

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

Военный факультет

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

УТИЛИТА ПО СБОРУ ИНФОРМАЦИИ О СИСТЕМЕ

БГУИР КП 1-40 02 01 109 ПЗ

Студент: группы 830501,
Зверев А. С.

Руководитель: ассистент каф. ЭВМ,
Желтко Ю. Ю.

Минск 2020

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

(подпись)

____ 20 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту _ Звереву Андрею Сергеевичу.

1. Тема проекта Утилита утилита по сбору информации о системе.
2. Срок сдачи студентом законченного проекта 20 мая 2020 г.
3. Исходные данные к проекту Операционная система – Linux, Язык программирования – C++.
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке) 1. Введение. 2. Обзор литературы 3. Структурное проектирование. 4. Функциональное проектирование. 5. Принципиальное проектирование 6. Тестирование программы и руководство пользователя 7. Заключение. Список использованных источников
5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков) 1. Схема структурная. 2. Диаграмма классов
6. Консультант по проекту Желтко Ю.Ю.
7. Дата выдачи задания 22 февраля 2020 г.
8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):
разделы 1,2 к 1 марта 2020 г. – 20 %;
разделы 3,4 к 1 апреля 2020 г. – 30 %;
разделы 5,6,7 к 1 мая 2020 г. – 30 %;
оформление пояснительной записки и графического материала к 20 мая 2020 г. 20 %
Защита курсового проекта с 01 июня 2020 г. по 10 июня 2020 г.

РУКОВОДИТЕЛЬ _____ Ю.Ю. Желтко
(подпись)

Задание принял к исполнению _____ А.С.Зверев
(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1.ОБЗОР ЛИТЕРАТУРЫ	5
2.СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ	7
2.1.Разделение программы на модули	7
2.2.Программные компоненты	8
3.ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ	9
4.РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ	15
4.1. Алгоритм получения информации о жёстком диске	15
4.2. Алгоритм получения информации об ОЗУ	15
4.3. Алгоритм получения информации о процессоре	16
4.4. Алгоритм получения информации об аппаратных средствах	16
5.ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ	17
5.1. Тестирование работоспособности меню	17
6.РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	19
ЗАКЛЮЧЕНИЕ	23
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	24
ПРИЛОЖЕНИЕ А	25
ПРИЛОЖЕНИЕ Б	38

ВВЕДЕНИЕ

Не секрет, что порой возникает необходимость узнать характеристики ПК, какие процессы запущены, список подключенных устройств, а также другую системную информацию. Эта информация хранится в различных файлах и разделах реестра, и выделить из неё то, что нужно, не является слишком простой задачей для обычного пользователя ПК.

С целью упрощения поиска необходимой информации о системе и предоставлении её пользователю в приятном виде была разработана данная утилита.

В данный момент одной из доминирующих операционных систем на ПК является Linux, поэтому в ходе работы было создано решение для этой ОС.

Для операционных систем Linux основными способами получения информации осуществляется путём использования команд терминала и прочтения из системных файлов напрямую.

В качестве языка разработки был выбран язык C++ в связи с его высокой производительности и гибкости этого языка программирования. Среда разработки CLion для Linux была выбрана из-за её удобства.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор существующих аналогов

CPU-Z - программа, которая позволяет узнать корректную информацию об используемом в компьютере оборудовании. Вы узнаете подробную информацию о производителе деталей устройства - начиная от процессора и заканчивая используемой операционной системой.

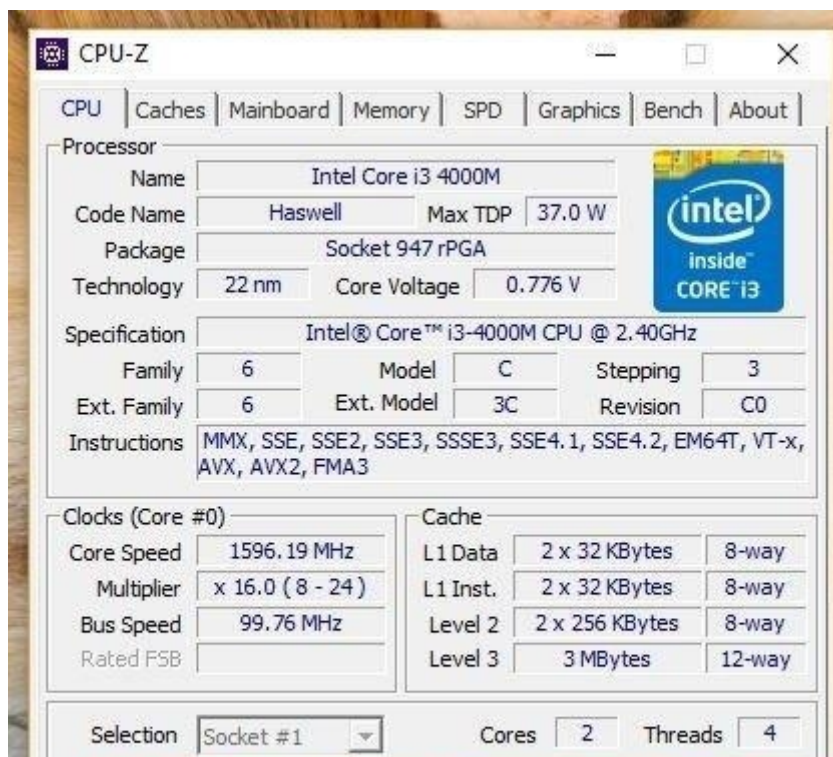


Рисунок 1.1 – CPU-Z

Возможности CPU-Z:

Программа способна выводить подробный отчет об установленной операционной системе и DirectX. Собранные программой сведения укажут на производителя платы, производителя BIOS, версию BIOS, чипсет, графический интерфейс. С помощью данной утилиты можно отслеживать работоспособность внутреннего оборудования компьютерной техники. Тем самым предоставляется возможность своевременного устранения неисправностей и исправления ошибок. Интерфейс самой программы достаточно прост и интуитивно понятен. Среди режимов работы программы встречается скрытый вариант. Тем самым программа будет просто выдавать готовый отчет по используемому оборудованию, при этом не занимая пространства рабочего стола.

SysInfo Detector - бесплатная утилита для получения информации о компонентах вашего персонального компьютера с детализированными отчетами. Программа совместима со всеми версиями операционной системы Windows.

Основные возможности SysInfo Detector:

Подробные сведения об аппаратной конфигурации компьютера, на котором установлена и используется программа. Предоставляются данные обо всем «железе», установленном в ПК, начиная со сведений о материнской плате и заканчивая всеми «вставленными» в USB-порты устройствами. Информация включает наименование производителя, модель, серийный номер и другие данные. Детектирование моделей подключенных устройств. Модели устройств PCI/PCI-X/AGP/USB/PCI-E и мониторов определяются по справочникам. Серийные номера. Все оборудование имеет серийные номера, информацию о которых способна получать и отображать программа. Это модули ОЗУ, материнские платы, жесткие диски, устройства USB, процессоры, мониторы, а также другое оборудование. Все обнаруженные устройства USB дополнительно разделяются по типам. Все оборудование, подключенное к USB, делится на классы (например, мыши, клавиатуры, принтеры, МФУ). Программа определяет, к какому из классов относится то или иное устройство, и соответствующим образом группирует их. Если для устройства не смог быть определен класс, оно получает класс «устройство USB». Информация о неполадках в работе ПК. Программа выводит информацию об обнаруженных ею проблемах и даже способна выявить потенциальные проблемы. Какое установлено ПО. SysInfo Detector формирует и выводит перечень всех установленных на данный момент приложений в операционной системе. Это позволяет осуществлять контроль за используемым на рабочих местах программным обеспечением. Запущенные процессы, настройки ОС и лицензии на программы. Определяются данные о лицензионном использовании программ и основные параметры ОС. Мониторинг системных датчиков. Программа умеет считывать информацию, показывающую скорости вращения вентиляторов, напряжения питания, температуры компонентов. Пользователь может создавать исправленные варианты информации об устройствах и высылать их разработчикам.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После определения требований к функциональности разрабатываемой системы ее следует разбить на функциональные блоки. Такой подход упростит понимание системы, позволит устранить проблемы в архитектуре, обеспечит гибкость и масштабируемость системы в будущем путем добавления новых блоков.

