минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Институт Информационных Технологий |
| Кафедра | Математического и Программного Обеспечения ЭВМ |

**КУРСОВАЯ РАБОТА**

|  |
| --- |
| по модулю Программирование |

|  |  |
| --- | --- |
| на тему | Объектно-ориентированное программирование на языке C++ |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-3оп-22 |
| направление подготовки (специальности) |
| 09.03.04, Программная инженерия |
| *шифр, наименование* |
| Маркелов Сергей Александрович |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Кустов Максим Александрович |
| *фамилия, имя, отчество* |
| ассистент |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_13\_\_»\_\_\_\_\_\_\_июня \_\_\_\_\_2024 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2024

*Год*

Аннотация

Данную курсовую работу по модулю «Программирование» на тему «Объектно-ориентированное программирование на языке C++» выполнил студент группы 1ПИб-02-3оп-22 Института информационных технологий Череповецкого государственного университета Маркелов Сергей Александрович.

Целью курсовой работы является разработка программы на языке C++, в которой реализована иерархия родственных классов для моделирования и обработки данных предметной области согласно варианта А11 «Устройства передачи данных». Программа должна содержать производный класс-шаблон для хранения указателей на абстрактный базовый класс-интерфейс согласно варианта B2 «Динамический вектор». Для хранения объектов каждого производного класса должна использоваться структура данных согласно варианта C8 «Очередь».

Курсовая включает в себя разделы: аннотация, оглавление, введение, объектно-ориентированный анализ предметной области, проектирование классов, разработка логической структуры программы, разработка модульной структуры программы, тестирование программы, заключение, список литературы. Вместе с курсовой работой идут приложения: техническое задание, руководство пользователя, текст программы, детальная диаграмма классов.

Оглавление

[Введение 4](#_Toc169104652)

[1. Объектно-ориентированный анализ предметной области 6](#_Toc169104653)

[2. Проектирование классов 9](#_Toc169104654)

[3. Разработка логической структуры программы 20](#_Toc169104655)

[4. Разработка модульной структуры программы 23](#_Toc169104656)

[5. Тестирование программы 25](#_Toc169104657)

[5.1. Тестирование классов 25](#_Toc169104658)

[5.2. Тестирование внешних функций 40](#_Toc169104659)

[5.3. Тестирование требований технического задания 42](#_Toc169104660)

[Заключение 44](#_Toc169104661)

[Список литературы 45](#_Toc169104662)

[Приложение 1. Техническое задание 46](#_Toc169104663)

[Приложение 2. Руководство пользователя 52](#_Toc169104664)

[Приложение 3. Текст программы 62](#_Toc169104665)

[Приложение 4. Детальная диаграмма классов 103](#_Toc169104666)

# Введение

Программы играют огромную роль в современном мире. Существует большое количество различных видов программ: драйвера, утилиты, плееры, браузеры, различные редакторы (текстовые, графические, музыкальные видеоредакторы), мессенджеры, архиваторы, антивирусы и многие другие.

Благодаря программам мы можем смотреть, обрабатывать или создавать фотографии, видео, музыку, работать с документами, обрабатывать данные и многое другое.

Программы создаются благодаря языкам программирования – формальным знаковым системам, предназначенным для описания алгоритмов обработки данных. Каждый язык программирования имеет свой алфавит и синтаксис [4].

Объектно-ориентированное программирование (ООП) – это методология программирования, основанная на концепции объектов. ООП позволяет структурировать программы таким образом, чтобы данные и операции над ними были организованы в единый объект, что упрощает управление и изменение кода [3].

Объект представляет собой экземпляр класса. Он содержит данные в виде полей (переменных) и код в виде методов (функций), определенных в классе. Объекты являются основными составляющими ООП и используются для моделирования реальных или абстрактных сущностей. Каждый объект обладает своим уникальным состоянием и поведением, определенным в классе, а также может взаимодействовать с другими объектами через их интерфейсы.

Класс – это шаблон или тип данных, определяющий состояние (через поля) и поведение (через методы) объектов. Класс является абстракцией, которая описывает общие характеристики объектов определенного типа. Классы могут наследоваться друг от друга, что позволяет создавать иерархии классов. Классы являются ключевым инструментом в ООП для организации и структурирования кода, что обеспечивает модульность, гибкость и повторное использование [3].

ООП строится на 4 основных концепциях:

* Инкапсуляция – это концепция, согласно которой данные и методы, которые работают с этими данными, объединены в единый объект. При этом внутренняя реализация объекта изолирована от прямого доступа и изменения извне. Контролируемый доступ к данным осуществляется только через специально определенные методы (интерфейс). Этот подход повышает безопасность программы, так как предотвращает случайные или непреднамеренные изменения данных.
* Наследование – это концепция, позволяющая создавать новые классы на основе уже существующих (базовых) классов. При этом производные классы наследуют свойства и методы базовых классов, а также могут добавлять свои собственные свойства и методы или изменять существующие. Наследование способствует повторному использованию кода и созданию иерархии классов, что делает программу более гибкой и поддерживаемой.
* Полиморфизм – это концепция, которая позволяет объектам с одинаковым интерфейсом проявлять различное поведение в зависимости от их конкретного типа или контекста использования. Выбор конкретного действия, в зависимости от ситуации, возлагается на компилятор языка программирования.
* Абстракция – это выделение общих характеристик объектов и игнорирование их различий, которые не являются существенными для решаемой задачи. Абстракция позволяет упростить моделирование программы.

Целью курсовой работы является знакомство с принципами объектно-ориентированного программирования.

В рамках курсовой работы необходимо написать программу на языке C++ с применением концепций ООП, которая реализует иерархию родственных классов для моделирования и обработки данных предметной области согласно варианта А11 «Устройства передачи данных».

# Объектно-ориентированный анализ предметной области

На практике ООП заключается в создании некоторого количества классов, включающих в себя классы-интерфейсы и классы с реализацией, и последующем их использовании. Графическое представление этих классов и связей между ними можно выполнить в виде диаграммы классов. В процессе объектно-ориентированного анализа основное внимание уделяется определению и описанию объектов (или понятий) в терминах предметной области.

Объектно-ориентированный анализ – это методология, при которой требования к системе рассматриваются с точки зрения классов и объектов, выявленных в предметной области.

Объектно-ориентированное проектирование – это методология проектирования, включающая процесс объектной декомпозиции и приемы представления логических, физических, статических и динамических моделей проектируемой системы [5].

Основные этапы объектно-ориентированного анализа и проектирования:

1. Сначала проводится анализ требований, в ходе которого выделяются основные процессы, происходящие в моделируемой системе, и формулируются в виде прецедентов. Прецедент – это текстовое описание процессов, происходящих в предметной области.
2. Далее проводится объектно-ориентированный анализ предметной области. Его задача – определение видов деятельности участников процесса (и соответствующих понятий) и создание концептуальной модели, отражающей различные категории элементов предметной области.
3. Затем проводится объектно-ориентированное проектирование. Оно сосредоточено на распределении обязанностей, что означает выделение задач и обязанностей различных программных объектов в приложении.

Предметной областью для разработки иерархии родственных классов являются устройства передачи данных.

Устройства передачи данных – это оборудование, предназначенное для передачи цифровой информации между различными устройствами или сетями. Они обеспечивают физическую или беспроводную связь между источником и приемником данных, обеспечивая надежную передачу информации в соответствии с установленными протоколами и стандартами связи [6].

Устройства передачи данных можно разделить на 3 класса:

* Проводные устройства передачи данных – эти устройства используют проводные кабели для передачи данных между устройствами. К таким устройствам относятся модемы, коммутаторы, хабы.
* Беспроводные устройства передачи данных – эти устройства передают данные без проводов – по радиоволнам или инфракрасному излучению. К таким устройствам относятся Wi-Fi-роутеры, антенны мобильных телефонов.
* Физические носители данных – эти устройства используются как для хранения, так и для передачи данных. К ним относятся жесткие диски (HDD), твердотельные накопители (SSD), USB-флешки и другие устройства.

По данной классификации была построена контекстная диаграмма классов для разработки программы (рис. 1).



8

Рис. 1. Контекстная диаграмма классов

# Проектирование классов

Проектирование классов предполагает выделение значимой информации и исключение незначимой. В ООП все свойства абстракции, определяющие состояние и поведение объекта, объединяются в единую программную единицу – класс.

Каждый класс должен иметь имя, а также может включать атрибуты и операции для обработки этих атрибутов.

Атрибут (свойство) – это именованное свойство класса, описывающее диапазон значений, которые может принимать экземпляр атрибута. Класс может иметь любое количество атрибутов или не иметь их вовсе. В последнем случае блок атрибутов оставляют пустым. Атрибут представляет некоторое свойство моделируемой сущности, которым обладают все объекты данного класса. Имя атрибута, как и имя класса, может быть текстовым. На практике для именования атрибута используются одно или несколько коротких существительных, выражающих свойство класса, к которому относится атрибут.

Операция (метод) – это реализация метода класса. Класс может иметь любое количество операций или не иметь их вовсе. Часто вызов операции объекта изменяет его атрибуты. Графически операции отображаются в нижнем блоке описания класса. Допускается указание только имен операций. Имя операции, как и имя класса, должно быть текстовым. На практике для именования операции используются короткие глагольные конструкции, описывающие поведение класса, которому принадлежит операция. Можно специфицировать операцию, устанавливая ее сигнатуру, включающую имя, тип и значение по умолчанию всех параметров, а для функций – тип возвращаемого значения.

После проектирования классов и их компонентов можно составить детальную диаграмму классов. Диаграмма является графическим изображением набора элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и путями (связями).

Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

* имя класса;
* атрибуты класса;
* операции (методы) класса.

Видимость для полей и методов указывается в виде левого символа в строке с именем соответствующего элемента. Для атрибутов и операций может быть указан один из трех типов видимости:

* + — public (общий);
* – — private (частный);
* # — protected (защищенный).

В результате объектно-ориентированного анализа был выделен класс-интерфейс DataDevices\_i для полиморфной обработки данных предметной области. В нем находятся декларации виртуальных методов:

* string getName – чисто виртуальный метод, геттер для имени;
* int getTraffic – чисто виртуальный метод, геттер для траффика;
* double getTime – чисто виртуальный метод, геттер для времени подключения;
* int getSpeed – чисто виртуальный метод, геттер для скорости передачи данных;
* void setData – чисто виртуальный метод, сеттер для передачи данных.

Абстрактный базовый класс DataDevices реализует интерфейс DataDevices\_i. Он хранит в себе общие элементы для всей предметной области:

* string name – свойство, описывающее название устройства;
* string data – свойство, описывающее передаваемые данные;
* static string work – статическое свойство, описывающее работоспособность устройства;
* int speed – свойство, описывающее скорость передачи данных;
* int traffic – свойство, описывающее количество переданных данных (траффик);
* double time – свойство, описывающее общее время передачи данных;
* string getName – геттер для имени;
* int getTraffic – геттер для траффика;
* double getTime – геттер для времени подключения;
* int getSpeed – геттер для скорости передачи данных;
* void setData – чисто виртуальный метод, сеттер для передачи данных.

От базового класса DataDevices были выделены 3 класса-наследника: Wireless, Wired и Storage.

Абстрактный производный класс Wireless помимо наследуемых элементов также содержит:

* int radius – свойство, описывающее радиус распространения сигнала;
* double frequency – свойство, описывающее частоту сигнала.

Абстрактный производный класс Wired помимо наследуемых элементов также содержит:

* cable cord – свойство, описывающее тип кабеля;
* int length\_cable – свойство, описывающее длину кабеля;
* string getCord – геттер для типа кабеля.

Абстрактный производный класс Storage помимо наследуемых элементов также содержит:

* int capacity – свойство, описывающее объем накопителя.

Детальная диаграмма классов DataDevices\_i, DataDevices, Wireless, Wired и Storage представлена на рис. 2.



Рис. 2. Детальная диаграмма классов DataDevices\_i, DataDevices, Wireless, Wired, Storage

От класса Wireless наследуются 2 класса – WiFi\_router и Mobile\_antenna.

Производный класс WiFi\_router помимо наследуемых элементов также содержит:

* string password – свойство, описывающее пароль, для подключения к Wi-Fi сети;
* security protection – свойство, описывающее протокол безопасности Wi-Fi сети;
* string getPassword – геттер для пароля;
* security getProtection – геттер для протокола безопасности;
* String^ getInfo – геттер для всей информации о Wi-Fi роутере.

Производный класс Mobile\_antenna помимо наследуемых элементов также содержит:

* generation type – свойство, описывающее поколение мобильной сети;
* int signal – свойство, описывающее уровень сигнала;
* getType – геттер для поколения сети;
* String^ getInfo – геттер для всей информации о мобильной сети.

Детальная диаграмма классов Wireless, WiFi\_router и Mobile\_antenna представлена на рис. 3.



Рис. 3. Детальная диаграмма классов WiFi\_router, Mobile\_antenna

От класса Wired наследуются 3 класса – Switch, Modem и Hub.

