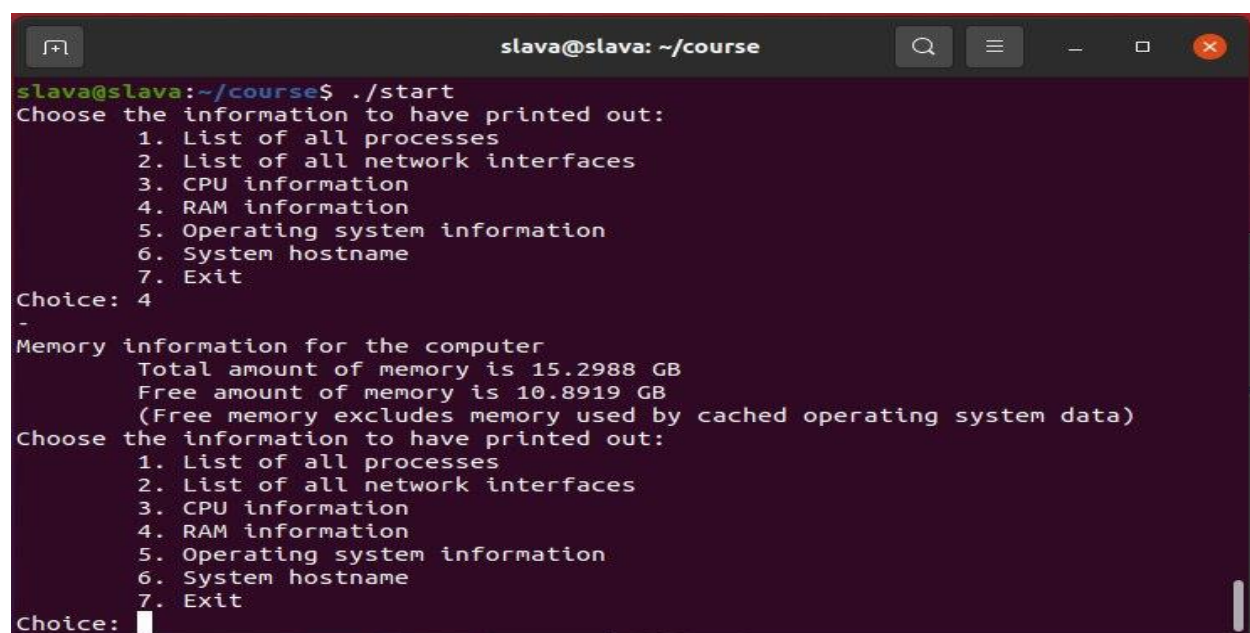
Выбор третьего пункта меню покажет пользователю информацию о процессоре, его модель, производителя, частоту и средние показатели нагрузки за разные периоды времени (рисунок 5.4).

A terminal window titled 'slava@slava: ~/course' with a dark purple background. It displays a menu with 7 options. Option 3 is selected, leading to 'CPU information for the computer'. The output shows the model name 'Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz', vendor 'GenuineIntel', clock speed '2000 MHz', and average CPU loads for 1, 5, and 15 minutes. A second menu is shown at the bottom, with the cursor at the 'Choice:' prompt.

```
slava@slava: ~/course
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: 3
-
CPU information for the computer
Model name is Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Vendor name is GenuineIntel
The processor clock speed is 2000 MHz
The average CPU loads are 0.625977% (1 min) 0.483887% (5 min) 0.355957%
(15 min)
Choose the information to have printed out:
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: █
```