2.1.Разделение программы на модули

Разделение исходного текста программы на несколько файлов становится необходимым по многим причинам:

1. С большим текстом просто неудобно работать.
- 2.Разделение программы на отдельные модули, которые решают конкретные подзадачи.
- 3.Разделение программы на отдельные модули, с целью повторного использования этих модулей в других программах.
- 4.Разделение интерфейса и реализации.

Я намеренно использовал слово «модуль», поскольку модулем может быть как класс, так и набор функций — вопрос используемой технологии программирования.

Как только мы решаем разделить исходный текст программы на несколько файлов, возникают две проблемы:

1. Необходимо от простой компиляции программы перейти к раздельной. Для этого надо внести соответствующие изменения либо в последовательность действий при построении приложения вручную, либо внести изменения в командные или make-файлы, автоматизирующие процесс построения, либо внести изменения в проект IDE.
2. Необходимо решить каким образом разбить текст программы на отдельные файлы.

Первая проблема — чисто техническая. Она решается посредством использования в проекте CLion.

Вторая проблема — требует гораздо более творческого подхода. Хотя и здесь существуют определённые рекомендации, несоблюдение которых приводит либо к невозможности собрать проект, либо к трудностям в дальнейшем развитии проекта.

Во-первых, нужно определить какие части программы выделить в отдельные модули. Что бы это получилось просто и естественно, программа должна быть правильно спроектирована. Вся программа должна состоять из слабо связанных фрагментов. Тогда каждый такой фрагмент может быть естественным образом преобразован в отдельный модуль (единицу компиляции). Под «фрагментом» подразумевается не просто произвольный кусок кода, а функция, или группа логически связанных функций, или класс, или несколько тесно взаимодействующих классов.

Во-вторых, нужно определить интерфейсы для модулей. Когда часть программы выделяется в модуль (единицу компиляции), остальной части

программы (а если быть точным, то компилятору, который будет обрабатывать остальную часть программы) надо каким-то образом объяснить что имеется в этом модуле. Для этого служат *заголовочные файлы*. Таким образом, модуль состоит из двух файлов: заголовочного (интерфейс) и файла реализации.

Благодаря разделению программы на части, разработчик может беспрепятственно дополнять свой проект новыми возможностями и, соответственно, тестировать их, не прибегая к работе во всем проекте сразу.

2.2. Программные компоненты

Следует отметить что код программы будет работать исключительно только в ОС Linux, т. к. все его возможности основаны на командах терминала и парсинге папки /proc.

Данная утилита по сбору информации о системе предположительно разделяется на восемь классов по сбору информации о системе, одним классом для реализации меню, одним классом для хранения информации о системе в файлах и одним классом для вывода информации на экран.

Все классы находятся в отдельных файлах для удобства пользования и упрощения тестирования отдельных модулей программы, а также для облегчения поиска ошибок. Сами классы мы компонуем в один проект с помощью CLion, но также это можно сделать с помощью QtCreator, так как в их основе на ОС Linux лежит утилита CMake, которая как раз и служит для компоновки многофайловых проектов.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Разработанная утилита по сбору информации о системе должна обладать следующими функциями:

1. Отображение информации об аппаратных средствах.
2. Отображение информации о процессоре.
3. Отображение системной информации.
4. Отображение информации о жёстком диске.
5. Отображение информации об устройствах и видеокарте.
6. Отображение информации об ОЗУ.

Рассмотрим несколько функций реализованных в данном курсовом проекте.

Получение информации об аппаратных средствах. Данную функцию выполняет метод getInfo() класса HardwareInfo. Данный метод вызывает другие методы этого класса: getUsb(), getNet(), getBios(), getSystemInfo(), getAudio(). Данные методы отображают информацию о портах USB, устройствах для подключения к сети, информации о BIOS, общей системной информации, и информации об аудиоустройствах соответственно. Каждый из методов отправляет определённую команду в терминал:

```
void HardwareInfo::getUsb(){
    system("lsusb");
    std::cout << std::endl;
    cout << "ДЛЯ ВЫХОДА ВВЕДИ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    system("lsusb -vt");}
void HardwareInfo::getNet(){
    system("clear");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    system("iwconfig");
    getchar();}
void HardwareInfo::getAudio(){
    system("arecord -l");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}
void HardwareInfo::getSystemInfo(){
    system("sudo dmidecode -t system");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HardwareInfo::getBios(){
    system("sudo dmidecode -t bios");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HardwareInfo::getInfo(){
    HardwareInfo temp_object;
```

```

cout << "1. Аудио" << endl;
cout << "2. BIOS" << endl;
cout << "3. LAN" << endl;
cout << "4. Системная информация" << endl;
cout << "5. USB" << endl;
char a;
cin >> a;
switch(a){
case '1':{
    temp_object.getAudio();
    break;
}
case '2':{
    temp_object.getBios();
    break;
}
case '3':{
    temp_object.getNet();
    break;
}
case '4':{
    temp_object.getSystemInfo();
    break;
}
case '5':{
    temp_object.getUsb();
    break;
}
default:{
    return;
}}
}
}
}

```

Получение информации о процессоре осуществляется методом getInfo() класса CpuInfo, который в свою очередь открывает файл через терминал в котором храниться информация о процессоре:

```

void CpuInfo::getInfo(){
    system("cat /proc/cpuinfo");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

```

Получение системной информации осуществляется методом getInfo() класса SystemInfo, который в свою очередь вызывает другие методы этого класса: getHostname(), getSystem(), getManufacture(), getProductName(), getFullInfo(), getUptime(), getLoad(). Данные методы отправляют определённые команды в терминал для получения информации об имени компьютера,

системной информации, производителе, названии изделия, о системе целиком, о пользователях и о загрузженности системы соответственно:

```
void SystemInfo::getHostname(){
    system("hostname");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getSystem(){
    system("uname -a");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getManufacturer(){
    system("sudo dmidecode -s system-manufacturer");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getProductName()
{
    system("sudo dmidecode -s system-product-name");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getFullInfo(){
    system("sudo dmidecode | more");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getUptime(){
    system("w");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getLoad(){
    system("cat /proc/loadavg");
    system("sudo top");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getInfo(){
    SystemInfo temp_object;
    cout << "1. Имя компьютера и DNS" << endl;
    cout << "2. Системная информация" << endl;
    cout << "3. Производитель" << endl;
    cout << "4. Название изделия" << endl;
    cout << "5. Информация о системе целиком" << endl;
    cout << "6. Информация о пользователях" << endl;
    cout << "7. Загруженность системы" << endl;
```

```

char a;
cin >> a;
switch(a){
case '1':{
    temp_object.getHostname();
    break;
}
case '2':{
    temp_object.getSystem();
    break;
}
case '3':{
    temp_object.getManufacturer();
    break;
}
case '4':{
    temp_object.getProductName();
    break;
}
case '5':{
    temp_object.getFullInfo();
    break;
}
case '6':{
    temp_object.getUptime();
    break;
}
case '7':{
    temp_object.getLoad();
    break;
}
default:{
    break;
    return;
}
}
}