Производный класс Modem помимо наследуемых элементов также содержит:

* line connect – свойство, описывающее тип интернет-линии;
* string getLine – геттер для типа линии;
* String^ getInfo – геттер для всей информации о модеме.

Производный класс Switch помимо наследуемых элементов также содержит:

* int ports – свойство, описывающее количество портов;
* String^ getInfo – геттер для всей информации о коммутаторе.

Производный класс Hub помимо наследуемых элементов также содержит:

* int ports – свойство, описывающее количество портов;
* String^ getInfo – геттер для всей информации о хабе.

Детальная диаграмма классов Wired, Switch, Modem и Hub представлена на рис. 4.



Рис. 4. Детальная диаграмма классов Switch, Modem, Hub

От класса Storage наследуются 3 класса – Flash, HDD и SSD.

Производный класс Flash помимо наследуемых элементов также содержит:

* double version – свойство, описывающее версию USB-разъема;
* char connector – свойство, описывающее тип USB-разъема;
* String^ getInfo – геттер для всей информации о флешке.

Производный класс HDD помимо наследуемых элементов также содержит:

* connect\_type connector – свойство, описывающее тип разъема;
* string getType – геттер для типа разъема;
* String^ getInfo – геттер для всей информации о жестком диске.

Производный класс SSD помимо наследуемых элементов также содержит:

* connect\_type2 connector – свойство, описывающее тип разъема;
* string getType – геттер для типа разъема;

Детальная диаграмма классов Storage, Flash, HDD и SSD представлена на рис. 5.



Рис. 5. Детальная диаграмма классов Flash, HDD, SSD

Хранение объектов классов происходит в очереди – шаблонном классе Queue. Она содержит следующие элементы:

* struct Node – структура, представляющая собой узел очереди. Содержит следующие элементы:
  + T data – объект, хранящийся в данном узле;
  + Node\* next – указатель на следующий узел очереди;
* Node\* head –указатель на начало очереди;
* Node\* tail – указатель на конец очереди;
* int size – размер очереди;
* bool empty – метод, проверяющий очередь на пустоту;
* void add – метод, добавляющий элемент в очередь;
* string del – метод, берущий элемент из очереди и удаляющий его из нее;
* void nullQueue – метод, обнуляющий очередь;
* int getSize – геттер для размера очереди;
* void arr – метод, записывающий названия объектов из очереди в массив для дальнейшей сортировки;
* void print\_NewOld – метод, выводящий названия всех элементов очереди в порядке от новых к старым;
* void print\_AZ – метод, выводящий названия всех элементов очереди в алфавитном порядке;
* void print\_ZA – метод, выводящий названия всех элементов очереди в обратном алфавитному порядке;
* void search – метод, осуществляющий поиск элемента в очереди по его названию.

Детальная диаграмма очереди Queue представлена на рис. 6.



Рис. 6. Детальная диаграмма класса Queue

Для хранения указателей на объекты классов был создан базовый класс-контейнер Container. От него с помощью закрытого наследования был создан производный шаблонный класс Vector, представляющий собой динамический вектор.

Класс Container содержит следующие элементы:

* void\*\* data – указатель на массив указателей;
* size\_t size – текущий размер контейнера;
* size\_t size – текущая полная емкость контейнера;
* void resize – метод, изменяющий ёмкость контейнера.

Класс Vector содержит следующие элементы:

* void clear – метод, очищающий вектор;
* bool empty – метод, проверяющий вектор на пустоту;
* void add – метод, добавляющий элемент в вектор;
* T get – метод, берущий элемент из вектора;
* int getSize – геттер для размера вектора;
* void arr – метод, записывающий названия объектов из вектора в массив для дальнейшей сортировки;
* void print\_NewOld – метод, выводящий названия всех элементов вектора в порядке от новых к старым;
* void print\_AZ – метод, выводящий названия всех элементов вектора в алфавитном порядке;
* void print\_ZA – метод, выводящий названия всех элементов вектора в обратном алфавитному порядке;
* void search – метод, осуществляющий поиск элемента в векторе по названию;
* void broadcast – метод, записывающий передаваемые данные на устройство.

Детальная диаграмма класса-контейнера Container и вектора Vector представлена на рис. 7.



Рис. 7. Детальная диаграмма классов Container и Vector

Полная детальная диаграмма классов представлена в приложении 4.

# Разработка логической структуры программы

Программа работает по следующему алгоритму:

1. Инициализируются объекты классов;
2. Программа предлагает выбрать пользователю класс устройств передачи данных;
3. Программа предлагает выбрать пользователю устройство (объект) из ранее выбранного класса;
4. В случае, если был выбран класс устройств «Wi-Fi роутеры», программа запрашивает у пользователя пароль от Wi-Fi сетей;
5. Если был выбран класс устройств «Wi-Fi роутеры», и пароль введен неверно, выводится надпись «Пароль неверный». Программа предлагает ввести пароль снова, либо выбрать другой класс/устройство;
6. Если был выбран класс устройств «Wi-Fi роутеры», и пароль введен верно, или если был выбран любой другой класс, выводится надпись «Подключение выполнено». В очередь, соответствующую классу, записывается данный объект. В вектор записывается указатель на данный объект. На экран выводится информация о сети, также эта информация записывается в файл;
7. Программа предлагает пользователю загрузить на подключенное устройство текстовый файл для передачи данных. После загрузки она выводит его содержимое на экран;
8. Также есть возможность вывести историю подключений к устройствам. Программа предлагает выбрать один из 5 режимов: хронологический порядок, обратный хронологический порядок, алфавитный порядок, обратный алфавитный порядок и поиск по названию. Можно вывести историю как для выбранного класса (из очереди), так и для всех классов сразу (из вектора);
9. При выборе первых 4 режимов выводится история в выбранном пользователем порядке с помощью соответствующих функций класса очереди/вектора. Если подключений было меньше 20 – на экран, если больше – в файл;
10. При выборе 5 режима (поиск) программа запрашивает у пользователя название устройства, а затем ищет его.

Программа работает по следующему принципу: создается абстрактный базовый класс-интерфейс DataDevices\_i, от него методом одиночного наследования создаются абстрактные классы Wireless, Wired, Storage. От класса Wireless с помощью одиночного наследования создаются классы WiFi\_router, Mobile\_antenna. От класса Wired с помощью одиночного наследования создаются классы Modem, Switch, Hub. От класса Storage с помощью одиночного наследования создаются классы Flash, HDD, SSD. Объекты классов WiFi\_router, Mobile\_antenna, Modem, Router, Switch, Hub, Flash, HDD, SSD записываются в очередь Queue, а указатели на них – в вектор Vector, являющийся производным от класса Container.

Логическая структура программы представлена на рис. 8.



22

Рис. 8. Логическая структура программы

# Разработка модульной структуры программы

Для логической структуры программы, представленной в п. 3, были разработаны следующие модули (табл. 1).

Таблица 1

Модули программы

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип |
| MyForm.h | Интерфейс программы | Form |
| DataDevices\_i.h | Класс-интерфейс для полиморфной обработки данных предметной области | Class |
| DataDevices.h | Базовый класс с общими полями для всей предметной области будущих дочерних классов | Class |
| Container.h | Класс для хранения указателей | Class |
| Vector.h | Шаблонный класс – наследник класса для хранения указателей на интерфейс, представляющий собой вектор | Class |
| Queue.h | Класс-контейнер для объектов, представляющий собой очередь | Class |
| Wireless.h | Класс-наследник базового класса DataDevices | Class |
| WiFi\_router.h | Класс-наследник класса Wireless | Class |
| Mobile\_antenna.h | Класс-наследник класса Wireless | Class |
| Wired.h | Класс-наследник базового класса DataDevices | Class |
| Modem.h | Класс-наследник класса Wired | Class |
| Switch.h | Класс-наследник класса Wired | Class |
| Hub.h | Класс-наследник класса Wired | Class |
| Storage.h | Класс-наследник базового класса DataDevices | Class |
| Flash.h | Класс-наследник класса Storage | Class |
| HDD.h | Класс-наследник класса Storage | Class |
| SSD.h | Класс-наследник класса Storage | Class |
| Exception.h | Класс для обработки исключений | Class |

Модульная структура программы представлена на рис. 9.



24

Рис. 9. Модульная структура программы

# Тестирование программы

## Тестирование классов

Результаты тестирования класса DataDevices представлены в табл. 1.

Таблица 1

Тестирование класса DataDevices

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат тестирования |
| 01.04.2024 | getName() | Нет | Маркелов С. А. | Получение информации об имени устройства | Успех |
| 01.04.2024 | getTraffic() | Нет | Маркелов С. А. | Получение информации о количестве переданных данных | Успех |
| 01.04.2024 | getTime() | Нет | Маркелов С. А. | Получение информации о времени подключения | Успех |
| 01.04.2024 | getSpeed() | Нет | Маркелов С. А. | Получение информации о скорости передачи данных | Успех |
| 01.04.2024 | setData() | string | Маркелов С. А. | Запись данных для их передачи | Успех |

Результаты тестирования класса WiFi\_router представлены в табл. 2.

Таблица 2

Тестирование класса WiFi\_router

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 02.04.2024 | WiFi\_router() | Нет | Маркелов С. А. | Создание объекта класса WiFi\_router без передачи параметров, со значениями по умолчанию | Успех |
| 02.04.2024 | WiFi\_router() | string, string, security, int, int, double | Маркелов С. А. | Создание объекта класса WiFi\_router | Успех |
| 02.04.2024 | WiFi\_router() | string | Маркелов С. А. | Создание объекта класса WiFi\_router | Успех |
| 02.04.2024 | WiFi\_router() | double, string | Маркелов С. А. | Создание объекта класса WiFi\_router | Успех |
| 02.04.2024 | WiFi\_router() | int, int, string | Маркелов С. А. | Создание объекта класса WiFi\_router | Успех |

Продолжение табл. 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 02.04.2024 | WiFi\_router() | WiFi\_router& | Маркелов С. А. | Создание объекта класса WiFi\_router – копии другого объекта данного класса | Успех |
| 02.04.2024 | getPassword() | Нет | Маркелов С. А. | Получение информации о пароле от Wi-Fi роутера | Ошибка. Функция выдавала неверный результат. Исправлено |
| 02.04.2024 | getProtection() | Нет | Маркелов С. А. | Получение информации о протоколе безопасности | Успех |
| 02.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о Wi-Fi роутере | Ошибка. Результат функции не выводился на экран. Исправлено |

Результаты тестирования класса Mobile\_antenna представлены в табл. 3.

Таблица 3

Тестирование класса Mobile\_antenna

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 03.04.2024 | Mobile\_antenna() | Нет | Маркелов С. А. | Создание объекта класса Mobile\_ antenna без передачи параме-тров, со значени-ями по умолчанию | Успех |
| 03.04.2024 | Mobile\_antenna() | string, generation, int, int, double | Маркелов С. А. | Создание объекта класса Mobile\_ antenna | Успех |
| 03.04.2024 | Mobile\_antenna() | string | Маркелов С. А. | Создание объекта класса Mobile\_ antenna | Успех |
| 03.04.2024 | Mobile\_antenna() | string, generation | Маркелов С. А. | Создание объекта класса Mobile\_ antenna | Успех |

Продолжение табл. 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 03.04.2024 | Mobile\_antenna() | string, generation, int | Маркелов С. А. | Создание объекта класса Mobile\_antenna | Успех |
| 03.04.2024 | Mobile\_antenna() | Mobile\_antenna& | Маркелов С. А. | Создание объекта класса Mobile\_antenna – копии другого объекта данного класса | Успех |
| 03.04.2024 | getType() | Нет | Маркелов С. А. | Получение информации о поколении мобильной сети | Успех |
| 03.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о мобильной сети | Успех |

Результаты тестирования класса Wired представлены в табл. 4.

Таблица 4

Тестирование класса Wired

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат тестирования |
| 03.04.2024 | getCord() | Нет | Маркелов С. А. | Получение информации о типе кабеля | Успех |

Результаты тестирования класса Modem представлены в табл. 5.

Таблица 5

Тестирование класса Modem

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 04.04.2024 | Modem() | Нет | Маркелов С. А. | Создание объекта класса Modem без передачи параметров, со значениями по умолчанию | Успех |
| 04.04.2024 | Modem() | string, cable, int, int, line | Маркелов С. А. | Создание объекта класса Modem | Успех |
| 04.04.2024 | Modem() | Modem& | Маркелов С. А. | Создание объекта класса Mobile\_antenna – копии другого объекта данного класса | Успех |
| 04.04.2024 | getLine() | string | Маркелов С. А. | Получение информации о типе линии | Успех |
| 04.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о модеме | Успех |

Результаты тестирования класса Switch представлены в табл. 6.