Рисунок 5.4 — Информация о процессоре

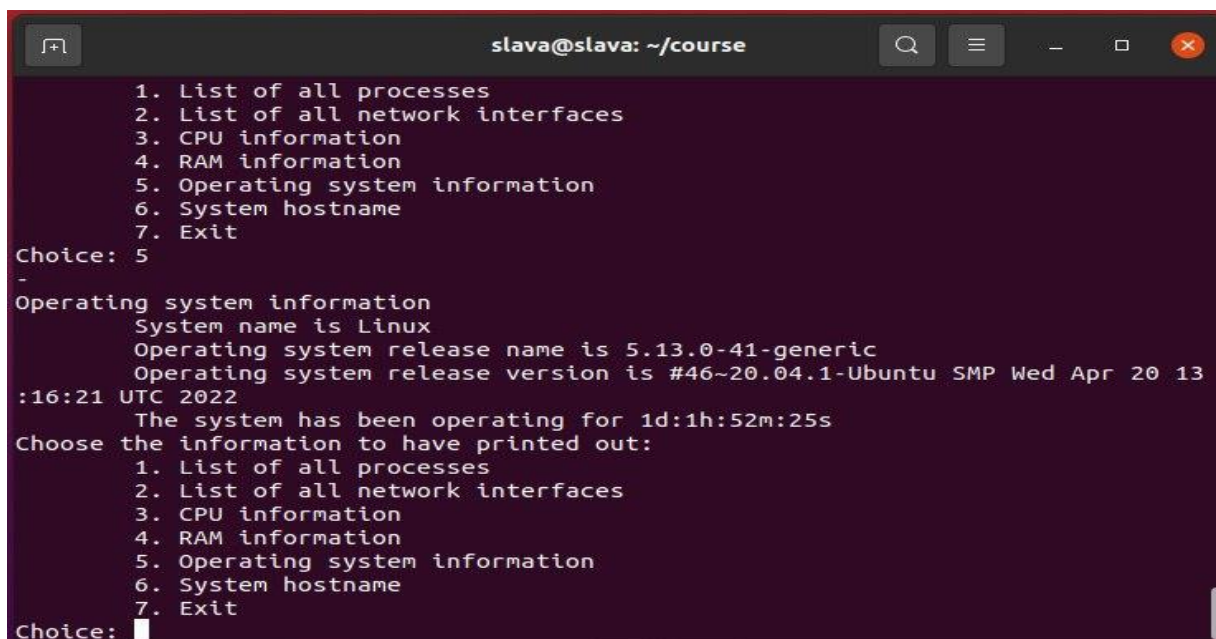
Для просмотра информации об оперативной памяти компьютера нужно выбрать 4 пункт меню. На консоли будет выведена о количестве общем оперативной памяти и свободной памяти (рисунок 5.5).

A terminal window titled 'slava@slava: ~/course' with a dark purple background. It shows the command './start' being executed. A menu is displayed, and option 4 is selected. The output shows 'Memory information for the computer' with 'Total amount of memory is 15.2988 GB' and 'Free amount of memory is 10.8919 GB'. A note states '(Free memory excludes memory used by cached operating system data)'. A second menu is shown at the bottom, with the cursor at the 'Choice:' prompt.

```
slava@slava:~/course$ ./start
Choose the information to have printed out:
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: 4
-
Memory information for the computer
Total amount of memory is 15.2988 GB
Free amount of memory is 10.8919 GB
(Free memory excludes memory used by cached operating system data)
Choose the information to have printed out:
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: █
```

Рисунок 5.5 — Информация о процессоре

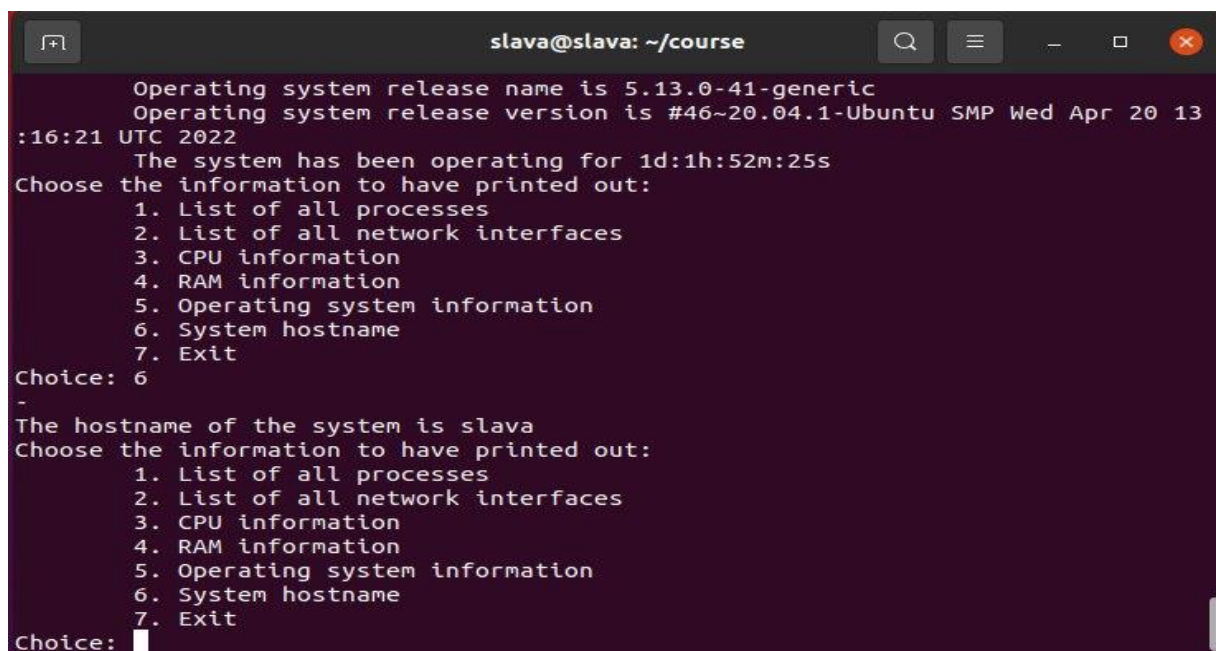
При выборе пятого пункта меню показывается информация об операционной системе: название ОС, название выпуска, версия выпуска и общее время, в течении которого операционная система была активна (рисунок 5.6).