```

Получение информации о жёстком диске осуществляется с помощью метода getInfo() класса HddInfo который вызывает другие методы этого класса: getHddInfo(), getFreespaceInfo(), getHddBlockInfo(). Данные методы в свою очередь отображают информацию о жёстком диске, свободном пространстве, ии и о блочных устройствах:

```

#include "../Headers/HddInfo.h"
HddInfo::HddInfo() {}
void HddInfo::getHddInfo(){
    system("sudo fdisk -l | grep '^Disk /dev'");
    system("sudo fdisk -l /dev/sda");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

```

```

}

void HddInfo::getHddBlockInfo(){

    system("lsblk");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HddInfo::getFreespaceInfo(){
    system("df -H");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HddInfo::getInfo(){
    HddInfo temp_object;
    cout << "1. Информация о жёстком диске" << endl;
    cout << "2. Свободное пространство" << endl;
    cout << "3. Инфо о блочных устройствах" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':{ temp_object.getHddInfo(); }
    case '2':{ temp_object.getFreespaceInfo();break;}
    case '3':{ temp_object.getHddBlockInfo(); break;}
    default:{ return;}
    }
}

```

Получение информации об устройствах и видеокарте осуществляется с помощью с помощью метода getInfo() класса PciInfo который в свою очередь отображает необходимую пользователю информацию:

```

#include "../Headers/PciInfo.h"
PciInfo::PciInfo(){}
void PciInfo::getPciInfo(){system("lspci");}
void PciInfo::getGpuInfo(){system("lspci | grep -i vga");}
void PciInfo::getNvidiaInfo(){system("nvidia-smi");
system(" nvidia-settings");}
void PciInfo::getRadeonInfo(){system("fglrxinfo");}
void PciInfo::getInfo()
{
    PciInfo temp_object;
    cout << "1. Информация об устройствах" << endl;
    cout << "2. Информация о видеокарте" << endl;
    cout << "3. NVidia GPU" << endl;
    cout << "4. Radeon GPU" << endl;
    char a;
    cin >> a;
    switch(a){

```

```
case '1':{
    temp_object.getPciInfo();
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    break;
}
case '2':{
    temp_object.getGpuInfo();
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    break;
}
case '3':{
    temp_object.getNvidiaInfo();
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    break;
}
case '4':{
    temp_object.getRadeonInfo();
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    break;
}
default:{
    return;
}
}
```

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данном разделе будут рассмотрены схема алгоритмы, используемые в программе.

4.1 Алгоритм получения информации о жёстком диске

Для данного алгоритма предназначен метод `getInfo` класса `HddInfo`.

1. Начало

2. Входные данные: отсутствуют.

Выходные данные: отсутствуют.

Промежуточные данные: переменная для хранения выбора пользователя в меню, объект класса `HddInfo`.

3. Реализация с помощью `switch-case` консольного меню для того, чтобы пользователь мог сам выбрать ту информацию, которую он желает посмотреть.

4. Обработка выбора пользователя и отправка соответствующей команды в терминал для отображения информации, которую выбрал пользователь.

5. Ожидание от пользователя нажатия любой клавиши для завершения работы этого метода и возврата в главное меню.

6. Конец.

4.2 Алгоритм получения информации об ОЗУ

Для данного алгоритма предназначен метод `getInfo` класса `MemoryInfo`.

1. Начало

2. Входные данные: отсутствуют.

Выходные данные: отсутствуют.

Промежуточные данные: символьная переменная для хранения выбора пользователя и объект класса `MemoryInfo`.

3. Реализация с помощью `switch-case` консольного меню для того, чтобы пользователь мог сам выбрать ту информацию, которую он желает посмотреть.

4. Обработка выбора пользователя и отправка соответствующей команды в терминал для отображения информации, которую выбрал пользователь.

5. Ожидание от пользователя нажатия любой клавиши для завершения работы этого метода и возврата в главное меню.

6. Конец.

4.3 Алгоритм получения информации о процессоре

Для данного алгоритма предназначен метод `getInfo` класса `CpuInfo`.

1. Начало
2. Входные данные: отсутствуют.

Выходные данные: отсутствуют.

Промежуточные данные: объект класса `CpuInfo`.

3. Отправка соответствующей команды в терминал для отображения информации, которую выбрал пользователь.
4. Ожидание от пользователя нажатия любой клавиши для завершения работы этого метода и возврата в главное меню.
5. Конец.

4.4 Алгоритм получения информации об аппаратных средствах

Для данного алгоритма предназначен метод `getInfo` класса `HardwareInfo`.

1. Начало
2. Входные данные: отсутствуют.

Выходные данные: отсутствуют.

Промежуточные данные: символьная переменная для хранения выбора пользователя и объект класса `HardwareInfo`.

3. Реализация с помощью `switch-case` консольного меню для того, чтобы пользователь мог сам выбрать ту информацию, которую он желает посмотреть. .
4. Обработка выбора пользователя и отправка соответствующей команды в терминал для отображения информации, которую выбрал пользователь.
5. Ожидание от пользователя нажатия любой клавиши для завершения работы этого метода и возврата в главное меню.
6. Конец.

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

В процессе разработки приложения было проведено тестирование приложения в различных сценариях использования.

5.1 Тестирование работоспособности меню.

В данном подразделе демонстрируются все варианты исходов выбора конкретного пункта меню:

1. Выбор несуществующего пункта меню:

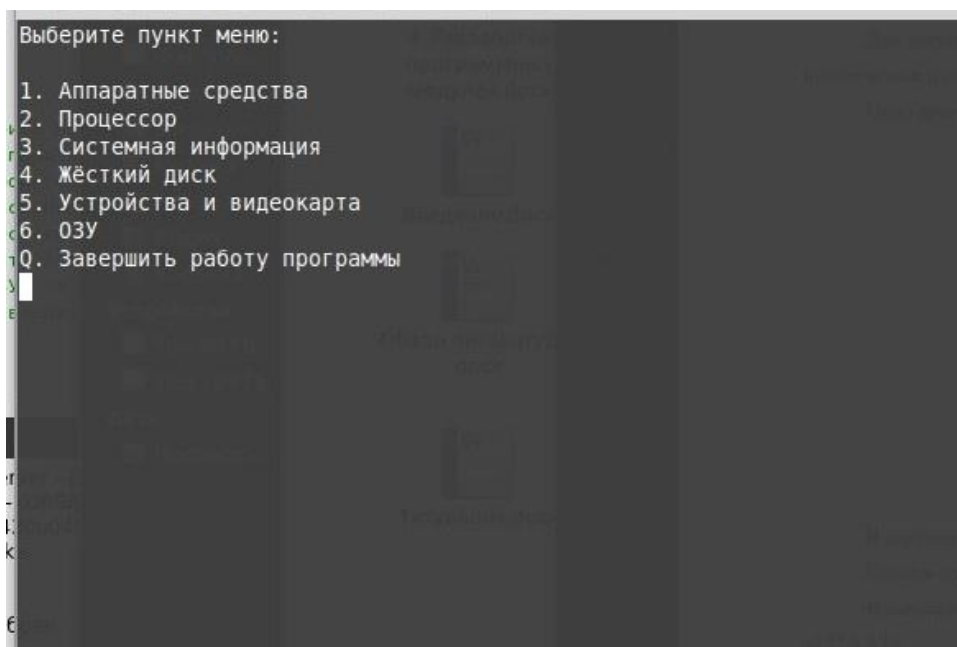


Рисунок 5.1.1 – Ввод некорректного пункта меню

2. Выбор определённого пункта меню:

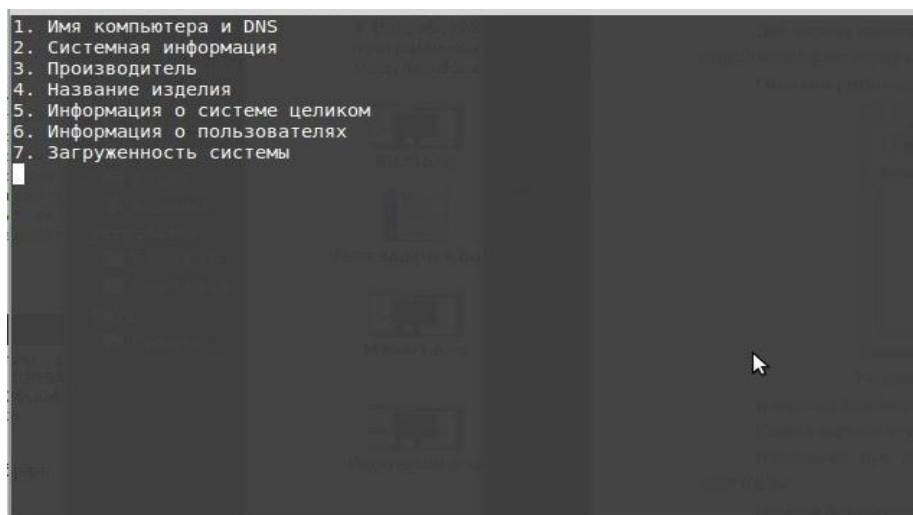


Рисунок 5.1.2 – Выбор определённого пункта меню

3. Загруженность системы:

```
top - 10:52:31 up 52 min, 1 user, load average: 0,83, 0,70, 0,63
Tasks: 176 total, 1 running, 134 sleeping, 0 stopped, 1 zombie
%Cpu(s): 15,9 us, 4,5 sy, 0,0 ni, 75,9 id, 3,6 wa, 0,0 hi, 0,2 si, 0,0 st
KiB Mem : 3917100 total, 957880 free, 1565796 used, 1393424 buff/cache
KiB Swap: 3906556 total, 3906556 free, 0 used. 1977296 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2199	angel	20	0	3149104	576468	117924	S	11,8	14,7	8:44.32	Web Content
4290	root	20	0	45360	4260	3632	R	11,8	0,1	0:00.04	top
1968	angel	20	0	3156880	379648	160612	S	5,9	9,7	4:38.26	firefox
1	root	20	0	160168	9292	6616	S	0,0	0,2	0:02.73	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:+
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_+
9	root	20	0	0	0	0	S	0,0	0,0	0:00.00	ksoftirqd/0
10	root	20	0	0	0	0	I	0,0	0,0	0:02.69	rcu_sched
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.04	migration/0
13	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_injec+
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/1
16	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_injec+
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.05	migration/1

Рисунок 5.1.3 – Загруженность системы

4. Возврат в главное меню:

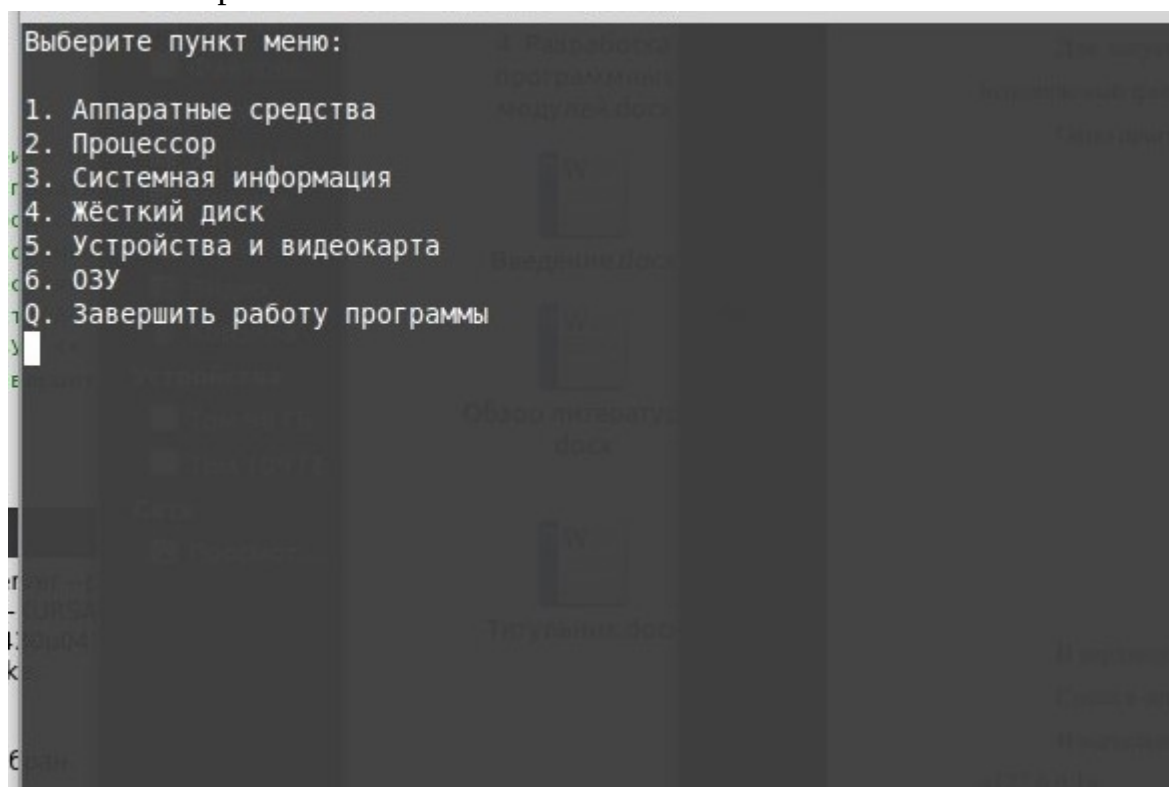


Рисунок 5.1.4 – Возврат в главное меню

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска приложения необходимо запустить исполняемый файл в котором откроется меню для выбора информации которую пользователь хочет посмотреть.

Окно программы с меню представлено на рисунке 6.1

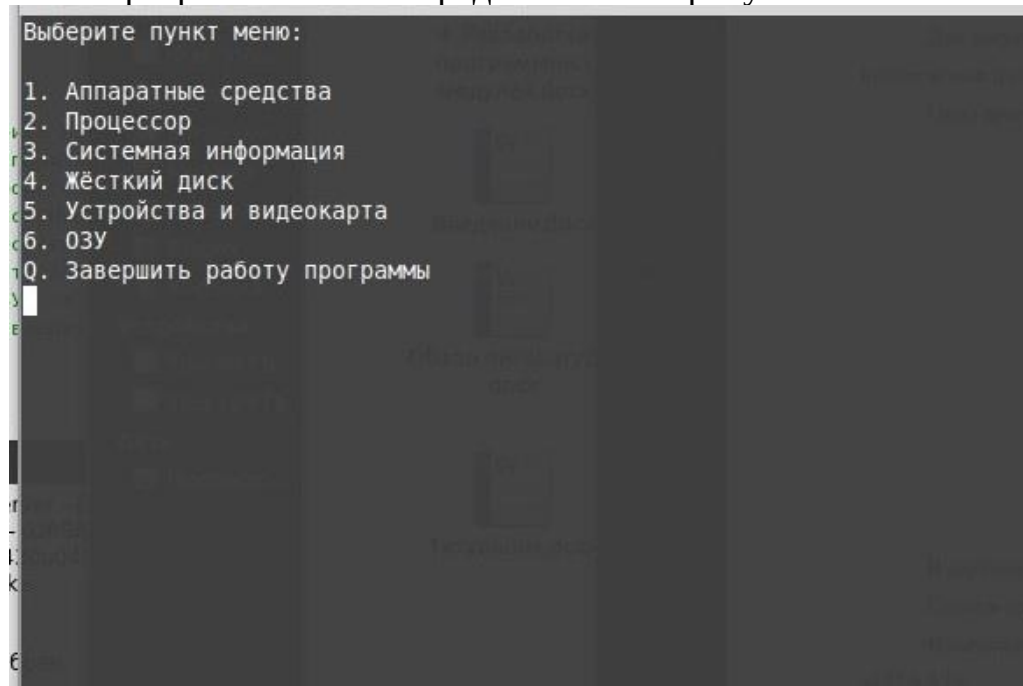


Рисунок 6.1. Окно программы с меню

В данном меню пользователь вводит с клавиатуры значение пункта меню и затем открывается или новое меню или информация.

На рисунке 6.2 представлено меню при выборе аппаратных средств

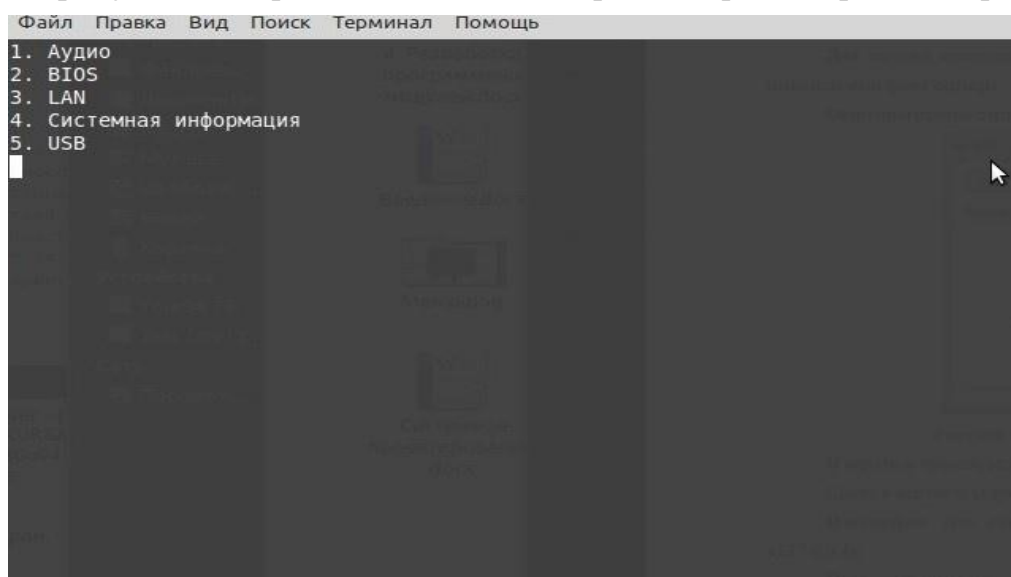


Рисунок 6.2 Отображение меню аппаратных средств

После ввода любого символа мы возвращаемся в главное меню. На рисунке 6.3 изображено действие программы при выборе пункта 2 в главном меню.