Таблица 6

Тестирование класса Switch

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 04.04.2024 | Switch() | Нет | Маркелов С. А. | Создание объекта класса Switch без передачи параметров, со значениями по умолчанию | Успех |
| 04.04.2024 | Switch() | string, cable, int, int, int | Маркелов С. А. | Создание объекта класса Switch | Успех |
| 04.04.2024 | Switch() | Switch& | Маркелов С. А. | Создание объекта класса Switch – копии другого объекта данного класса | Успех |
| 04.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о коммутаторе | Успех |

Результаты тестирования класса Hub представлены в табл. 7.

Таблица 7

Тестирование класса Hub

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 04.04.2024 | Hub() | Нет | Маркелов С. А. | Создание объекта класса Hub без передачи параметров, со значениями по умолчанию | Успех |
| 04.04.2024 | Hub() | string, cable, int, int, int | Маркелов С. А. | Создание объекта класса Hub | Успех |
| 04.04.2024 | Hub() | Hub& | Маркелов С. А. | Создание объекта класса Hub – копии другого объекта данного класса | Успех |
| 04.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о хабе | Ошибка при выводе результата функции на экран. Исправлено |

Результаты тестирования класса Flash представлены в табл. 8.

Таблица 8

Тестирование класса Flash

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 05.04.2024 | Flash() | Нет | Маркелов С. А. | Создание объекта класса Flash без передачи параметров, со значениями по умолчанию | Успех |
| 05.04.2024 | Flash() | string, int, double, char, int | Маркелов С. А. | Создание объекта класса Flash | Успех |
| 05.04.2024 | Flash() | string | Маркелов С. А. | Создание объекта класса Flash | Успех |
| 05.04.2024 | Flash() | string, int | Маркелов С. А. | Создание объекта класса Flash | Успех |
| 05.04.2024 | Flash() | Flash& | Маркелов С. А. | Создание объекта класса Flash – копии другого объекта данного класса | Успех |
| 05.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о флешке | Успех |

Результаты тестирования класса HDD представлены в табл. 9.

Таблица 9

Тестирование класса HDD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 05.04.2024 | HDD() | Нет | Маркелов С. А. | Создание объекта класса HDD без передачи параметров, со значениями по умолчанию | Успех |
| 05.04.2024 | HDD() | string, int, connect\_type, int | Маркелов С. А. | Создание объекта класса HDD | Успех |
| 05.04.2024 | HDD() | HDD& | Маркелов С. А. | Создание объекта класса HDD – копии другого объекта данного класса | Успех |
| 05.04.2024 | getType() | Нет | Маркелов С. А. | Получение информации о типе диска | Успех |
| 05.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о диске | Успех |

Результаты тестирования класса SSD представлены в табл. 10.

Таблица 10

Тестирование класса SSD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 05.04.2024 | SSD() | Нет | Маркелов С. А. | Создание объекта класса SSD без передачи параметров, со значениями по умолчанию | Успех |
| 05.04.2024 | SSD() | string, int, connect\_type2, int | Маркелов С. А. | Создание объекта класса SSD | Успех |
| 05.04.2024 | SSD() | SSD& | Маркелов С. А. | Создание объекта класса SSD – копии другого объекта данного класса | Успех |
| 05.04.2024 | getType() | Нет | Маркелов С. А. | Получение информации о типе диска | Успех |
| 05.04.2024 | getInfo() | Нет | Маркелов С. А. | Получение информации о диске | Успех |

Результаты тестирования класса Container представлены в табл. 11.

Таблица 11

Тестирование класса Container

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 06.04.2024 | resize() | size\_t | Маркелов С. А. | Изменение размера контейнера | Успех |
| 06.04.2024 | Container() | Нет | Маркелов С. А. | Создание контейнера | Успех |

Результаты тестирования класса Vector представлены в табл. 12.

Таблица 12

Тестирование класса Vector

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 1 | 2 | 3 | 5 | 5 | 6 |
| 06.04.2024 | Vector() | Нет | Маркелов С. А. | Создание динамического вектора | Успех |
| 06.04.2024 | Vector() | Vector& | Маркелов С. А. | Создание копии вектора | Успех |
| 06.04.2024 | clear() | Нет | Маркелов С. А. | Очистка вектора | Успех |
| 06.04.2024 | empty() | Нет | Маркелов С. А. | Проверка вектора на пустоту | Успех |
| 06.04.2024 | add() | T& | Маркелов С. А. | Добавление указателя в вектор | Успех |
| 06.04.2024 | get() | Нет | Маркелов С. А. | Взятие указателя из вектора | Успех |

Продолжение табл. 12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 5 | 5 | 6 |
| 06.04.2024 | getSize() | Нет | Маркелов С. А. | Получение информации о размере вектора | Успех |
| 06.04.2024 | arr() | string\* | Маркелов С. А. | Создание массива названий элементов вектора | Ошибка. При попытке создания массива программа вылетала. Исправлено |
| 06.04.2024 | print\_NewOld() | ofstream&, Device::MyForm^ | Маркелов С. А. | Вывод названий элементов вектора в порядке от новых к старым | Ошибка. Названия выводились в неправильном порядке. Исправлено |
| 06.04.2024 | print\_AZ() | ofstream&, Device::MyForm^ | Маркелов С. А. | Вывод названий элементов вектора в алфавитном порядке | Ошибка. При попытке вывода программа вылетала. Исправлено |
| 06.04.2024 | print\_ZA() | ofstream&, Device::MyForm^ | Маркелов С. А. | Вывод названий элементов вектора в обратном алфавитному порядке | Ошибка. При попытке вывода программа вылетала. Исправлено |

Продолжение табл. 12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 06.04.2024 | search() | string&, Device::MyForm^ | Маркелов С. А. | Поиск по названию элемента вектора | Успех |
| 06.04.2024 | broadcast() | String^, Device::MyForm^ | Маркелов С. А. | Запись в последний элемент вектора информации для передачи | Успех |

Результаты тестирования класса Queue представлены в табл. 13.

Таблица 13

Тестирование класса Queue

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 1 | 2 | 3 | 5 | 5 | 6 |
| 06.04.2024 | Queue() | Нет | Маркелов С. А. | Создание очереди | Успех |
| 06.04.2024 | Queue() | Queue& | Маркелов С. А. | Создание копии очереди | Успех |
| 06.04.2024 | empty() | Нет | Маркелов С. А. | Проверка очереди на пустоту | Успех |
| 06.04.2024 | add() | T | Маркелов С. А. | Добавление объекта в очередь | Успех |

Продолжение табл. 13

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 06.04.2024 | del() | Нет | Маркелов С. А. | Взятие и удаление объекта из очереди | Успех |
| 06.04.2024 | nullQueue() | Нет | Маркелов С. А. | Обнуление очереди | Успех |
| 06.04.2024 | getSize() | Нет | Маркелов С. А. | Получение информации о размере очереди | Успех |
| 06.04.2024 | arr() | string\* | Маркелов С. А. | Создание массива названий элементов очереди | Успех |
| 06.04.2024 | print\_NewOld() | ofstream&, Device::MyForm^ | Маркелов С. А. | Вывод названий элементов очереди в порядке от новых к старым | Ошибка. Названия выводились в неправильном порядке. Исправлено |
| 06.04.2024 | print\_AZ() | ofstream&, Device::MyForm^ | Маркелов С. А. | Вывод названий элементов очереди в алфавитном порядке | Успех |

Продолжение табл. 13

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 06.04.2024 | print\_ZA() | ofstream&, Device::MyForm^ | Маркелов С. А. | Вывод названий элементов очереди в обратном алфавитному порядке | Успех |
| 06.04.2024 | search() | string&, Device::MyForm^ | Маркелов С. А. | Поиск по названию элемента очереди | Успех |

## Тестирование внешних функций

Результаты тестирования внешних функций представлены в табл. 14.

Таблица 14

Тестирование внешних функций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата | Тестируемая функция | Входные данные | Кто проводил тестирование | Описание теста | Результат работы |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 07.04.2024 | Authorization() | T&, Queue<T>& | Маркелов С. А. | Подключение к устройству, запись объекта в очередь и указателя на объект в вектор | Успех |
| 07.04.2024 | getIndex1() | Нет | Маркелов С. А. | Получение номера выбранного типа устройств | Успех |

Продолжение табл. 14

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 07.04.2024 | getIndex2() | int | Маркелов С. А. | Получение номера выбранного устройства | Успех |
| 07.04.2024 | getIndex3() | Нет | Маркелов С. А. | Получение номера выбранной сортировки | Успех |
| 07.04.2024 | setLucky() | String^ | Маркелов С. А. | Изменение значения счетчика удачных подключений | Успех |
| 07.04.2024 | setUnlucky() | String^ | Маркелов С. А. | Изменение значения счетчика неудачных подключений | Успех |
| 07.04.2024 | setTraffic() | String^ | Маркелов С. А. | Изменение значения общего количества переданной информации | Успех |
| 07.04.2024 | setTime() | String^ | Маркелов С. А. | Изменение значения общего времени подключения | Успех |
| 07.04.2024 | setText() | String^ | Маркелов С. А. | Запись текста для вывода на экран | Успех |
| 07.04.2024 | setClear() | Нет | Маркелов С. А. | Очистка поля для вывода текста | Успех |
| 07.04.2024 | Button1\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button2\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button3\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |

Продолжение табл. 14

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 07.04.2024 | button4\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button5\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button6\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button7\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button8\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button8\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |
| 07.04.2024 | button8\_Click | System::Object^, System::EventArgs^ | Маркелов С. А. | Выполнение действий при нажатии кнопки | Успех |

## Тестирование требований технического задания

Результаты тестирования требований технического задания представлены в табл. 15.

Таблица 15

Тестирование требований технического задания

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Тестируемая функция или модуль | Кто проводил тестирование | Описание теста | Результат работы |
| 08.04.2024 | Модуль MyForm.h | Маркелов С. А. | Создание объектов классов из иерархии | Успех |
| 08.04.2024 | Модуль Vector.h | Маркелов С. А. | Создание обобщенного контейнерного класса для храненния указателей на объекты | Успех |
| 08.04.2024 | Модуль Queue.h | Маркелов С. А. | Создание класса-контейнера для хранения объектов | Успех |
| 08.04.2024 | Модуль MyForm.h | Маркелов С. А. | Файловый ввод-вывод данных, ввод данных с клавиатуры, вывод данных на дисплей | Успех |
| 08.04.2024 | Модуль MyForm.h | Маркелов С. А. | Вызов исключительных ситуаций | Успех |

# Заключение

В ходе выполнения курсовой работы была разработана программа на языке C++. Программа предоставляет пользователю возможность подключится к различным классам устройств передачи данных. Также пользователь может загружать текстовые файлы для передачи, выводить историю подключений и осуществлять поиск по ней, а также просматривать статистику подключений.

В ходе разработки программы были освоены основные принципы ООП, такие как полиморфизм, инкапсуляция, наследование и абстракция, получены умения проектировать и разрабатывать локальные приложения.

# Список литературы

1. Методика и организация самостоятельной работы студентов: учебно-методическое пособие / Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских [и др.]. – Череповец: ФГБОУ ВПО ЧГУ, 2015. – 243 с.
2. Программирование. Базовые средства языка программирования C++: Учеб. Пособие / Е.В. Ершов, О.Г. Ганичева, В.В. Селивановских [и др.]. – Череповец: ЧГУ, 2011. – 181 с.
3. Страуструп, Бьярне. Язык программирования C++. Краткий курс, 2-е изд. : Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 320 с.
4. Язык программирования — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Язык_программирования>. Дата обращения: 12.06.2024.
5. UML — Википедия [электр.ресурс] https://ru.wikipedia.org/wiki/UML. Дата обращения: 12.06.2024.
6. Передача данных — Википедия [электр.ресурс] https://ru.wikipedia.org/wiki/Передача\_данных. Дата обращения: 12.06.2024.

# Приложение 1

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Модуль: Программирование

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2024 г.

Объектно-ориентированное программирование на языке C++

Техническое задание на курсовую работу

Листов 6

Руководитель: Кустов М. А.

Исполнитель: студент гр. 1ПИб-02-3оп-22

Маркелов С. А.

2024 г.

Введение

Тема курсовой работы – «Объектно-ориентированное программирование на языке C++».

Целью курсовой работы является разработка программы на языке C++, в которой реализована иерархия родственных классов предметной области согласно варианта А11 «Устройства передачи данных».

Программа предназначена для изучения основных принципов ООП и получения опыта их применения на практике.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Объектно-ориентированное программирование», выданное на кафедре МПО ЭВМ.

Общие требования к программному обеспечению для разработки следующие:

* Разработать иерархии родственных типов, корневой класс которых класс-интерфейс, для моделирования и обработки данных предметных областей набором отложенных методов - полиморфная обработка родственных объектов согласно варианту А11 (Устройства передачи данных). Организовать взаимодействие иерархий типов через указатели на интерфейсы и виртуальные функции (отложенные методы).
* Создать обобщенный (void\*) контейнерный класс (базовый) и от него, используя закрытое наследование, производный класс – шаблон для хранения указателей на абстрактный базовый класс - интерфейс согласно варианту В2 (Динамический вектор).
* Для хранения объектов каждого производного класса использовать структуру данных согласно варианту С8 (Очередь).