A terminal window titled 'slava@slava: ~/course' with a dark purple background. It displays a menu with 7 options. Option 5 is selected, showing system information: 'System name is Linux', 'Operating system release name is 5.13.0-41-generic', 'Operating system release version is #46~20.04.1-Ubuntu SMP Wed Apr 20 13:16:21 UTC 2022', and 'The system has been operating for 1d:1h:52m:25s'. A second menu is shown below, with option 5 again selected.

```
slava@slava: ~/course
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: 5
-
Operating system information
System name is Linux
Operating system release name is 5.13.0-41-generic
Operating system release version is #46~20.04.1-Ubuntu SMP Wed Apr 20 13:16:21 UTC 2022
The system has been operating for 1d:1h:52m:25s
Choose the information to have printed out:
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: 5
```

Рисунок 5.6 — Информация об операционной системе

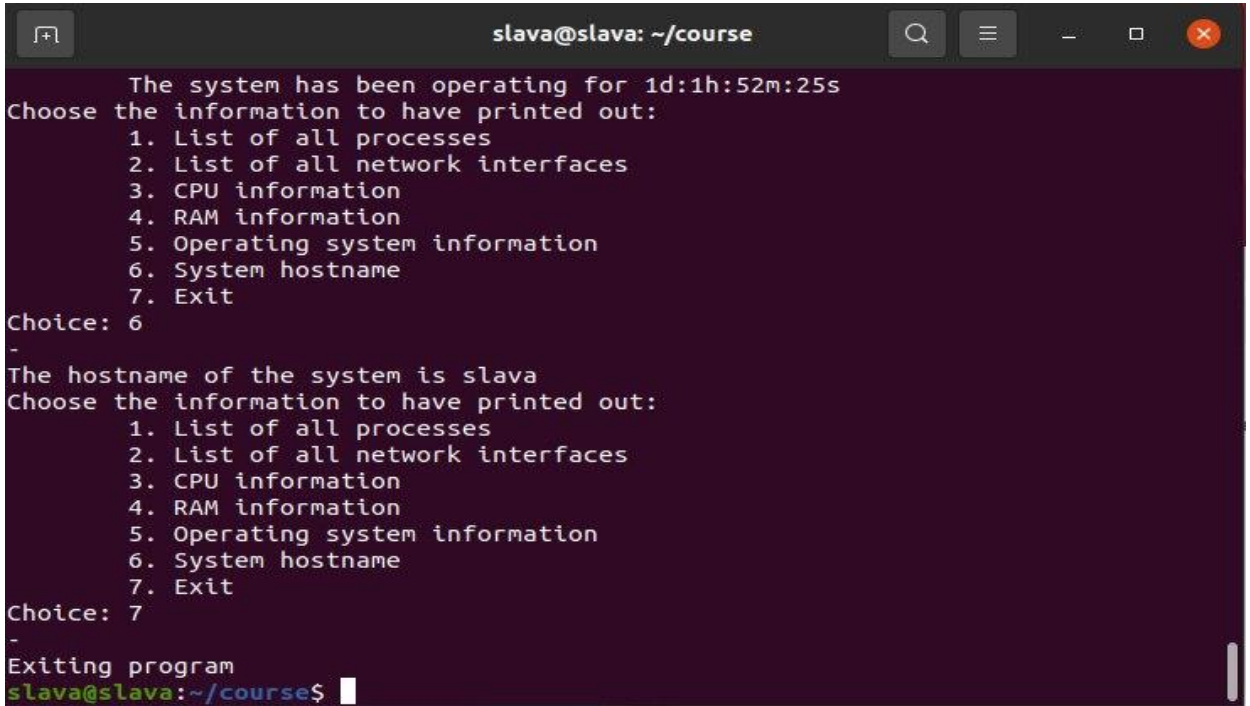
При выборе шестого пункта меню показывается информация об имени пользователя компьютера (рисунок 5.7).

A terminal window titled 'slava@slava: ~/course' with a dark purple background. It displays the same system information as Figure 5.6. Below it, option 6 is selected from the second menu, showing the hostname: 'The hostname of the system is slava'. A third menu is shown below, with option 6 again selected.

```
slava@slava: ~/course
Operating system release name is 5.13.0-41-generic
Operating system release version is #46~20.04.1-Ubuntu SMP Wed Apr 20 13:16:21 UTC 2022
The system has been operating for 1d:1h:52m:25s
Choose the information to have printed out:
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: 6
-
The hostname of the system is slava
Choose the information to have printed out:
1. List of all processes
2. List of all network interfaces
3. CPU information
4. RAM information
5. Operating system information
6. System hostname
7. Exit
Choice: 6
```

Рисунок 5.7 — Имя пользователя

Для выхода из программы пользователю требуется выбрать шестой пункт меню, после этого пользователь дальше сможет работать с терминалом в Linux (рисунок 5.8).