```

core id      : 1
cpu cores   : 2
apicid      : 2
initial apicid : 2
fpu         : yes
fpu_exception : yes
cpuid level : 11
wp          : yes
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm c
onstant_tsc arch_perfmon pebs bts rep_good nopl xtopology tsc_reliable nonstop_t
sc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est t
m2 ssse3 cx16 xtpr pdcm sse4_1 sse4_2 movbe popcnt tsc_deadline_timer rdrand lah
f_lm 3dnowprefetch epb pti tpr_shadow vnmi flexpriority ept vpid tsc_adjust smep
erms dtherm ida arat
bugs        : cpu_meltdown spectre_v1 spectre_v2 mds msbds_only
bogomips    : 4331.60
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:

для выхода ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!

```

Рисунок 6.3. Информация о процессоре.

На рисунке 6.4 изображено меню которое появляется при выборе пункта системная информация в главном меню.

```

1. Имя компьютера и DNS
2. Системная информация
3. Производитель
4. Название изделия
5. Информация о системе целиком
6. Информация о пользователях
7. Загруженность системы

```

Рисунок 6.4 Меню системной информации.

Результат выбора загруженность системы, а именно ввод числа 7 с клавиатуры представлен на рисунке 6.5.

```
top - 10:52:31 up 52 min, 1 user, load average: 0,83, 0,70, 0,63
Tasks: 176 total, 1 running, 134 sleeping, 0 stopped, 1 zombie
%Cpu(s): 15,9 us, 4,5 sy, 0,0 ni, 75,9 id, 3,6 wa, 0,0 hi, 0,2 si, 0,0 st
KiB Mem : 3917100 total, 957880 free, 1565796 used, 1393424 buff/cache
KiB Swap: 3906556 total, 3906556 free, 0 used. 1977296 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2199	angel	20	0	3149104	576468	117924	S	11,8	14,7	8:44.32	Web Content
4290	root	20	0	45360	4260	3632	R	11,8	0,1	0:00.04	top
1968	angel	20	0	3156880	379648	160612	S	5,9	9,7	4:38.26	firefox
1	root	20	0	160168	9292	6616	S	0,0	0,2	0:02.73	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:+
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu +
9	root	20	0	0	0	0	S	0,0	0,0	0:00.20	ksoftirqd/0
10	root	20	0	0	0	0	I	0,0	0,0	0:02.69	rcu_sched
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.04	migration/0
13	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_injec+
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/1
16	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_injec+
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.05	migration/1

Рисунок 6.5 Загруженность системы.

Так эта информация является постоянно изменяющейся то для выхода из неё нам необходимо нажать сочетание клавиш Ctrl+C затем ввести любой символ и у нас снова откроется главное меню представленное на рисунке 6.6.