2. Назначение разработки

Функциональным назначением программы является реализация классов предметной области «Устройства передачи данных», а также реализация контейнерных классов для хранения объектов этих классов.

3. Требования к программе

* 1. Требования к функциональным характеристикам

К разрабатываемой программе предъявляются следующие требования:

* программа должна содержать иерархию родственных типов для моделирования и обработки данных предметной области «Устройства передачи данных»;
* в иерархии корневой класс должен быть абстрактным базовым классом (классом-интерфейсом);
* в программе должна быть реализована полиморфная обработка родственных объектов (обработка данных должна производится набором отложенных методов);
* программа должна содержать обобщенный (void\*) контейнерный класс (базовый), и от него, с применением закрытого наследования, производный класс-шаблон (динамический вектор) для хранения указателей на абстрактный базовый класс-интерфейс;
* программа должна хранить объекты каждого производного класса в структуре данных – очереди;
* программа должна содержать функции обработки данных – сортировка и поиск по выбранным полям и задаваемым диапазонам значений и другие функции, в том числе перегруженные;
* в программе должен быть реализован файловый ввод/вывод данных, ввод данных с клавиатуры, вывод данных на дисплей;
* в программе должна быть предусмотрена обработка различных исключительных ситуаций.
  1. Требования к надежности

К разрабатываемой программе предъявляются следующие требования:

* работа всех функций должна быть проверена и результаты проверки оформлены протоколом тестирования;
* должна быть реализована обработка исключительных ситуаций при загрузке объектов из файлов;
* должна быть реализована обработка исключительных ситуаций при добавлении объектов с помощью пользовательской формы;
* должен быть реализован корректный вывод информации на экран.
  1. Условия эксплуатации

Требования не предъявляются.

* 1. Требования к составу и параметрам технических средств

Для корректной работы программы необходимо:

* оперативная память: не менее 1 Гб для 32-разрядной ОС, 2 Гб для 64-разрядной ОС;
* процессор не менее чем с 2 ядрами и тактовой частотой не ниже 1 ГГц;
* разрешение экрана не менее 800 х 600;
* видеокарта с видеопамятью не менее 1 Гб;
* наличие мыши и клавиатуры.
  1. Требования к информационной и программной совместимости

Для корректной работы программы необходимо:

* установленная современная операционная система (например, Windows 7, 8, 8.1, 10, 11, macOS, Linux);
* установленный на устройстве пакет библиотек «Microsoft Visual C++».

4. Требования к программной документации

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

1. Титульный лист
2. Аннотация
3. Оглавление
4. Введение
5. Основная часть
6. Заключение
7. Список литературы
8. Приложения (техническое задание, руководство пользователя, текст программы, детальная диаграмма классов)

5. Стадии и этапы разработки

В данном пункте описаны стадии и этапы разработки программы (табл. П1.1).

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Оформление технического задания | 21.03.2024 | Оформленное техническое задание | Выполнено |
| Объектно-ориентированный анализ предметной области | 21.03.2024 | Проведен анализ предметной области | Выполнено |
| Создание иерархии родственных классов | 28.03.2024 | Готовая иерархия классов | Выполнено |
| Проектирование приложения | 25.04.2024 | Готовое приложение | Выполнено |
| Тестирование и отладка | 16.05.2024 | Конечный вариант программы | Выполнено |
| Оформление сопроводительной документации | 30.05.2024 | Оформленная сопроводительная документация | Выполнено |

6. Порядок контроля и приемки

Проверка контроля проходит в соответствии с графиком стадий и этапов разработки:

* сдача технического задания и иерархии родственных классов;
* обработка исключительных ситуаций и тестирование программы;
* предоставление программы в электронном виде;
* сдача расчетно-пояснительной записки.

# Приложение 2

Руководство пользователя

1. Общие сведения о программе

Файл программы называется «Device.exe» (рис. П2.1). Программа позволяет создавать объекты устройств передачи данных, загружать файлы для передачи и выводить историю подключений, а также осуществлять поиск по ней.

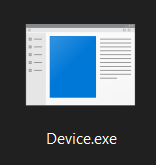


Рис. П2.1. Файл программы

1. Описание установки

Установка программы не требуется, достаточно лишь запустить exe-файл. Сторонних программ для работы также не требуется.

1. Описание запуска

Для запуска программы необходимо дважды нажать на файл Device.exe, либо на его ярлык.

1. Инструкции по работе

В выпадающем списке необходимо выбрать класс устройств, к которому нужно подключиться, и нажать кнопку «ОК» (рис. П2.2).

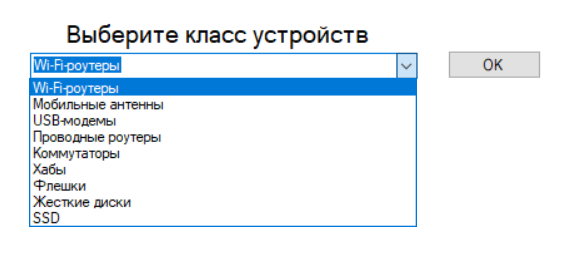


Рис. П2.2. Выбор класса устройств

Затем появляется ещё один выпадающий список, в котором необходимо выбрать одно из устройств данного класса, и затем нужно нажать кнопку «ОК» (рис. П2.3).

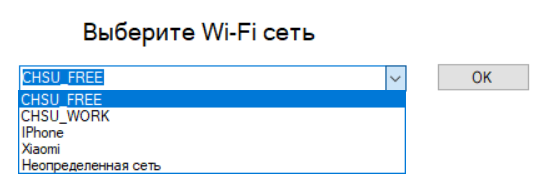


Рис. П2.3. Выбор устройства

В случае выбора Wi-Fi роутера необходимо будет ввести пароль в появившемся текстовом блоке и нажать кнопку «ОК» (рис. П2.4).

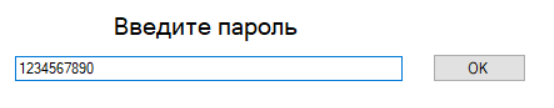


Рис. П2.4. Ввод пароля от Wi-Fi роутера

Под списками и кнопками расположена статистика по подключениям: количество удачных и неудачных подключений, количество переданной информации и общее время подключения. При необходимости обнулить статистику можно нажать кнопку «Сброс». Ниже расположена информация о подключенном устройстве (рис. П2.5). Эта информация также записывается в файл (рис. П2.6).

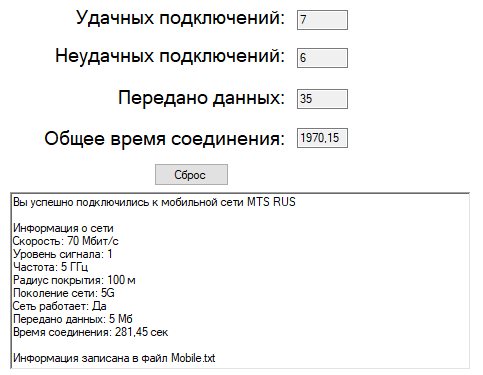


Рис. П2.5. Статистика по подключениям к устройствам и информация о подключенном устройстве

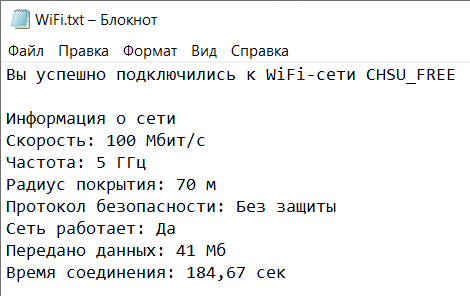


Рис. П2.5. Информация о подключенном устройстве в файле

После подключения появляется возможность выбрать текстовый файл для передачи (рис. П2.7). Его содержание отображается в отдельном текстовом блоке и записывается в объект класса (рис. П2.8).

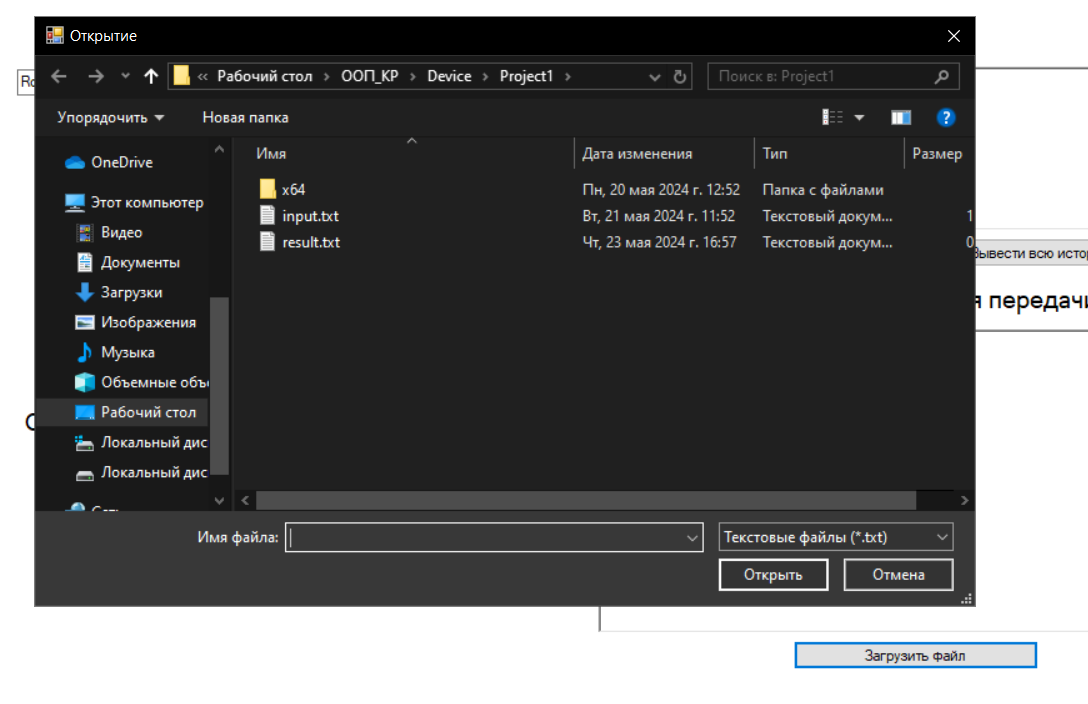


Рис. П2.7. Загрузка файла

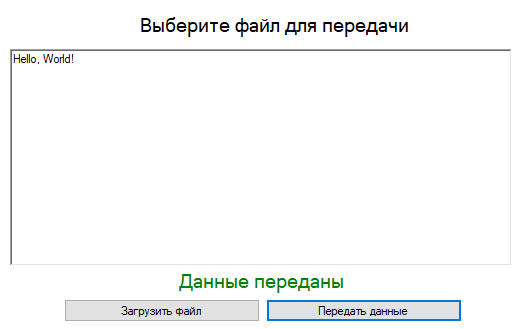


Рис. П2.8. Вывод содержимого файла

Также пользователь может вывести историю подключений. Для этого в выпадающем списке нужно выбрать один из режимов сортировки: хронологический порядок (от старых к новым), обратный хронологический порядок (от новых к старым), алфавитный порядок (от A до Z), обратный алфавитный порядок (от Z до A), режим поиска (рис. П2.9).

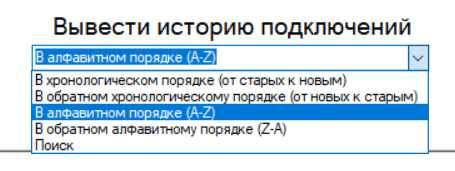


Рис. П2.9. Выбор режима сортировки истории подключений

После выбора режима сортировки нужно нажать одну из двух кнопок. При нажатии кнопки «Вывести для выбранного класса» будет выведена история подключений только выбранного класса. При нажатии кнопки «Вывести всю историю» – история всех подключений (рис. П2.10).

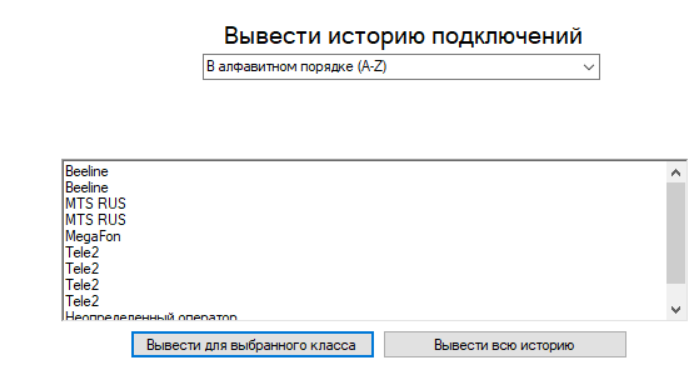


Рис. П2.10. Вывод истории подключений в алфавитном порядке

Если выбран поиск – появится текстовое поле, в котором нужно ввести название устройства, и затем нажать кнопку «ОК» (рис. П2.11).