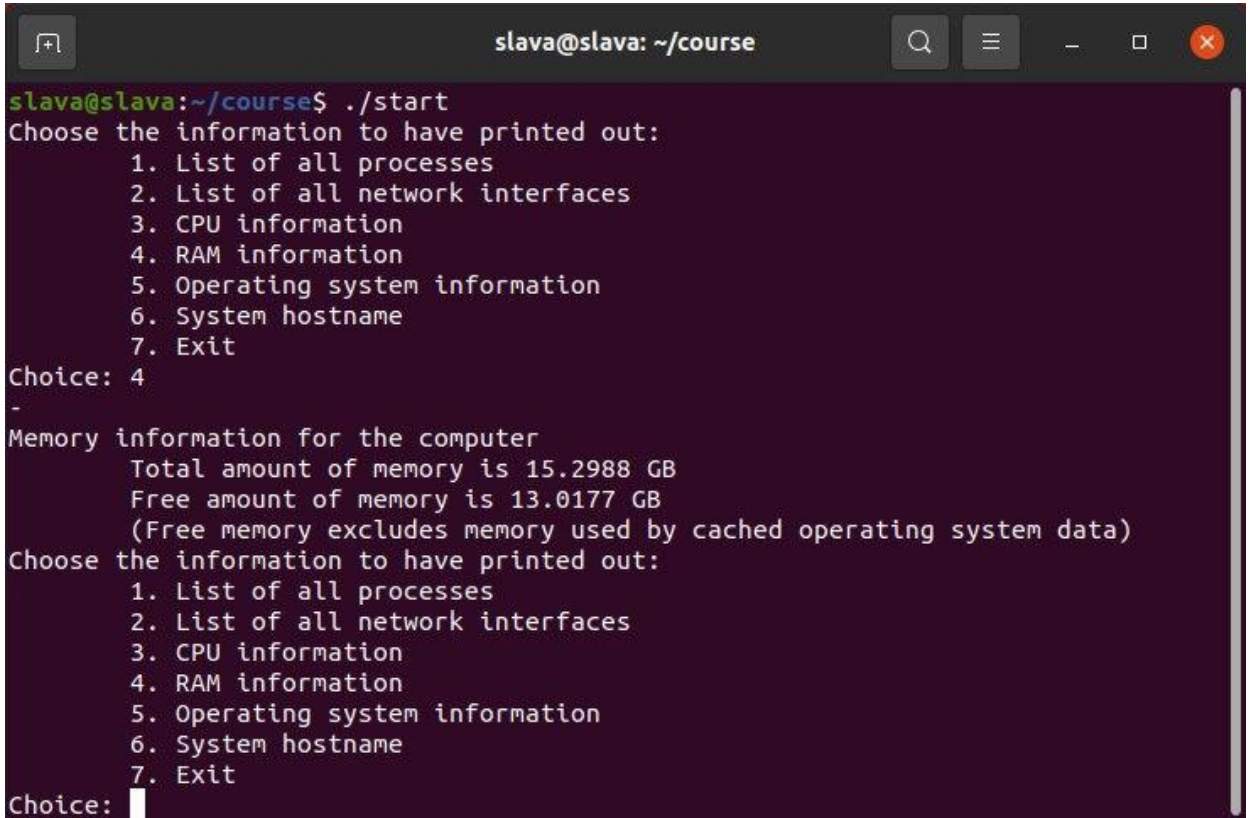
```
slava@slava: ~/course
The system has been operating for 1d:1h:52m:25s
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: 6
-
The hostname of the system is slava
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: 7
-
Exiting program
slava@slava:~/course$
```

Рисунок 5.8 — Выход из программы

6 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

6.1 Проверка работы ввода выбранного пункта пользователем

Цель: Проверка работы программы при вводе выбора необходимого пользователю пункта меню (рисунок 6.1).



```
slava@slava: ~/course
slava@slava:~/course$ ./start
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: 4
-
Memory information for the computer
  Total amount of memory is 15.2988 GB
  Free amount of memory is 13.0177 GB
  (Free memory excludes memory used by cached operating system data)
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: █
```

Рисунок 6.1 – Корректный выбор пользователя

Результат: при вводе нужного пункта меню, информация и дальнейшее продолжение программы выводится корректно.

6.2 Проверка работы ввода при ошибочном вводе пользователем

Цель: Проверка работы программы при ошибочном вводе выбора пользователя (рисунок 6.2).

```
slava@slava: ~/course
slava@slava:~/course$ ./start
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: d
-
Wrong number
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: 123
-
Wrong number
Choose the information to have printed out:
  1. List of all processes
  2. List of all network interfaces
  3. CPU information
  4. RAM information
  5. Operating system information
  6. System hostname
  7. Exit
Choice: █
```

Рисунок 6.2 – Некорректный ввод пользователя

Результат: при ошибочном вводе выводится сообщение об ошибке и пользователю заново выводится текст с информацией для повторного выбора пользователем меню.