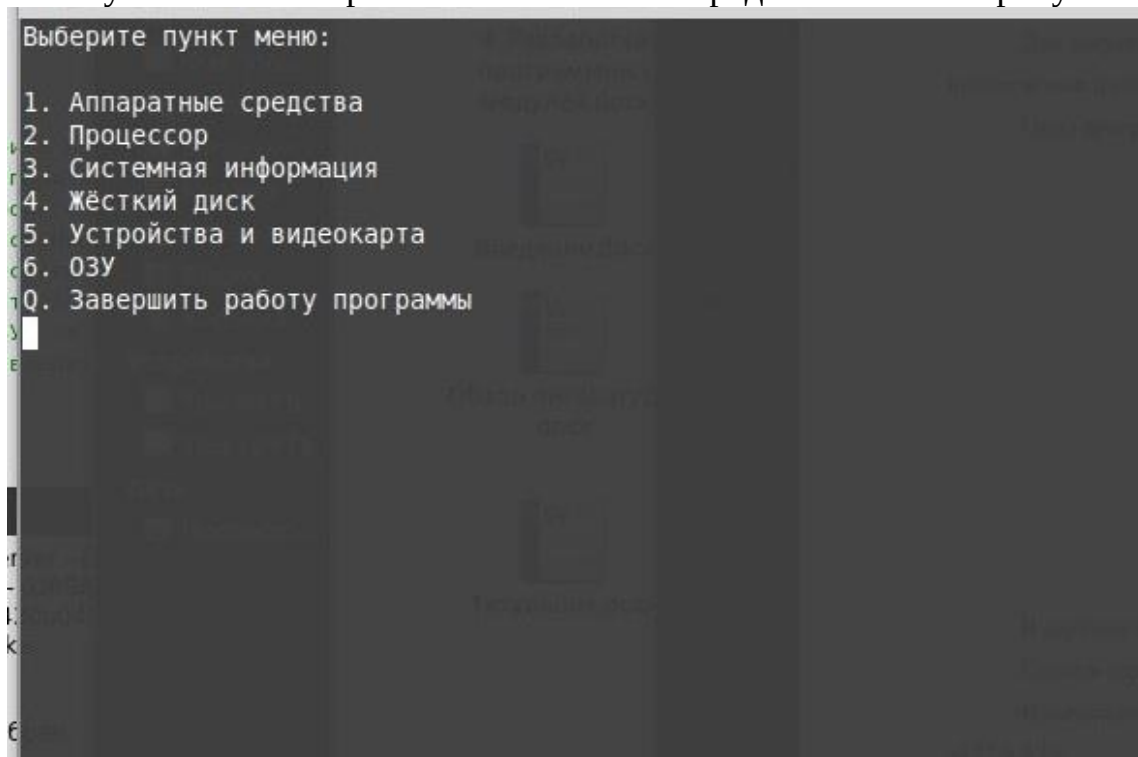


Рисунок 6.6 Меню, которое появляется после просмотра любой информации.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была осуществлена разработка и документирование утилиты по сбору информации о системе Linux. Данная утилита позволяет пользователю без особого труда найти нужную ему информацию о системе, а также посмотреть запущенные процессы и USB устройства подключенные на данный момент к ПК.

В процессе работы были получены, а также закреплены, знания и опыт в объектно-ориентированном программировании и создании приложений при помощи среды разработки CLion под операционную систему Linux. Также изучены новые функциональные возможности операционной системы Linux.

К достоинствам программы можно отнести простой, интуитивно понятный интерфейс, высокую скорость работы даже на самых слабых компьютерах и модернизацию программы.

В дальнейших перспективах планируется добавление нового функционала и усовершенствование старого.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Эффективное программирование на C++. Практическое программирование на примерах/ Кёниг Эндрю, Му Барбара М. :Издательский дом "Вильямс", 2002. — 368 с.
- [2] Программирование Realcoding.Net [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.realcoding.net/>
- [3] RTFM: Linux, DevOps и системное администрирование [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://rtfm.co.ua/c-sokety-i-primer-modeli-client-server/>

ПРИЛОЖЕНИЕ А

Листинг кода

Main.cpp

```
#include <iostream>
#include <fstream>
#include <pwd.h>
#include <unistd.h>
#include <sys/sysinfo.h>
#include <vector>
#include <locale>
#include <cstdlib>

#include "Headers/CpuInfo.h"
#include "Headers/SystemInfo.h"
#include "Headers/PciInfo.h"
#include "Headers/HardwareInfo.h"
#include "Headers/HddInfo.h"
#include "Headers/MemoryInfo.h"

using namespace std;

int main()
{
    setlocale (LC_ALL, "RUS");

    while(1)
    {
        cout << "Выберите пункт меню:" << endl << endl;
        cout << "1. Аппаратные средства" << endl;
        cout << "2. Процессор" << endl;
        cout << "3. Системная информация" << endl;
        cout << "4. Жёсткий диск" << endl;
        cout << "5. Устройства и видеокарта" << endl;
        cout << "6. ОЗУ" << endl;
        cout << "Q. Завершить работу программы"<<endl;
        char a;
        cin >> a;
        switch(a){
            case '1':{
                HardwareInfo hardwareInfo;
                system("clear");
                hardwareInfo.getInfo();
                cin.ignore();
                cin.get();
                system("clear");
                break;
            }
            case '2':{
                CpuInfo cpuInfo;
                system("clear");
```



```

        cpuInfo.getInfo();
        cin.ignore();
        cin.get();
        system("clear");
        break;
    }
    case '3':{
        SystemInfo systemInfo;
        system("clear");
        systemInfo.getInfo();
        cin.ignore();
        cin.get();
        system("clear");
        break;
    }
    case '4':{
        HddInfo hddInfo;
        system("clear");
        hddInfo.getInfo();
        cin.ignore();
        cin.get();
        system("clear");
        break;
    }
    case '5':{
        PciInfo pciInfo;
        system("clear");
        pciInfo.getInfo();
        cin.ignore();
        cin.get();
        system("clear");
        break;
    }
    case '6':{
        MemoryInfo memoryInfo;
        system("clear");
        memoryInfo.getInfo();
        cin.ignore();
        cin.get();
        system("clear");
        break;
    }
    case 'Q':{return 1;break;}}return 0;}

```

ComputerPart.cpp

```

#include "../Headers/ComputerPart.h"
void ComputerPart :: getInfo(){}

```

CpuInfo.cpp

```

#include "../Headers/CpuInfo.h"
CpuInfo::CpuInfo(){}

```

```

void CpuInfo::getInfo(){
system("cat /proc/cpuinfo");
cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;}

HardwareInfo.cpp

#include "../Headers/HardwareInfo.h"

using namespace std;

HardwareInfo::HardwareInfo()
{
}

void HardwareInfo::getUsb()
{
    system("lsusb");
    std::cout << std::endl;
    cout << "ДЛЯ ВЫХОДА ВВЕДИ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    system("lsusb -vt");
}

void HardwareInfo::getNet()
{
    system("clear");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    system("iwconfig");
    getchar();
    //system("clear");
    //system("watch -n 1 cat /proc/net/wireless");
}

void HardwareInfo::getAudio()
{
    system("arecord -l");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HardwareInfo::getSystemInfo()
{
    system("sudo dmidecode -t system");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HardwareInfo::getBios()
{
    system("sudo dmidecode -t bios");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

```

```

void HardwareInfo::getInfo()
{
    HardwareInfo temp_object;
    cout << "1. Аудио" << endl;
    cout << "2. BIOS" << endl;
    cout << "3. LAN" << endl;
    cout << "4. Системная информация" << endl;
    cout << "5. USB" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':{
        temp_object.getAudio();
        break;
    }
    case '2':{
        temp_object.getBios();
        break;
    }
    case '3':{
        temp_object.getNet();
        break;
    }
    case '4':{
        temp_object.getSystemInfo();
        break;
    }
    case '5':{
        temp_object.getUsb();
        break;
    }
    default:{return;}}}}

```

HddInfo.cpp

```

#include "../Headers/HddInfo.h"
HddInfo::HddInfo() {
}

void HddInfo::getHddInfo()
{
    system("sudo fdisk -l | grep '^Disk /dev'");
    system("sudo fdisk -l /dev/sda");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HddInfo::getHddBlockInfo()
{
    system("lsblk");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HddInfo::getFreespaceInfo()