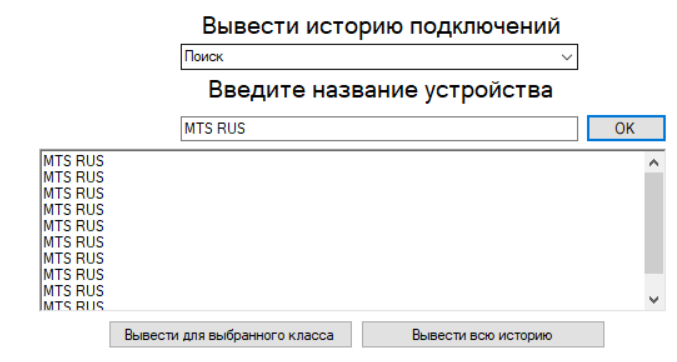


Рис. П2.11. Поиск по истории подключений

Если подключений было менее 20 – информация будет выведена на экран. Если более 20 – в файл (рис П2.12).

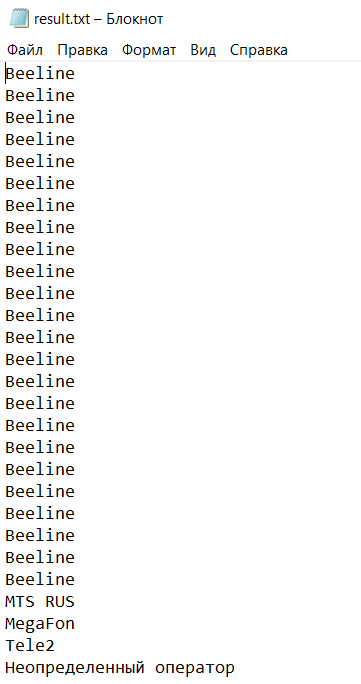


Рис. П2.12. История подключений в файле

1. Сообщения пользователю

При вводе неверного пароля для Wi-Fi роутера программа выводит сообщение «Пароль неверный» (рис. П2.13). Пользователю необходимо ввести пароль еще раз, либо выбрать другое устройство/класс.

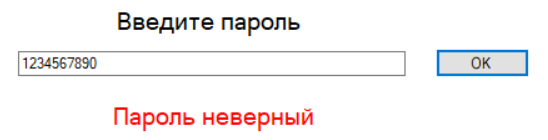


Рис. П2.13. Сообщение «Пароль неверный»

При вводе верного пароля, либо при подключении к любому иному устройству выводится сообщение «Подключение выполнено» (рис. П2.14). Пользователь может работать дальше.

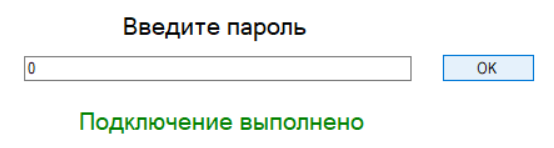


Рис. П2.14. Сообщение «Подключение выполнено»

Если пользователь нажал на кнопку «Загрузить файл», а затем закрыл окно загрузки, будет выведено сообщение «Ошибка загрузки файла!» (рис. П2.15). Пользователю необходимо нажать на кнопку и выбрать файл еще раз.

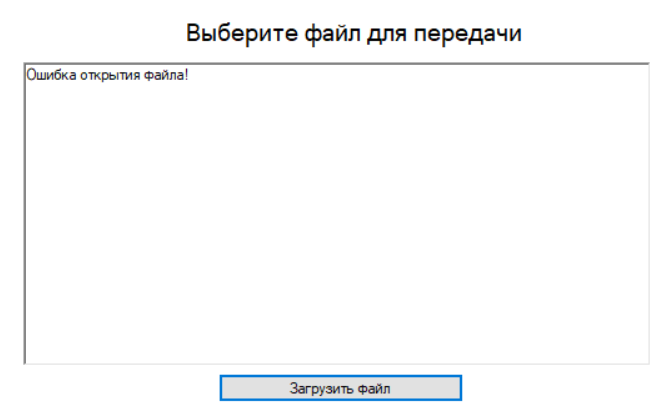


Рис. П2.15. Сообщение «Ошибка открытия файла!»

Если при выводе истории программа не обнаружит подключений, будет выведено сообщение «Подключений не было» (рис. П2.16).

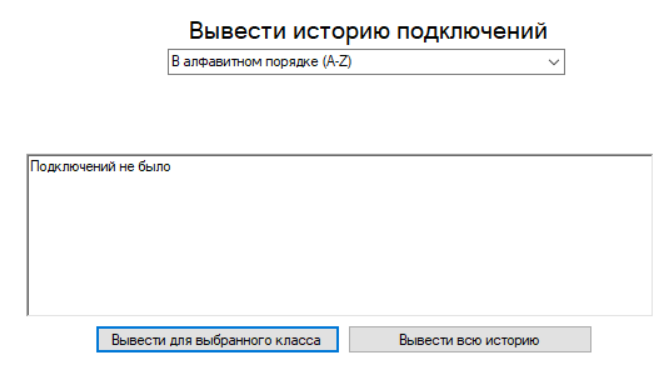


Рис. П2.16. Сообщение «Подключений не было»

Если при поиске пользователь задаст название устройства, к которому не было подключений, будет выведено сообщение «Устройство не найдено» (рис. П2.17).

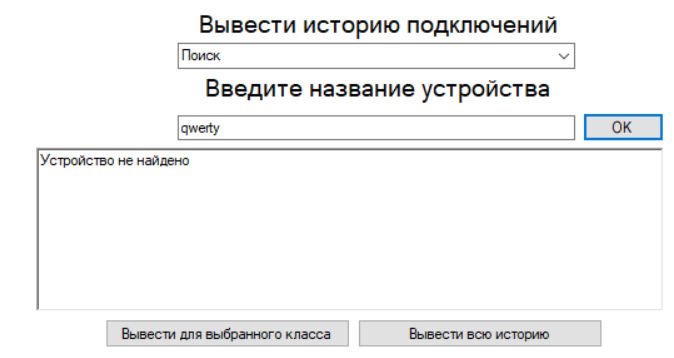


Рис. П2.17. Сообщение «Устройство не найдено»

# Приложение 3

Текст программы

Текст модуля MyForm.h представлен на рис. П3.1.

#pragma once

#include "windows.h"

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <stdlib.h>

#include <string.h>

#include <msclr\marshal.h>

#include <msclr\marshal\_cppstd.h>

#include <msclr\marshal\_windows.h>

#include <iomanip>

#include <map>

namespace Device {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

using namespace System::Text;

using namespace System::Collections::Generic;

using namespace System::Runtime::InteropServices;

using namespace msclr::interop;

using namespace std;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::ColorDialog^ colorDialog1;

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::ComboBox^ comboBox2;

Рис. П3.1. Текст модуля MyForm.h

Продолжение рис. П3.1

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::ComboBox^ comboBox3;

private: System::Windows::Forms::ComboBox^ comboBox4;

private: System::Windows::Forms::ComboBox^ comboBox6;

private: System::Windows::Forms::ComboBox^ comboBox7;

private: System::Windows::Forms::ComboBox^ comboBox8;

private: System::Windows::Forms::ComboBox^ comboBox9;

private: System::Windows::Forms::ComboBox^ comboBox10;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Label^ label12;

private: System::Windows::Forms::Label^ label13;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Label^ label14;

private: System::Windows::Forms::Label^ label15;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::Label^ label16;

private: System::Windows::Forms::Label^ label17;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::ComboBox^ comboBox11;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Button^ button5;

private: System::Windows::Forms::Label^ label18;

private: System::Windows::Forms::Label^ label19;

private: System::Windows::Forms::TextBox^ textBox6;

private: System::Windows::Forms::Button^ button7;

private: System::Windows::Forms::RichTextBox^ richTextBox2;

private: System::Windows::Forms::Label^ label20;

private: System::Windows::Forms::Button^ button9;

private: System::Windows::Forms::RichTextBox^ richTextBox3;

private: System::Windows::Forms::Button^ button6;

private: System::Windows::Forms::Button^ button8;

private: System::Windows::Forms::Button^ button10;

private: System::Windows::Forms::Label^ label21;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->colorDialog1 = (gcnew System::Windows::Forms::ColorDialog());

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());

Продолжение рис. П3.1

this->label3 = (gcnew System::Windows::Forms::Label());

this->comboBox3 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox4 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox6 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox7 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox8 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox9 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox10 = (gcnew System::Windows::Forms::ComboBox());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->label8 = (gcnew System::Windows::Forms::Label());

this->label9 = (gcnew System::Windows::Forms::Label());

this->label10 = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->label11 = (gcnew System::Windows::Forms::Label());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->button3 = (gcnew System::Windows::Forms::Button());

this->label12 = (gcnew System::Windows::Forms::Label());

this->label13 = (gcnew System::Windows::Forms::Label());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->label14 = (gcnew System::Windows::Forms::Label());

this->label15 = (gcnew System::Windows::Forms::Label());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->label16 = (gcnew System::Windows::Forms::Label());

this->label17 = (gcnew System::Windows::Forms::Label());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->button4 = (gcnew System::Windows::Forms::Button());

this->comboBox11 = (gcnew System::Windows::Forms::ComboBox());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->button5 = (gcnew System::Windows::Forms::Button());

this->label18 = (gcnew System::Windows::Forms::Label());

this->label19 = (gcnew System::Windows::Forms::Label());

this->textBox6 = (gcnew System::Windows::Forms::TextBox());

this->button7 = (gcnew System::Windows::Forms::Button());

this->richTextBox2 = (gcnew System::Windows::Forms::RichTextBox());

this->label20 = (gcnew System::Windows::Forms::Label());

this->button9 = (gcnew System::Windows::Forms::Button());

this->richTextBox3 = (gcnew System::Windows::Forms::RichTextBox());

this->button6 = (gcnew System::Windows::Forms::Button());

this->button8 = (gcnew System::Windows::Forms::Button());

this->button10 = (gcnew System::Windows::Forms::Button());

this->label21 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// comboBox1

//

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(8) {

L"Wi-Fi-роутеры", L"Мобильные антенны", L"Модемы",

L"Коммутаторы", L"Хабы", L"Флешки", L"Жесткие диски", L"SSD"

});

this->comboBox1->Location = System::Drawing::Point(40, 51);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(310, 21);

this->comboBox1->TabIndex = 0;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label1->Location = System::Drawing::Point(65, 24);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(252, 24);

this->label1->TabIndex = 1;

this->label1->Text = L"Выберите класс устройств";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label2->Location = System::Drawing::Point(87, 98);

Продолжение рис. П3.1

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(196, 24);

this->label2->TabIndex = 2;

this->label2->Text = L"Выберите Wi-Fi сеть";

this->label2->Visible = false;

//

// comboBox2

//

this->comboBox2->FormattingEnabled = true;

this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(5) {

L"CHSU\_FREE", L"CHSU\_WORK", L"IPhone", L"Xiaomi",

L"Неопределенная сеть"

});

this->comboBox2->Location = System::Drawing::Point(40, 137);

this->comboBox2->Name = L"comboBox2";

this->comboBox2->Size = System::Drawing::Size(310, 21);

this->comboBox2->TabIndex = 3;

this->comboBox2->Visible = false;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label3->Location = System::Drawing::Point(65, 98);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(253, 24);

this->label3->TabIndex = 4;

this->label3->Text = L"Выберите мобильную сеть";

this->label3->Visible = false;

//

// comboBox3

//

this->comboBox3->FormattingEnabled = true;

this->comboBox3->Items->AddRange(gcnew cli::array< System::Object^ >(5) { L"MTS RUS", L"Beeline", L"MegaFon", L"Tele2", L"Неопределенный оператор" });

this->comboBox3->Location = System::Drawing::Point(40, 137);

this->comboBox3->Name = L"comboBox3";

this->comboBox3->Size = System::Drawing::Size(310, 21);

this->comboBox3->TabIndex = 5;

this->comboBox3->Visible = false;

//

// comboBox4

//

this->comboBox4->FormattingEnabled = true;

this->comboBox4->Items->AddRange(gcnew cli::array< System::Object^ >(5) {

L"Kingston", L"WD", L"Corsair", L"Неопознанное USB-устройство",

L"Kingston"

});

this->comboBox4->Location = System::Drawing::Point(40, 137);

this->comboBox4->Name = L"comboBox4";

this->comboBox4->Size = System::Drawing::Size(310, 21);

this->comboBox4->TabIndex = 6;

this->comboBox4->Visible = false;

//

// comboBox6

//

this->comboBox6->FormattingEnabled = true;

this->comboBox6->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"Rostelecom", L"Неизвестный модем", L"Rostelecom" });

this->comboBox6->Location = System::Drawing::Point(40, 137);

this->comboBox6->Name = L"comboBox6";

this->comboBox6->Size = System::Drawing::Size(310, 21);

this->comboBox6->TabIndex = 7;

this->comboBox6->Visible = false;