ЗАКЛЮЧЕНИЕ

В результате работы над данным курсовым проектом было разработано работоспособное приложение со своим набором функций и интерфейсом. Данный курсовой проект был разработан в соответствии с поставленными задачами, весь функционал был реализован в полном объеме.

При создании проекта были исследованы библиотеки для работы и просмотра информации о компьютере. В ходе разработки были углублены знания языка программирования C++ и в области объектно-ориентированного программирования, а также получен опыт работы с операционной системой Linux.

Работа была разделена на такие этапы, как анализ существующих аналогов, литературных источников, постановка требований к проектируемому программному продукту, системное и функциональное проектирование, разработка программных модулей и тестирование проекта. После последовательного выполнения вышеперечисленных этапов разработки было получено исправно работающее приложение.

Перспективными путями дальнейшего развития проекта является создание непосредственно приложения с удобным графическим интерфейсом, увеличение количества отображаемой информации, отображение динамических характеристик компонентов системы и любое прочее усовершенствование текущего функционала приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1]. Русскоязычная документация по семейству операционных систем Ubuntu [Электронный ресурс]. – Режим доступа: <https://help.ubuntu.ru>.
- [2]. Opennet – форума с большим количеством информации о Linux [Электронный ресурс]. – Режим доступа: <https://www.opennet.ru>.
- [3]. Лафоре, Р. Объектно-ориентированное программирование в C++. 4-е изд. / Р. Лафоре – СПб.: Питер, 2011. – 928 с.: ил.
- [4]. Хабр – информационный портал для разработчиков [Электронный ресурс]. – Режим доступа: <https://habr.com/ru>.

ПРИЛОЖЕНИЕ А
(Обязательное)

Структурная схема программы

ПРИЛОЖЕНИЕ Б
(Обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ В
(Обязательное)

Код программы

Файл utility.h:

```
#include <iostream>
#include <string>
#include <limits>
#include "hostname.h"
#include "memory.h"
#include "networklist.h"
#include "operatingsystem.h"
#include "processlist.h"
#include "processor.h"
namespace util
{
    void print_all(std::ostream &out);
    void print_hostname(std::ostream &out);
    void print_memory(std::ostream &out);
    void print_networklist(std::ostream &out);
    void print_os(std::ostream &out);
    void print_processlist(std::ostream &out);
    void print_processor(std::ostream &out);
    void print_menu(std::ostream &out, std::istream &in);
}
```

Файл utility.cpp:

```
#include "utility.h"
void util::print_all(std::ostream &out)
{
    out
        << ProcessList().to_string() << "\n"
        << "-\n"
        << Processor().to_string() << "\n"
        << "-\n"
        << HostName().to_string() << "\n"
        << "-\n"
        << Memory().to_string() << "\n"
        << "-\n"
        << NetworkList().to_string() << "\n"
        << "-\n"
        << OperatingSystem().to_string()
        << "-" << std::endl;
}
void util::print_hostname(std::ostream &out)
{
    out << HostName().to_string() << std::endl;
}
void util::print_memory(std::ostream &out)
{
    out << Memory().to_string() << std::endl;
}
void util::print_networklist(std::ostream &out)
{
    out << NetworkList().to_string() << std::endl;
}
void util::print_os(std::ostream &out)
{
    out << OperatingSystem().to_string() << std::endl;
}
void util::print_processlist(std::ostream &out)
{
    out << ProcessList().to_string() << std::endl;
}
```

```

}
void util::print_processor(std::ostream &out)
{
    out << Processor().to_string() << std::endl;
}
void util::print_menu(std::ostream &out, std::istream &in)
{
    bool running = true;
    while (running)
    {
        out << "Choose the information to have printed out:\n"
            << "\t1. List of all processes\n"
            << "\t2. List of all network interfaces\n"
            << "\t3. CPU information\n"
            << "\t4. RAM information\n"
            << "\t5. Operating system information\n"
            << "\t6. System hostname\n"
            << "\t7. Exit" << std::endl;

        out << "Choice: ";
        int ch;
        while (!(in >> ch) && ch < 1 && ch > 6) {
            out << "Bad value!";
            in.clear();
            in.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        }
        out << "-\n";
        switch (ch)
        {
            case 1:
                print_processlist(out);
                break;
            case 2:
                print_networklist(out);
                break;
            case 3:
                print_processor(out);
                break;
            case 4:
                print_memory(out);
                break;
            case 5:
                print_os(out);
                break;
            case 6:
                print_hostname(out);
                break;
            case 7:
                out << "Exiting program" << std::endl;
                running = false;
        }
    }
}