```

```

{
    system("df -H");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void HddInfo::getInfo()
{
    HddInfo temp_object;
    cout << "1. Информация о жёстком диске" << endl;
    cout << "2. Свободное пространство" << endl;
    cout << "3. Инфо о блочных устройствах" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':{
        temp_object.getHddInfo();
    }
    case '2':{
        temp_object.getFreespaceInfo();
        break;
    }
    case '3':{
        temp_object.getHddBlockInfo();
        break;
    }
    default:{
        return;
    }
    }
}

```

MemoryInfo.cpp

```

#include "../Headers/MemoryInfo.h"
MemoryInfo::MemoryInfo()
{

}

void MemoryInfo::freeMemory()
{
    system("less /proc/meminfo");
}

void MemoryInfo::freeMemorym()
{
    system("free -m");
}

void MemoryInfo::freeMemoryg()
{
    system("free -g");
}

```

```

void MemoryInfo::getInfo()
{
    MemoryInfo temp_object;
    cout << "1. Свободно памяти (МБ)" << endl;
    cout << "2. Свободно памяти (ГБ)" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':{
        temp_object.freeMemorym();
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
        break;
    }
    case '2':{
        temp_object.freeMemoryg();
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
        break;
    }
    default:{
        return;
    }
    }
}

```

PciInfo.cpp

```

#include "../Headers/PciInfo.h"

PciInfo::PciInfo()
{
}

void PciInfo::getPciInfo()
{
    system("lspci");
}

void PciInfo::getGpuInfo()
{
    system("lspci | grep -i vga");
}

void PciInfo::getNvidiaInfo()
{
    system("nvidia-smi");
    system(" nvidia-settings");
}

void PciInfo::getRadeonInfo()
{
    system("fglrxinfo");
}

```

```

void PciInfo::getInfo()
{
    PciInfo temp_object;
    cout << "1. Информация об устройствах" << endl;
    cout << "2. Информация о видеокарте" << endl;
    cout << "3. NVidia GPU" << endl;
    cout << "4. Radeon GPU" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':{
        temp_object.getPciInfo();
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
        break;
    }
    case '2':{
        temp_object.getGpuInfo();
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
        break;
    }
    case '3':{
        temp_object.getNvidiaInfo();
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
        break;
    }
    case '4':{
        temp_object.getRadeonInfo();
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
        break;
    }
    default:{
        return;
    }
    }
}

```

SystemInfo.cpp

```

#include "../Headers/SystemInfo.h"

SystemInfo::SystemInfo()
{

}

void SystemInfo::getHostname()
{
    system("hostname");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getSystem()
{

```

```

        system("uname -a");
        cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
    }

void SystemInfo::getManufacturer()
{
    system("sudo dmidecode -s system-manufacturer");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getProductName()
{
    system("sudo dmidecode -s system-product-name");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getFullInfo()
{
    system("sudo dmidecode | more");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getUptime()
{
    system("w");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getLoad()
{
    system("cat /proc/loadavg");
    system("sudo top");
    cout << "ДЛЯ ВЫХОДА ВВЕДИТЕ ЛЮБОЕ ЗНАЧЕНИЕ!!!"<<endl;
}

void SystemInfo::getInfo()
{
    SystemInfo temp_object;
    cout << "1. Имя компьютера и DNS" << endl;
    cout << "2. Системная информация" << endl;
    cout << "3. Производитель" << endl;
    cout << "4. Название изделия" << endl;
    cout << "5. Информация о системе целиком" << endl;
    cout << "6. Информация о пользователях" << endl;
    cout << "7. Загруженность системы" << endl;
    char a;
    cin >> a;
    switch(a){
    case '1':{
        temp_object.getHostname();
        break;
    }
    case '2':{

```

```

        temp_object.getSystem();
        break;
    }
    case '3':{
        temp_object.getManufacturer();
        break;
    }
    case '4':{
        temp_object.getProductName();
        break;
    }
    case '5':{
        temp_object.getFullInfo();
        break;
    }
    case '6':{
        temp_object.getUptime();
        break;
    }
    case '7':{
        temp_object.getLoad();
        break;
    }
    default:{
        break;
        return;
    }
}
}
}

```

ComputerPart.h

```

#ifndef COMPUTERPART_H

#define COMPUTERPART_H

class ComputerPart
{
public:
    void getInfo();
};

#endif // COMPUTERPART_H
CpuInfo.h

```

```

#ifndef CPUINFO_H

#define CPUINFO_H

#include "ComputerPart.h"

#include <iostream>

```



```

#include <fstream>
#include <pwd.h>
#include <unistd.h>
#include <sys/sysinfo.h>
#include <vector>
#include <cstdlib>

using namespace std;

class CpuInfo : protected ComputerPart
{
public:
    CpuInfo();
    void getInfo();
};

#endif // CPUINFO_H

```

HardwareInfo.h

```

#ifndef HARDWAREINFO_H

#define HARDWAREINFO_H

#include "ComputerPart.h"

#include <iostream>
#include <fstream>
#include <pwd.h>
#include <unistd.h>
#include <sys/sysinfo.h>
#include <vector>
#include <cstdlib>

class HardwareInfo : protected ComputerPart
{
public:
    HardwareInfo();
    void getUsb();
    void getNet();
    void getAudio();
    void getSystemInfo();
    void getBios();
    void getInfo();
};

#endif // HARDWAREINFO_H

```

HddInfo.h

```

#ifndef HDDINFO_H

#define HDINFOD_H

#include "ComputerPart.h"

#include <iostream>
#include <fstream>
#include <pwd.h>
#include <unistd.h>
#include <sys/sysinfo.h>
#include <vector>
#include <cstdlib>

using namespace std;

class HddInfo : protected ComputerPart
{
public:
    HddInfo();
    void getHddInfo();
    void getHddBlockInfo();
    void getFreespaceInfo();
    void getInfo();
};#endif // HDDINFO_H

```

MemoryInfo.h

```

#ifndef MEMORYINFO_H

#define MEMORYINFO_H

#include "ComputerPart.h"

#include <iostream>
#include <fstream>
#include <pwd.h>
#include <unistd.h>
#include <sys/sysinfo.h>
#include <vector>
#include <cstdlib>

using namespace std;

class MemoryInfo : protected ComputerPart
{
public:
    MemoryInfo();
    void freeMemory();
    void freeMemorym();
    void freeMemoryg();
    void getInfo();

```

```
};
```

```
#endif // MEMORYINFO_H
```

PciInfo.h

```
#ifndef PCIINFO_H
```

```
#define PCIINFO_H
```

```
#include "ComputerPart.h"
```