//

// comboBox7

//

this->comboBox7->FormattingEnabled = true;

this->comboBox7->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"iZet", L"Неопознанный коммутатор", L"iZet" });

this->comboBox7->Location = System::Drawing::Point(40, 137);

this->comboBox7->Name = L"comboBox7";

this->comboBox7->Size = System::Drawing::Size(310, 21);

this->comboBox7->TabIndex = 9;

this->comboBox7->Visible = false;

Продолжение рис. П3.1

//

// comboBox8

//

this->comboBox8->FormattingEnabled = true;

this->comboBox8->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"Beeline", L"Неопознанный хаб", L"Beeline" });

this->comboBox8->Location = System::Drawing::Point(40, 137);

this->comboBox8->Name = L"comboBox8";

this->comboBox8->Size = System::Drawing::Size(310, 21);

this->comboBox8->TabIndex = 10;

this->comboBox8->Visible = false;

//

// comboBox9

//

this->comboBox9->FormattingEnabled = true;

this->comboBox9->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"Kingston", L"Неопознанный жесткий диск", L"Kingston" });

this->comboBox9->Location = System::Drawing::Point(40, 137);

this->comboBox9->Name = L"comboBox9";

this->comboBox9->Size = System::Drawing::Size(310, 21);

this->comboBox9->TabIndex = 11;

this->comboBox9->Visible = false;

//

// comboBox10

//

this->comboBox10->FormattingEnabled = true;

this->comboBox10->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"WD", L"Неопознанный SSD-диск", L"WD" });

this->comboBox10->Location = System::Drawing::Point(40, 137);

this->comboBox10->Name = L"comboBox10";

this->comboBox10->Size = System::Drawing::Size(310, 21);

this->comboBox10->TabIndex = 12;

this->comboBox10->Visible = false;

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label4->Location = System::Drawing::Point(107, 98);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(165, 24);

this->label4->TabIndex = 13;

this->label4->Text = L"Выберите модем";

this->label4->Visible = false;

//

// label5

//

this->label5->Location = System::Drawing::Point(0, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(100, 23);

this->label5->TabIndex = 51;

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label6->Location = System::Drawing::Point(87, 98);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(212, 24);

this->label6->TabIndex = 15;

this->label6->Text = L"Выберите коммутатор";

this->label6->Visible = false;

//

// label7

//

this->label7->AutoSize = true;

this->label7->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label7->Location = System::Drawing::Point(121, 98);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(136, 24);

this->label7->TabIndex = 16;

this->label7->Text = L"Выберите хаб";

this->label7->Visible = false;

//

// label8

Продолжение рис. П3.1

//

this->label8->AutoSize = true;

this->label8->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label8->Location = System::Drawing::Point(87, 98);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(215, 24);

this->label8->TabIndex = 17;

this->label8->Text = L"Выберите USB-флешку";

this->label8->Visible = false;

//

// label9

//

this->label9->AutoSize = true;

this->label9->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label9->Location = System::Drawing::Point(74, 98);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(228, 24);

this->label9->TabIndex = 18;

this->label9->Text = L"Выберите жесткий диск";

this->label9->Visible = false;

//

// label10

//

this->label10->AutoSize = true;

this->label10->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label10->Location = System::Drawing::Point(93, 98);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(190, 24);

this->label10->TabIndex = 19;

this->label10->Text = L"Выберите SSD-диск";

this->label10->Visible = false;

//

// button1

//

this->button1->Location = System::Drawing::Point(374, 49);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 20;

this->button1->Text = L"ОК";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// button2

//

this->button2->Location = System::Drawing::Point(374, 135);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(75, 23);

this->button2->TabIndex = 21;

this->button2->Text = L"ОК";

this->button2->UseVisualStyleBackColor = true;

this->button2->Visible = false;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// label11

//

this->label11->AutoSize = true;

this->label11->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label11->Location = System::Drawing::Point(115, 172);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(157, 24);

this->label11->TabIndex = 22;

this->label11->Text = L"Введите пароль";

this->label11->Visible = false;

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(40, 208);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(310, 20);

this->textBox1->TabIndex = 23;

this->textBox1->Visible = false;

//

// button3

//

this->button3->Location = System::Drawing::Point(374, 206);

Продолжение рис. П3.1

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(75, 23);

this->button3->TabIndex = 24;

this->button3->Text = L"ОК";

this->button3->UseVisualStyleBackColor = true;

this->button3->Visible = false;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// label12

//

this->label12->AutoSize = true;

this->label12->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label12->ForeColor = System::Drawing::Color::Green;

this->label12->Location = System::Drawing::Point(80, 248);

this->label12->Name = L"label12";

this->label12->Size = System::Drawing::Size(238, 24);

this->label12->TabIndex = 25;

this->label12->Text = L"Подключение выполнено";

this->label12->Visible = false;

//

// label13

//

this->label13->AutoSize = true;

this->label13->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label13->ForeColor = System::Drawing::Color::Red;

this->label13->Location = System::Drawing::Point(112, 248);

this->label13->Name = L"label13";

this->label13->Size = System::Drawing::Size(171, 24);

this->label13->TabIndex = 26;

this->label13->Text = L"Пароль неверный";

this->label13->Visible = false;

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(299, 291);

this->textBox2->Name = L"textBox2";

this->textBox2->ReadOnly = true;

this->textBox2->Size = System::Drawing::Size(51, 20);

this->textBox2->TabIndex = 28;

this->textBox2->Text = L"0";

//

// textBox3

//

this->textBox3->Location = System::Drawing::Point(299, 329);

this->textBox3->Name = L"textBox3";

this->textBox3->ReadOnly = true;

this->textBox3->Size = System::Drawing::Size(51, 20);

this->textBox3->TabIndex = 29;

this->textBox3->Text = L"0";

//

// label14

//

this->label14->AutoSize = true;

this->label14->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label14->Location = System::Drawing::Point(74, 286);

this->label14->Name = L"label14";

this->label14->Size = System::Drawing::Size(219, 24);

this->label14->TabIndex = 30;

this->label14->Text = L"Удачных подключений:";

//

// label15

//

this->label15->AutoSize = true;

this->label15->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label15->Location = System::Drawing::Point(53, 324);

this->label15->Name = L"label15";

this->label15->Size = System::Drawing::Size(240, 24);

this->label15->TabIndex = 31;

this->label15->Text = L"Неудачных подключений:";

//

// textBox4

//

this->textBox4->Location = System::Drawing::Point(299, 370);

this->textBox4->Name = L"textBox4";

this->textBox4->ReadOnly = true;

Продолжение рис. П3.1

this->textBox4->Size = System::Drawing::Size(51, 20);

this->textBox4->TabIndex = 32;

this->textBox4->Text = L"0";

//

// label16

//

this->label16->AutoSize = true;

this->label16->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label16->Location = System::Drawing::Point(116, 366);

this->label16->Name = L"label16";

this->label16->Size = System::Drawing::Size(177, 24);

this->label16->TabIndex = 33;

this->label16->Text = L"Передано данных:";

//

// label17

//

this->label17->AutoSize = true;

this->label17->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label17->Location = System::Drawing::Point(42, 407);

this->label17->Name = L"label17";

this->label17->Size = System::Drawing::Size(251, 24);

this->label17->TabIndex = 34;

this->label17->Text = L"Общее время соединения:";

//

// textBox5

//

this->textBox5->Location = System::Drawing::Point(299, 409);

this->textBox5->Name = L"textBox5";

this->textBox5->ReadOnly = true;

this->textBox5->Size = System::Drawing::Size(51, 20);

this->textBox5->TabIndex = 35;

this->textBox5->Text = L"0";

//

// button4

//

this->button4->Location = System::Drawing::Point(156, 444);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(75, 23);

this->button4->TabIndex = 36;

this->button4->Text = L"Сброс";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4\_Click);

//

// comboBox11

//

this->comboBox11->FormattingEnabled = true;

this->comboBox11->Items->AddRange(gcnew cli::array< System::Object^ >(5) {

L"В хронологическом порядке (от старых к новым)",

L"В обратном хронологическому порядке (от новых к старым)", L"В алфавитном порядке (A-Z)", L"В обратном алфавитному порядке (Z-A)",

L"Поиск"

});

this->comboBox11->Location = System::Drawing::Point(618, 51);

this->comboBox11->Name = L"comboBox11";

this->comboBox11->Size = System::Drawing::Size(318, 21);

this->comboBox11->TabIndex = 38;

//

// richTextBox1

//

this->richTextBox1->Location = System::Drawing::Point(505, 135);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(502, 164);

this->richTextBox1->TabIndex = 39;

this->richTextBox1->Text = L"";

//

// button5

//

this->button5->Location = System::Drawing::Point(559, 305);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(196, 23);

this->button5->TabIndex = 40;

this->button5->Text = L"Вывести для выбранного класса";

this->button5->UseVisualStyleBackColor = true;

this->button5->Click += gcnew System::EventHandler(this, &MyForm::button5\_Click);

//

Продолжение рис. П3.1

// label18

//

this->label18->AutoSize = true;

this->label18->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label18->Location = System::Drawing::Point(631, 24);

this->label18->Name = L"label18";

this->label18->Size = System::Drawing::Size(297, 24);

this->label18->TabIndex = 37;

this->label18->Text = L"Вывести историю подключений";

//

// label19

//

this->label19->AutoSize = true;

this->label19->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label19->Location = System::Drawing::Point(636, 75);

this->label19->Name = L"label19";

this->label19->Size = System::Drawing::Size(286, 24);

this->label19->TabIndex = 42;

this->label19->Text = L"Введите название устройства";

this->label19->Visible = false;

//

// textBox6

//

this->textBox6->Location = System::Drawing::Point(618, 109);

this->textBox6->Name = L"textBox6";

this->textBox6->Size = System::Drawing::Size(318, 20);

this->textBox6->TabIndex = 43;

this->textBox6->Visible = false;

//

// button7

//

this->button7->Location = System::Drawing::Point(942, 107);

this->button7->Name = L"button7";

this->button7->Size = System::Drawing::Size(65, 23);

this->button7->TabIndex = 44;

this->button7->Text = L"ОК";

this->button7->UseVisualStyleBackColor = true;

this->button7->Visible = false;

this->button7->Click += gcnew System::EventHandler(this, &MyForm::button7\_Click);

//

// richTextBox2

//

this->richTextBox2->Location = System::Drawing::Point(505, 378);

this->richTextBox2->Name = L"richTextBox2";

this->richTextBox2->Size = System::Drawing::Size(502, 217);

this->richTextBox2->TabIndex = 47;

this->richTextBox2->Text = L"";

this->richTextBox2->Visible = false;

//

// label20

//

this->label20->AutoSize = true;

this->label20->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label20->Location = System::Drawing::Point(631, 342);

this->label20->Name = L"label20";

this->label20->Size = System::Drawing::Size(279, 24);

this->label20->TabIndex = 48;

this->label20->Text = L"Выберите файл для передачи";

this->label20->Visible = false;

//

// button9

//

this->button9->Location = System::Drawing::Point(559, 628);

this->button9->Name = L"button9";

this->button9->Size = System::Drawing::Size(196, 23);

this->button9->TabIndex = 49;

this->button9->Text = L"Загрузить файл";

this->button9->UseVisualStyleBackColor = true;

this->button9->Visible = false;

this->button9->Click += gcnew System::EventHandler(this, &MyForm::button9\_Click);

//

// richTextBox3

//

this->richTextBox3->Location = System::Drawing::Point(12, 473);

this->richTextBox3->Name = L"richTextBox3";

Продолжение рис. П3.1

this->richTextBox3->Size = System::Drawing::Size(461, 178);

this->richTextBox3->TabIndex = 50;

this->richTextBox3->Text = L"";

//

// button6

//

this->button6->Location = System::Drawing::Point(761, 305);

this->button6->Name = L"button6";

this->button6->Size = System::Drawing::Size(196, 23);

this->button6->TabIndex = 52;

this->button6->Text = L"Вывести всю историю";

this->button6->UseVisualStyleBackColor = true;

this->button6->Click += gcnew System::EventHandler(this, &MyForm::button6\_Click);

//

// button8

//

this->button8->Location = System::Drawing::Point(942, 107);

this->button8->Name = L"button8";

this->button8->Size = System::Drawing::Size(65, 23);

this->button8->TabIndex = 53;

this->button8->Text = L"ОК";

this->button8->UseVisualStyleBackColor = true;

this->button8->Visible = false;

this->button8->Click += gcnew System::EventHandler(this, &MyForm::button8\_Click);

//

// button10

//

this->button10->Location = System::Drawing::Point(761, 628);

this->button10->Name = L"button10";

this->button10->Size = System::Drawing::Size(196, 23);

this->button10->TabIndex = 54;

this->button10->Text = L"Передать данные";

this->button10->UseVisualStyleBackColor = true;

this->button10->Visible = false;

this->button10->Click += gcnew System::EventHandler(this, &MyForm::button10\_Click);