```

Файл processor.h:

```

#include <string>
#include <cstring>
#include <iostream>
#include <sstream>
#include <algorithm>

```

```

#include <sys/sysinfo.h>
#include "errorcodes.h"
#define CPU_INFO_FILE "/proc/cpuinfo"

class Processor
{
    std::string vendor;
    std::string model_name;
    double clock_speed;
    float loads[3];

public:
    Processor();
    double get_clock_speed() const;
    float get_cpu_load(size_t index) const;
    std::string get_vendor() const;
    std::string get_model_name() const;
    std::string to_string() const;
};

```

Файл processor.cpp:

```

#include "processor.h"
Processor::Processor()
{
    FILE *f;
    f = fopen(CPU_INFO_FILE, "r");
    char vendor_key[] = "vendor_id";
    char model_name_key[] = "model name";
    char clock_speed_key[] = "cpu MHz";
    size_t constexpr LINE_BUFFER = 256;
    char line[LINE_BUFFER];
    while (fgets(line, LINE_BUFFER, f))
    {
        char *key = strtok(line, "\t");
        if (strcmp(key, vendor_key) == 0)
        {
            char *value = strtok(NULL, ":\t\n") + 1;
            vendor = value;
        }
        else if (strcmp(key, model_name_key) == 0)
        {
            char *value = strtok(NULL, ":\t\n") + 1;
            model_name = value;
        }
        else if (strcmp(key, clock_speed_key) == 0)
        {
            char *value = strtok(NULL, ":\t\n") + 1;
            clock_speed = atof(value);
        }
    }

    fclose(f);
    struct sysinfo s_buf;
    if (sysinfo(&s_buf) != 0)
    {
        fprintf(stderr, "sysinfo function did not return\n");
        exit(SYS_FUNC_ERROR);
    }
    std::transform(

```

```

        s_buf.loads, s_buf.loads + 3, loads,
        [](auto e) { return e / (static_cast<float>(1 <<
SI_LOAD_SHIFT)); }
    );
}
double Processor::get_clock_speed() const
{
    return clock_speed;
}
float Processor::get_cpu_load(size_t index) const
{
    if (index > 2 || index < 0)
    {
        fprintf(stderr, "incorrect index provided for cpu load\n");
        exit(INDEX_OUT_OF_BOUNDS);
    }
    else
    {
        return loads[index];
    }
}
std::string Processor::get_vendor() const
{
    return vendor;
}
std::string Processor::get_model_name() const
{
    return model_name;
}
std::string Processor::to_string() const
{
    std::stringstream ss;
    ss << "CPU information for the computer";
    ss << "\n\tModel name is " << model_name;
    ss << "\n\tVendor name is " << vendor;
    ss << "\n\tThe processor clock speed is " << clock_speed << " MHz";
    ss << "\n\tThe average CPU loads are " << loads[0] << "% (1 min) "
        << loads[1] << "% (5 min) " << loads[2] << "% (15 min)";
    return ss.str();
}

```

Файл processlist.h:

```

#include <sys/types.h>
#include <cctype>
#include <dirent.h>
#include <vector>
#include <sstream>
#include "errorcodes.h"
#include "process.h"
class ProcessList
{
    std::vector<Process> process_list;
public:
    ProcessList();
    std::vector<Process> get_process_list() const;
    std::string to_string() const;
};

```

Файл processlist.cpp:

```
#include "processlist.h"

ProcessList::ProcessList()
{
    DIR *d;
    dirent *ent;
    if ((d = opendir(PROC_INFO_DIR)) == 0)
    {
        fprintf(stderr, "the %s directory failed to open\n",
PROC_INFO_DIR);
        exit(DIR_OPEN_ERROR);
    }

    while ((ent = readdir(d)) != 0)
    {
        bool b = true;
        for (const char *c = ent->d_name; *c != 0 && (b = isdigit(*c));
++c);
        if (b)
        {
            process_list.emplace_back(atoi(ent->d_name));
        }
    }

    std::vector<Process> ProcessList::get_process_list() const
    {
        return process_list;
    }

    std::string ProcessList::to_string() const
    {
        std::stringstream ss;
        ss << "Full process list description:\n"
<<
"*****\n";
        for (auto const& process : process_list)
        {
            ss << process.to_string() << "\n";
        }
        ss
<<
"*****";
        return ss.str();
    }
}
```

Файл process.h:

```
#include <sstream>
#include <string>
#include <vector>
#include <cstring>
#include "errorcodes.h"
#define PROC_INFO_DIR "/proc"
#define PROC_INFO_FILE "/proc/%d/status"
```

```

class Process
{
    unsigned long pid;
    unsigned long ppid;
    std::string name;
    std::string owner;
    std::string state;
public:
    Process(unsigned long pid);
    unsigned long get_pid() const;
    unsigned long get_ppid() const;
    std::string get_name() const;
    std::string get_owner() const;
    std::string get_state() const;
    std::string to_string() const;
};