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <pwd.h>
```

```
#include <unistd.h>
```

```
#include <sys/sysinfo.h>
```

```
#include <vector>
```

```
#include <cstdlib>
```

```
using namespace std;
```

```
class PciInfo : protected ComputerPart
```

```
{
```

```
public:
```

```
    PciInfo();
```

```
    void getPciInfo();
```

```
    void getGpuInfo();
```

```
    void getNvidiaInfo();
```

```
    void getRadeonInfo();
```

```
    void getInfo();
```

```
};
```

```
#endif // PCIINFO_H
```

SystemInfo.h

```
#ifndef SYSTEMINF_H
```

```
#define SYSTEMINF_H
```

```
#include "ComputerPart.h"
```

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <pwd.h>
```

```
#include <unistd.h>
```

```
#include <sys/sysinfo.h>
```

```
#include <vector>
```

```
#include <cstdlib>
```

```
using namespace std;

class SystemInfo : protected ComputerPart
{
public:
    SystemInfo();

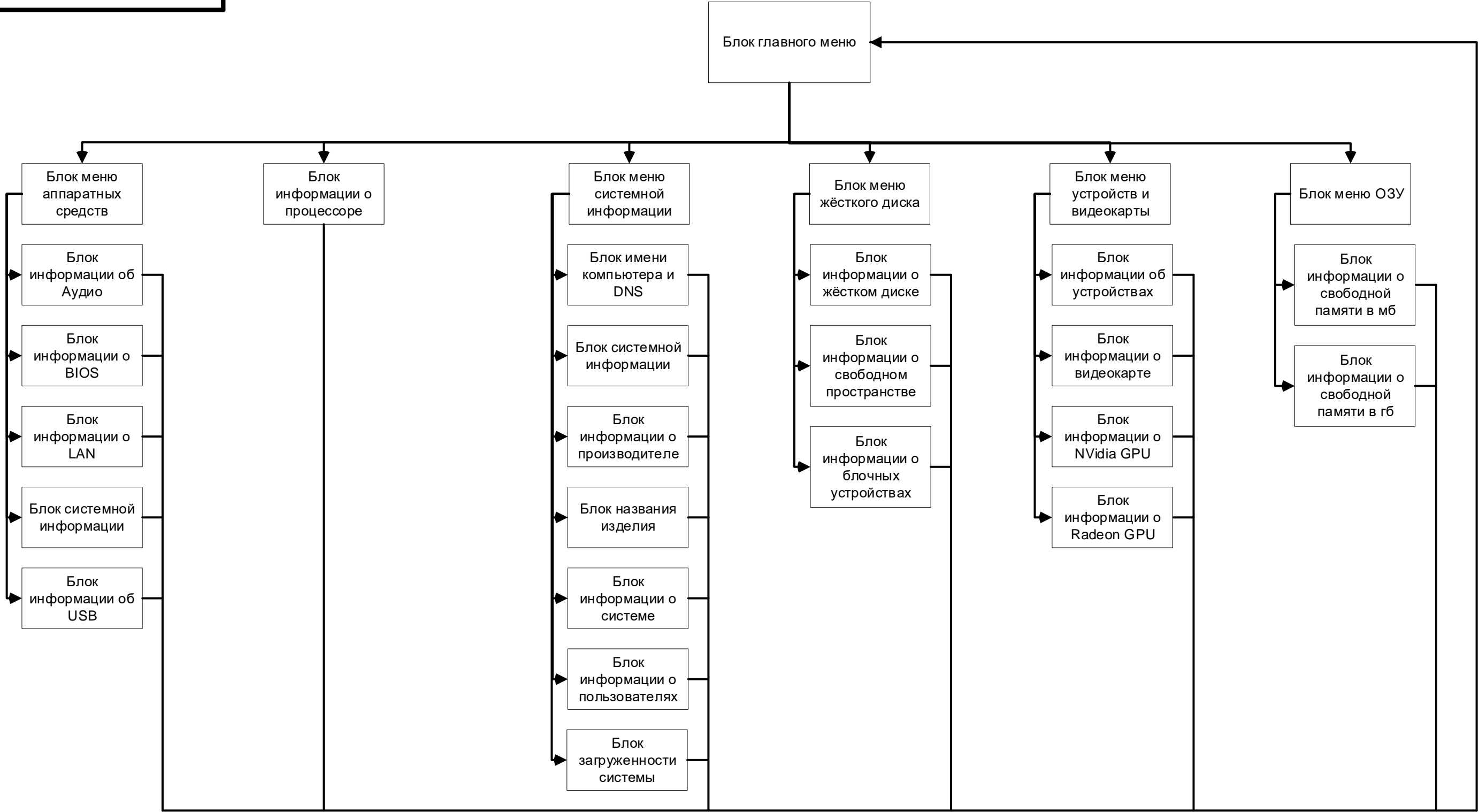
    void getHostname();
    void getSystem();
    void getSerialNumber();
    void getManufacturer();
    void getProductName();
    void getFullInfo();
    void getUptime();
    void getLoad();
    void getInfo();
};

#endif // SYSTEMINF_H
```

ПРИЛОЖЕНИЕ Б

(обязательно)

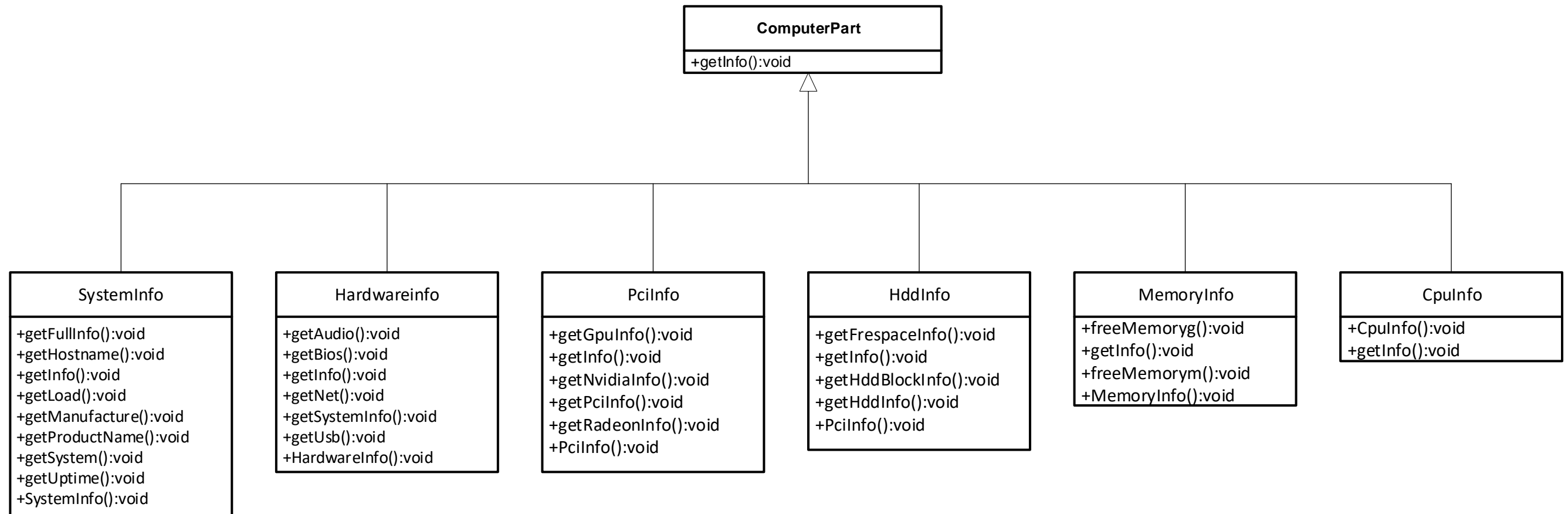
Структурная схема



					ГУИР.400201.009 С1				
					Утилита по сбору информации о системе. Схема структурная	Лит.	Масса	Масштаб	
Изм.	Лист	№ документа	Подпись	Дата					
Разраб.	Зверев					у			
Пров.	Желтко								
						Лист	Листов	1	
						ЭВМ, гр. 830501			

ПРИЛОЖЕНИЕ В
(обязательно)

Диаграмма классов



					ГУИР.400201.009 РР1					
					Утилита по сбору информации о системе. Диаграмма классов	Лит.		Масса	Масштаб	
Изм.	Лист	№ документа	Подпись	Дата			у			
Разраб.	Зверев									
Пров.	Желтко									
						Лист		Листов	1	
					ЭВМ, гр. 830501					