//

// label21

//

this->label21->AutoSize = true;

this->label21->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14));

this->label21->ForeColor = System::Drawing::Color::Green;

this->label21->Location = System::Drawing::Point(670, 598);

this->label21->Name = L"label21";

this->label21->Size = System::Drawing::Size(175, 24);

this->label21->TabIndex = 55;

this->label21->Text = L"Данные переданы";

this->label21->Visible = false;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::SystemColors::Window;

this->ClientSize = System::Drawing::Size(1019, 663);

this->Controls->Add(this->label21);

this->Controls->Add(this->button10);

this->Controls->Add(this->button8);

this->Controls->Add(this->button6);

this->Controls->Add(this->richTextBox3);

this->Controls->Add(this->button9);

this->Controls->Add(this->label20);

this->Controls->Add(this->richTextBox2);

this->Controls->Add(this->button7);

this->Controls->Add(this->textBox6);

this->Controls->Add(this->label19);

this->Controls->Add(this->button5);

this->Controls->Add(this->richTextBox1);

this->Controls->Add(this->comboBox11);

this->Controls->Add(this->label18);

this->Controls->Add(this->button4);

this->Controls->Add(this->textBox5);

this->Controls->Add(this->label17);

this->Controls->Add(this->label16);

this->Controls->Add(this->textBox4);

this->Controls->Add(this->label15);

Продолжение рис. П3.1

this->Controls->Add(this->label14);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label13);

this->Controls->Add(this->label12);

this->Controls->Add(this->button3);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->label11);

this->Controls->Add(this->button2);

this->Controls->Add(this->button1);

this->Controls->Add(this->label10);

this->Controls->Add(this->label9);

this->Controls->Add(this->label8);

this->Controls->Add(this->label7);

this->Controls->Add(this->label6);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->comboBox10);

this->Controls->Add(this->comboBox9);

this->Controls->Add(this->comboBox8);

this->Controls->Add(this->comboBox7);

this->Controls->Add(this->comboBox6);

this->Controls->Add(this->comboBox4);

this->Controls->Add(this->comboBox3);

this->Controls->Add(this->label3);

this->Controls->Add(this->comboBox2);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->comboBox1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;

this->MaximizeBox = false;

this->Name = L"MyForm";

this->Text = L"Устройства передачи данных";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public: int getIndex1() {

return comboBox1->SelectedIndex;

}

public: int getIndex2(int index) {

switch (index) {

case 0: return comboBox2->SelectedIndex;

break;

case 1: return comboBox3->SelectedIndex;

break;

case 2: return comboBox6->SelectedIndex;

break;

case 3: return comboBox7->SelectedIndex;

break;

case 4: return comboBox8->SelectedIndex;

break;

case 5: return comboBox4->SelectedIndex;

break;

case 6: return comboBox9->SelectedIndex;

break;

case 7: return comboBox10->SelectedIndex;

break;

}

}

public: int getIndex3() {

return comboBox11->SelectedIndex;

}

public: void setLucky(String^ value) {

this->textBox2->Text = value;

}

public: void setUnlucky(String^ value) {

this->textBox3->Text = value;

}

public: void setTraffic(String^ value) {

this->textBox4->Text = value;

}

public: void setTime(String^ value) {

Продолжение рис. П3.1

this->textBox5->Text = value;

}

public: void setText(String^ value) {

this->richTextBox1->AppendText(value + "\n");

}

public: void setClear() {

this->richTextBox1->Clear();

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

button2->Visible = true;

this->textBox1->Visible = false;

this->label11->Visible = false;

this->label12->Visible = false;

this->label13->Visible = false;

this->button3->Visible = false;

button9->Visible = false;

label20->Visible = false;

richTextBox2->Visible = false;

switch (comboBox1->SelectedIndex) {

case 0: {

label2->Visible = true;

comboBox2->Visible = true;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = false;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = false;

label7->Visible = false;

comboBox7->Visible = false;

label8->Visible = false;

comboBox8->Visible = false;

label9->Visible = false;

comboBox9->Visible = false;

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 1: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = true;

comboBox3->Visible = true;

label4->Visible = false;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = false;

label7->Visible = false;

comboBox7->Visible = false;

label8->Visible = false;

comboBox8->Visible = false;

label9->Visible = false;

comboBox9->Visible = false;

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 2: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = true;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = true;

label6->Visible = false;

label7->Visible = false;

comboBox7->Visible = false;

label8->Visible = false;

comboBox8->Visible = false;

label9->Visible = false;

comboBox9->Visible = false;

Продолжение рис. П3.1

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 3: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = false;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = true;

label7->Visible = false;

comboBox7->Visible = true;

label8->Visible = false;

comboBox8->Visible = false;

label9->Visible = false;

comboBox9->Visible = false;

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 4: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = false;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = false;

label7->Visible = true;

comboBox7->Visible = false;

label8->Visible = false;

comboBox8->Visible = true;

label9->Visible = false;

comboBox9->Visible = false;

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 5: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = false;

comboBox4->Visible = true;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = false;

label7->Visible = false;

comboBox7->Visible = false;

label8->Visible = true;

comboBox8->Visible = false;

label9->Visible = false;

comboBox9->Visible = false;

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 6: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = false;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = false;

label7->Visible = false;

Продолжение рис. П3.1

comboBox7->Visible = false;

label8->Visible = false;

comboBox8->Visible = false;

label9->Visible = true;

comboBox9->Visible = true;

label10->Visible = false;

comboBox10->Visible = false;

}

break;

case 7: {

label2->Visible = false;

comboBox2->Visible = false;

label3->Visible = false;

comboBox3->Visible = false;

label4->Visible = false;

comboBox4->Visible = false;

label5->Visible = false;

comboBox6->Visible = false;

label6->Visible = false;

label7->Visible = false;

comboBox7->Visible = false;

label8->Visible = false;

comboBox8->Visible = false;

label9->Visible = false;

comboBox9->Visible = false;

label10->Visible = true;

comboBox10->Visible = true;

}

break;

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button8\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button10\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void button9\_Click(System::Object^ sender, System::EventArgs^ e) {

setlocale(LC\_ALL, "Russian");

OpenFileDialog^ openFile = gcnew OpenFileDialog();

openFile->Filter = "Текстовые файлы (\*.txt)|\*.txt";

String^ fileName;

if (System::Windows::Forms::DialogResult::OK == openFile->ShowDialog()) fileName = openFile->FileName;

label21->Visible = false;

button10->Visible = true;

try {

StreamReader^ file = gcnew StreamReader(fileName, Encoding::GetEncoding("windows-1251"));

richTextBox2->Text = file->ReadToEnd();

}

catch (Exception^ e)

{

richTextBox2->Text = "Ошибка открытия файла!";

}

}

};

}

Текст модуля MyForm.cpp представлен на рис. П3.2.

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

int main(array<String^>^ args)

{

Рис. П3.2. Текст модуля MyForm.cpp

Продолжение рис. П3.2

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Device::MyForm form;

Application::Run(% form);

}

Текст модуля DataDevices\_i.h представлен на рис. П3.3.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

using namespace msclr::interop;

class DataDevices\_i {

public:

virtual string getName() = 0; //геттер для имени

virtual int getTraffic() = 0; //геттер для траффика

virtual double getTime() = 0; //геттер для времени

virtual int getSpeed() = 0; //геттер для скорости

virtual void setData(string value) = 0; //сеттер для передачи данных

virtual ~DataDevices\_i() = default; // Виртуальный деструктор

};

Рис. П3.3. Текст модуля DataDevices\_i.h

Текст модуля DataDevices.h представлен на рис. П3.4.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "DataDevices\_i.h"

using namespace std;

class DataDevices : public DataDevices\_i {

protected:

string name, data;

static string work;

int speed, traffic;

double time;

public:

string getName() { //геттер для имени

return name;

}

int getTraffic() { //геттер для траффика

return traffic;

}

double getTime() { //геттер для времени

return time;

}

int getSpeed() { //геттер для скорости

return speed;

}

Рис. П3.4. Текст модуля DataDevices.h

Продолжение рис. П3.4

void setData(string value) { //cеттер для передачи данных

this->data = value;

}

};

Текст модуля Wireless.h представлен на рис. П3.5.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "DataDevices.h"

using namespace std;

class Wireless : public DataDevices {

protected:

int radius;

double frequency;

};

Рис. П3.5. Текст модуля Wireless.h

Текст модуля WiFi\_router.h представлен на рис. П3.6.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Wireless.h"

using namespace std;

using namespace System;

enum security { WEP, WPA, WPA2, WPA3, NoProtection };

class WiFi\_router : public Wireless {

private:

string password;

security protection;

public:

WiFi\_router(string valueName, string valuePassword, security valueProtection, int valueSpeed, int valueRadius, double valueFrequency) {

this->name = valueName; //конструктор со всеми параметрами

this->password = valuePassword;

this->protection = valueProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi\_router() {

this->name = "Неопределенная сеть"; //конструктор по умолчанию

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

Рис. П3.6. Текст модуля WiFi\_router.h

Продолжение рис. П3.6

this->time = rand() / 100.0;

}

WiFi\_router(string valueName) {

this->name = valueName; //конструктор с 1 параметром

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi\_router(double valueFrequency, string valueName) {

this->name = valueName; //конструктор с 2 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi\_router(int valueSpeed, int valueRadius, string valueName) {

this->name = valueName; //конструктор с 3 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi\_router(const WiFi\_router& obj) {

this->name = obj.name; //копирующий конструктор

this->password = obj.password;

this->protection = obj.protection;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getPassword() { //геттер для пароля

return password;

}

string getProtection() { //геттер для протокола безопасности

switch (protection) {

case WEP: return "WEP";

break;

case WPA: return "WPA";

break;

case WPA2: return "WPA2";

break;

case WPA3: return "WPA3";

break;

case NoProtection: return "Без защиты";

break;

default: return "N/A";

break;

}

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Вы успешно подключились к WiFi-сети " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация о сети\n";

info += "Скорость: " + speed + " Мбит/с\n";

info += "Частота: " + frequency + " ГГц\n";

Продолжение рис. П3.6

info += “Радиус покрытия: “ + radius + “ м\n”;

info += “Протокол безопасности: “ + msclr::interop::marshal\_as<String^>(getProtection()) + “\n”;

info += “Cеть работает: “ + msclr::interop::marshal\_as<String^>(work) + “\n”;

info += «Передано данных: « + traffic + « Мб\n»;

info += «Время соединения: « + time + « сек\n»;

ofstream result;

result.open(“WiFi.txt”);

result << marshal\_as<string>(info);

result.close();

return info;

}

~WiFi\_router() { //деструктор

}

};

Текст модуля Mobile\_antenna.h представлен на рис. П3.7.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Wireless.h"

using namespace std;

using namespace System;

enum generation { GPRS, Edge, \_2G, \_3G, \_4G, \_5G, unknown };

class Mobile\_antenna : public Wireless {

private:

generation type;

int signal;

public:

Mobile\_antenna(string valueName, generation valueType, int valueSpeed, int valueRadius, double valueFrequency) {

this->name = valueName; //конструктор со всеми параметрами

this->type = valueType;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile\_antenna() {

this->name = "Неопределенный оператор"; //конструктор по умолчанию

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile\_antenna(string valueName) {

this->name = valueName; //конструктор с 1 параметром

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Рис. П3.7. Текст модуля Mobile\_antenna.h

Продолжение рис. П3.7

Mobile\_antenna(string valueName, generation valueType) {

this->name = valueName; //конструктор с 2 параметрами

this->type = valueType;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile\_antenna(string valueName, generation valueType, int valueSpeed) {

this->name = valueName; //конструктор с 3 параметрами

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile\_antenna(const Mobile\_antenna& obj) {

this->name = obj.name; //конструктор с 3 параметрами

this->type = obj.type;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

string getType() { //геттер для поколения сети

switch (type) {

case GPRS: return "GPRS";

break;

case Edge: return "Edge";

break;

case \_2G: return "2G";

break;

case \_3G: return "3G";

break;

case \_4G: return "4G";

break;

case \_5G: return "5G";

break;

case unknown: return "unknown";

break;

default: return "N/A";

break;

}

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Вы успешно подключились к мобильной сети " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация о сети\n";

info += "Скорость: " + speed + " Мбит/с\n";

info += "Уровень сигнала: " + signal + "\n";

info += "Частота: " + frequency + " ГГц\n";

info += "Радиус покрытия: " + radius + " м\n";

info += "Поколение сети: " + msclr::interop::marshal\_as<String^>(getType()) + "\n";

info += "Cеть работает: " + msclr::interop::marshal\_as<String^>(work) + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("Mobile.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~Mobile\_antenna() { //деструктор

}

};

Текст модуля Wired.h представлен на рис. П3.8.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "DataDevices.h"

using namespace std;

enum cable { Twisted, Coaxial, Optical, other };

class Wired : public DataDevices {

protected:

cable cord;

int length\_cable;

public:

string getCord() { //геттер для кабеля

switch (cord) {

case Twisted: return "Витая пара";

case Coaxial: return "Коаксиальный кабель";

case Optical: return "Оптоволокно";

case other: return "Другой тип кабеля";

default: return "N/A";