```

Файл process.cpp:

```

#include "process.h"
Process::Process(unsigned long pid) : pid{ pid }
{
    char fname[32];
    sprintf(fname, PROC_INFO_FILE, pid);
    FILE *f;
    if ((f = fopen(fname, "r")) == 0)
    {
        fprintf(stderr, "file %s failed to open\n", fname);
        exit(FILE_OPEN_ERROR);
    }

    char name_key[] = "Name";
    char owner_key[] = "Uid";
    char state_key[] = "State";
    char ppid_key[] = "PPid";
    size_t constexpr LINE_BUFFER = 256;
    char line[LINE_BUFFER];
    while (fgets(line, LINE_BUFFER, f))
    {
        char *key = strtok(line, ":");
        if (strcmp(key, name_key) == 0)
        {
            char *value = strtok(NULL, "\t\n");
            name = value;
        }
        else if (strcmp(key, owner_key) == 0)
        {
            char *value = strtok(NULL, "\n");
            owner = value;
        }
        else if (strcmp(key, state_key) == 0)
        {
            char *value = strtok(NULL, "\t\n");
            state = value;
        }
        else if (strcmp(key, ppid_key) == 0)
        {
            char *value = strtok(NULL, "\n");
            ppid = atoi(value);
        }
    }
}

```

```

        fclose(f);
    }
    unsigned long Process::get_pid() const
    {
        return pid;
    }
    unsigned long Process::get_ppid() const
    {
        return ppid;
    }
    std::string Process::get_name() const
    {
        return name;
    }
    std::string Process::get_owner() const
    {
        return owner;
    }
    std::string Process::get_state() const
    {
        return state;
    }
    std::string Process::to_string() const
    {
        std::stringstream ss;
        ss << "Process information for PID " << pid;
        ss << "\n\tName is '" << name << "'";
        ss << "\n\tOwner UID is " << owner;
        ss << "\n\tParent PID is " << ppid;
        ss << "\n\tThe process is in state " << state;
        return ss.str();
    }
}

```

Файл operatingsystem.h:

```

#include <string>
#include <sstream>
#include <sys/utsname.h>
#include <sys/sysinfo.h>
#include "errorcodes.h"
class OperatingSystem
{
    std::string system_name;
    std::string os_release;
    std::string os_version;
    long uptime;
public:
    OperatingSystem();
    std::string get_system_name() const;
    std::string get_os_release() const;
    std::string get_os_version() const;
    long get_uptime() const;
    std::string to_string() const;
};

```

Файл operatingsystem.cpp:

```

#include "operatingsystem.h"
OperatingSystem::OperatingSystem()
{

```

```

    struct utsname u_buf;
    if (uname(&u_buf) != 0)
    {
        fprintf(stderr, "uname function did not return correctly\n");
        exit(SYS_FUNC_ERROR);
    }
    system_name = u_buf.sysname;
    os_release = u_buf.release;
    os_version = u_buf.version;
    struct sysinfo s_buf;
    if (sysinfo(&s_buf) != 0)
    {
        fprintf(stderr, "sysinfo function did not return
correctly\n");
        exit(SYS_FUNC_ERROR);
    }
    uptime = s_buf.uptime;
}
std::string OperatingSystem::get_system_name() const
{
    return system_name;
}
std::string OperatingSystem::get_os_release() const
{
    return os_release;
}
std::string OperatingSystem::get_os_version() const
{
    return os_version;
}
long OperatingSystem::get_uptime() const
{
    return uptime;
}
std::string OperatingSystem::to_string() const
{
    int day, hour, min, second;
    int constexpr
    day_s = 86400,
    hour_s = 3600,
    min_s = 60;
    day = uptime / day_s;
    hour = (uptime - day * day_s) / hour_s;
    min = (uptime - day * day_s - hour * hour_s) / min_s;
    second = uptime - day * day_s - hour * hour_s - min * min_s;
    std::stringstream ss;
    ss << "Operating system information ";
    ss << "\n\tSystem name is " << system_name;
    ss << "\n\tOperating system release name is " << os_release;
    ss << "\n\tOperating system release version is " << os_version;
    ss << "\n\tThe system has been operating for " << day << "d:" <<
hour << "h:" << min << "m:" << second << "s";
    return ss.str();
}

```

Файл networklist.h:

```

#include <sys/types.h>
#include <cctype>
#include <dirent.h>

```



```

#include <string>
#include <sstream>
#include <vector>
#include "errorcodes.h"
#include "network.h"
class NetworkList
{
    std::vector<Network> network_list;
public:
    NetworkList();
    std::vector<Network> get_network_list() const;
    std::string to_string() const;
};

```

Файл networklist.cpp:

```

#include "networklist.h"
NetworkList::NetworkList()
{
    DIR *d;
    dirent *ent;
    if ((d = opendir(NET_INFO_DIR)) == 0)
    {
        fprintf(stderr, "the %s directory failed to open\n",
NET_INFO_DIR);
        exit(DIR_OPEN_ERROR);
    }
    readdir(d), readdir(d);
    while ((ent = readdir(d)) != 0)
    {
        network_list.emplace_back(ent->d_name);
    }
}
std::vector<Network> NetworkList::get_network_list() const
{
    return network_list;
}
std::string NetworkList::to_string() const
{
    std::stringstream ss;
    ss << "Full network interface description:\n"
    <<
    "*****\n";
    for (auto const& network : network_list)
    {
        ss << network.to_string() << "\n";
    }
    ss
    "*****\n";
    return ss.str();
}

```

Файл network.h:

```

#include <sstream>
#include <string>
#include <cstring>

```

```

#include "errorcodes.h"
#define NET_INFO_DIR "/sys/class/net"
#define NET_INFO_FILE "/sys/class/net/%s/address"

class Network
{
    std::string name;
    std::string mac_address;
public:
    Network(const char *name);
    std::string get_name() const;
    std::string get_mac_address() const;
    std::string to_string() const;
};

```

Файл network.cpp:

```

#include "network.h"
Network::Network(const char *name) : name{ name }
{
    char fname[64];
    sprintf(fname, NET_INFO_FILE, name);
    FILE *f;
    if ((f = fopen(fname, "r")) == 0)
    {
        fprintf(stderr, "file %s failed to open\n", fname);
        exit(FILE_OPEN_ERROR);
    }
    size_t constexpr LINE_BUFFER = 256;
    char line[LINE_BUFFER];
    fgets(line, LINE_BUFFER, f);
    mac_address = strtok(line, "\n");
    fclose(f);
}
std::string Network::get_name() const
{
    return name;
}
std::string Network::get_mac_address() const
{
    return mac_address;
}
std::string Network::to_string() const
{
    std::stringstream ss;
    ss << "Network information for interface " << name;
    ss << "\n\tMac address is " << mac_address;
    return ss.str();
}

```

Файл memory.h:

```

#include <string>
#include <sys/sysinfo.h>
#include <sstream>
#include "errorcodes.h"

class Memory
{
    unsigned long totalram;

```

```

        unsigned long freeram;
        unsigned int mem_unit;
public:
    Memory();
    unsigned long get_totalram() const;
    unsigned long get_freeram() const;
    std::string to_string() const;
};

```

Файл memory.cpp:

```

#include "memory.h"
Memory::Memory()
{
    struct sysinfo s_buf;
    if (sysinfo(&s_buf) != 0)
    {
        fprintf(stderr, "sysinfo function did not return
correctly\n");
        exit(SYS_FUNC_ERROR);
    }
    mem_unit = s_buf.mem_unit;
    totalram = s_buf.totalram;
    freeram = s_buf.freeram;
}
unsigned long Memory::get_totalram() const
{
    return totalram;
}
unsigned long Memory::get_freeram() const
{
    return freeram;
}
std::string Memory::to_string() const
{
    double constexpr MEM_CONV = 1024 * 1024 * 1024;
    std::stringstream ss;
    ss << "Memory information for the computer";
    ss << "\n\tTotal amount of memory is " << (totalram /
static_cast<double>(mem_unit * MEM_CONV)) << " GB";
    ss << "\n\tFree amount of memory is " << (freeram /
static_cast<double>(mem_unit * MEM_CONV)) << " GB";
    ss << "\n\t(Free memory excludes memory used by cached operating
system data)";
    return ss.str();
}

```

Файл hostname.h:

```

#include <string>
#include <sstream>
#include <unistd.h>
#include "errorcodes.h"
class HostName
{
    std::string hostname;
public:
    HostName();
    std::string get_hostname() const;
    std::string to_string() const;
}

```

```
};
```

Файл hostname.cpp:

```
#include "hostname.h"
HostName::HostName()
{
    size_t constexpr NAME_BUFFER = 64;
    char tempname[NAME_BUFFER];
    gethostname(tempname, NAME_BUFFER);
    hostname = tempname;
}
std::string HostName::get_hostname() const
{
    return hostname;
}
std::string HostName::to_string() const
{
    std::stringstream ss;
    ss << "The hostname of the system is " << hostname;
    return ss.str();
}
```

Файл main.cpp:

```
#include <iostream>
#include "utility.h"
int main()
{
    util::print_menu(std::cout, std::cin);
    return 0;
}
```

Файл errorcodes.h:

```
#define DIR_OPEN_ERROR 2
#define FILE_OPEN_ERROR 3
#define SYS_FUNC_ERROR 4
#define INDEX_OUT_OF_BOUNDS 5
```