}

}

};

Рис. П3.8. Текст модуля Wired.h

Текст модуля Modem.h представлен на рис. П3.9.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Wired.h"

using namespace std;

using namespace System;

enum line { Dial\_Up, ADSL, Ethernet, NoConnect };

class Modem : public Wired {

private:

line connect;

public:

Modem(string valueName, cable valueCord, int valueSpeed, int valueLength, line valueConnect) {

this->name = valueName; //конструктор со всеми параметрами

this->speed = valueSpeed;

this->cord = valueCord;

this->connect = valueConnect;

this->length\_cable = valueLength;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Modem() {

this->name = "Неизвестный роутер"; //конструктор по умолчанию

this->speed = 0;

this->cord = other;

this->connect = NoConnect;

this->length\_cable = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Рис. П3.9. Текст модуля Wired.h

Продолжение рис. П3.9

Modem(const Modem& obj) {

this->name = obj.name; //копирующий конструктор

this->speed = obj.speed;

this->cord = obj.cord;

this->connect = obj.connect;

this->length\_cable = obj.length\_cable;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getLine() { //геттер для типа линии

switch (connect) {

case Dial\_Up: return "Dial Up";

break;

case ADSL: return "ADSL";

break;

case Ethernet: return "Ethernet";

break;

case NoConnect: return "Другой тип линии";

break;

default: return "N/A";

break;

}

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Вы успешно подключились к модему " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация о сети\n";

info += "Скорость: " + speed + " Мбит/с\n";

info += "Тип подключения: " + msclr::interop::marshal\_as<String^>(getCord()) + "\n";

info += "Тип линии: " + msclr::interop::marshal\_as<String^>(getLine()) + "\n";

info += "Длина кабеля: " + length\_cable + " м\n";

info += "Cеть работает: " + msclr::interop::marshal\_as<String^>(work) + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("Modem.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~Modem() { //деструктор

}

};

Текст модуля Switch.h представлен на рис. П3.10.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Wired.h"

using namespace std;

using namespace System;

class Switch : public Wired {

private:

int ports;

public:

Switch(string valueName, cable valueCord, int valueSpeed, int valueLength, int valuePorts) {

this->name = valueName; //конструктор со всеми параметрами

this->speed = valueSpeed;

this->cord = valueCord;

Рис. П3.10. Текст модуля Switch.h

Продолжение рис. П3.10

this->length\_cable = valueLength;

this->ports = valuePorts;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Switch() {

this->name = "Неизвестный коммутатор"; //конструктор по умолчанию

this->speed = 0;

this->cord = other;

this->length\_cable = 0;

this->ports = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Switch(const Switch& obj) {

this->name = obj.name; //копирующий конструктор

this->speed = obj.speed;

this->cord = obj.cord;

this->length\_cable = obj.length\_cable;

this->ports = obj.ports;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Вы успешно подключились к коммутатору " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация о сети\n";

info += "Скорость: " + speed + " Мбит/с\n";

info += "Тип подключения: " + msclr::interop::marshal\_as<String^>(getCord()) + "\n";

info += "Длина кабеля: " + length\_cable + " м\n";

info += "Количество портов: " + ports + "\n";

info += "Cеть работает: " + msclr::interop::marshal\_as<String^>(work) + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("Switch.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~Switch() { //деструктор

}

};

Текст модуля Hub.h представлен на рис. П3.11.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Wired.h"

using namespace std;

using namespace System;

class Hub : public Wired {

private:

int ports;

public:

Hub(string valueName, cable valueCord, int valueSpeed, int valueLength, int valuePorts) {

this->name = valueName; //конструктор со всеми параметрами

this->speed = valueSpeed;

Рис. П3.11. Текст модуля Hub.h

Продолжение рис. П3.11

this->cord = valueCord;

this->length\_cable = valueLength;

this->ports = valuePorts;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Hub() {

this->name = "Неизвестный хаб"; //конструктор по умолчанию

this->speed = 0;

this->cord = other;

this->length\_cable = 0;

this->ports = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Hub(const Hub& obj) {

this->name = obj.name; //копирующий конструктор

this->speed = obj.speed;

this->cord = obj.cord;

this->length\_cable = obj.length\_cable;

this->ports = obj.ports;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Вы успешно подключились к хабу " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация о сети\n";

info += "Скорость: " + speed + " Мбит/с\n";

info += "Тип подключения: " + msclr::interop::marshal\_as<String^>(getCord()) + "\n";

info += "Длина кабеля: " + length\_cable + " м\n";

info += "Количество портов: " + ports + "\n";

info += "Cеть работает: " + msclr::interop::marshal\_as<String^>(work) + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("Hub.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~Hub() { //деструктор

}

};

Текст модуля Storage.h представлен на рис. П3.12.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "DataDevices.h"

using namespace std;

class Storage : public DataDevices {

protected:

int capacity;

};

Рис. П3.12. Текст модуля Storage.h

Текст модуля Flash.h представлен на рис. П3.13.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Storage.h"

using namespace std;

using namespace System;

class Flash : public Storage {

private:

double version;

char connector;

public:

Flash(string valueName, int valueCapacity, double valueVersion, char valueConnector, int valueSpeed) {

this->name = valueName;

this->speed = valueSpeed;

this->capacity = valueCapacity;

this->version = valueVersion;

this->connector = valueConnector;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Flash() {

this->name = "Неопознанное USB-устройство";

this->speed = 0;

this->capacity = 0;

this->version = 0.0;

this->connector = 'N';

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Flash(string valueName) {

this->name = valueName;

this->speed = 0;

this->capacity = 0;

this->version = 0.0;

this->connector = 'N';

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Flash(string valueName, int valueCapacity) {

this->name = valueName;

this->speed = 0;

this->capacity = valueCapacity;

this->version = 0.0;

this->connector = 'N';

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

Flash(const Flash& obj) {

this->name = obj.name;

this->speed = obj.speed;

this->capacity = obj.capacity;

this->version = obj.version;

this->connector = obj.connector;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Подключено USB-устройство " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация об устройстве\n";

info += "Скорость: " + speed + " Мб/с\n";

Рис. П3.13. Текст модуля Storage.h

Продолжение рис. П3.13

info += "Объем USB-накопителя: " + capacity + " Гб\n";

info += "Версия USB: " + version + "\n";

info += "Разъем: Type-" + connector + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("Flash.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~Flash() { //деструктор

}

};

Текст модуля HDD.h представлен на рис. П3.14.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Storage.h"

using namespace std;

using namespace System;

enum connect\_type { SATA, IDE, SCSI, SAS, M2, U2, PCIe, other\_type };

class HDD : public Storage {

private:

connect\_type connector;

public:

HDD(string valueName, int valueCapacity, connect\_type valueConnector, int valueSpeed) {

this->name = valueName;

this->speed = valueSpeed;

this->capacity = valueCapacity;

this->connector = valueConnector;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

HDD() {

this->name = "Неопознанный жесткий диск";

this->speed = 0;

this->capacity = 0;

this->connector = other\_type;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

HDD(const HDD& obj) {

this->name = obj.name;

this->speed = obj.speed;

this->capacity = obj.capacity;

this->connector = obj.connector;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getType() { //геттер для разъема

switch (connector) {

case SATA: return "SATA";

break;

case IDE: return "IDE";

break;

Рис. П3.14. Текст модуля HDD.h

Продолжение рис. П3.14

case SCSI: return "SCSI";

break;

case SAS: return "SAS";

break;

case M2: return "M2";

break;

case U2: return "U2";

break;

case PCIe: return "PCIe";

break;

case other\_type: return "Другой разъем";

break;

default: return "N/A";

break;

}

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Подключен жесткий диск " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация об устройстве\n";

info += "Скорость: " + speed + " Мб/с\n";

info += "Объем диска: " + capacity + " Гб\n";

info += "Разъем: " + msclr::interop::marshal\_as<String^>(getType()) + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("HDD.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~HDD() { //деструктор

}

};

Текст модуля SSD.h представлен на рис. П3.15.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <msclr\marshal.h>

#include "Storage.h"

using namespace std;

using namespace System;

enum connect\_type2 { \_SATA, NVMe, \_M2, \_U2, \_PCIe, \_other\_type };

class SSD : public Storage {

private:

connect\_type2 connector;

public:

SSD(string valueName, int valueCapacity, connect\_type2 valueConnector, int valueSpeed) {

this->name = valueName;

this->speed = valueSpeed;

this->capacity = valueCapacity;

this->connector = valueConnector;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

SSD() {

this->name = "Неопознанный SSD-диск";

this->speed = 0;

Рис. П3.15. Текст модуля SSD.h

Продолжение рис. П3.15

this->capacity = 0;

this->connector = \_other\_type;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

SSD(const SSD& obj) {

this->name = obj.name;

this->speed = obj.speed;

this->capacity = obj.capacity;

this->connector = obj.connector;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getType() { //геттер для разъема

switch (connector) {

case \_SATA: return "SATA";

break;

case NVMe: return "NVMe";

break;

case \_M2: return "M2";

break;

case \_U2: return "U2";

break;

case \_PCIe: return "PCIe";

break;

case \_other\_type: return "Другой разъем";

break;

default: return "N/A";

break;

}

}

String^ getInfo() { //вывод информации о сети

String^ info;

info += "Подключен SSD-диск " + msclr::interop::marshal\_as<String^>(name) + "\n\n";

info += "Информация об устройстве\n";

info += "Скорость: " + speed + " Мб/с\n";

info += "Объем диска: " + capacity + " Гб\n";

info += "Разъем: " + msclr::interop::marshal\_as<String^>(getType()) + "\n";

info += "Передано данных: " + traffic + " Мб\n";

info += "Время соединения: " + time + " сек\n";

ofstream result;

result.open("SSD.txt");

result << marshal\_as<string>(info);

result.close();

return info;

}

~SSD() { //деструктор

}

};

Текст модуля KP.cpp представлен на рис. П3.16.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

#include "Queue.h"

#include "DataDevices\_i.h"

#include "Exception.h"

#include "Vector.h"

#include "MyForm.h"

using namespace std;

Рис. П3.16. Текст модуля KP.cpp

Продолжение рис. П3.16

int lucky = 0, unlucky = 0, traffic = 0;

double times = 0;

string name;

string DataDevices::work = "Да";

Vector<DataDevices\_i\*> v;

Queue<WiFi\_router> q\_wifi;

Queue<Mobile\_antenna> q\_mob;

Queue<Modem> q\_rout;

Queue<Switch> q\_sw;

Queue<Hub> q\_hub;

Queue<Flash> q\_fl;

Queue<HDD> q\_hdd;

Queue<SSD> q\_ssd;

WiFi\_router WiFi\_1("CHSU\_FREE", "0", security::NoProtection, 100, 70, 5); //описание доступных WiFi-сетей

WiFi\_router WiFi\_2("CHSU\_WORK", "1234567890", security::WPA3, 120, 50, 5);

WiFi\_router WiFi\_3("IPhone", "ghfdgldklkl", security::WPA2, 80, 40, 2.4);

WiFi\_router WiFi\_4("Xiaomi", "h5hjg54g5", security::WPA2, 70, 40, 2.4);

WiFi\_router WiFi\_5;

Mobile\_antenna Mobile\_1("MTS RUS", generation::\_5G, 70, 100, 5); //описание доступных мобильных сетей

Mobile\_antenna Mobile\_2("Beeline", generation::\_2G, 50, 90, 5);

Mobile\_antenna Mobile\_3("MegaFon", generation::\_4G, 80, 100, 5);

Mobile\_antenna Mobile\_4("Tele2", generation::\_3G, 50, 80, 5);

Mobile\_antenna Mobile\_5;

Modem Router\_1("Rostelecom", Optical, 200, 5, line::Ethernet);

Modem Router\_2;

Modem Router\_3(Router\_1);

Switch Switch\_1("iZet", Coaxial, 100, 3, 8);

Switch Switch\_2;

Switch Switch\_3(Switch\_1);

Hub Hub\_1("Beeline", Twisted, 150, 4, 32);

Hub Hub\_2;

Hub Hub\_3(Hub\_1);

Flash USB\_1("Kingston", 8, 3.1, 'A', 30);

Flash USB\_2("WD");

Flash USB\_3("Corsair", 2);

Flash USB\_4;

HDD HDD\_1("Kingston", 1024, connect\_type::SATA, 100);

HDD HDD\_2;

HDD HDD\_3(HDD\_1);

SSD SSD\_1("WD", 512, connect\_type2::\_M2, 500);

SSD SSD\_2;

SSD SSD\_3(SSD\_1);

template <class T>

String^ authorization(T& obj, Queue<T>& q) {

traffic += obj.getTraffic();

times += obj.getTime();

q.add(obj);

v.add(&obj);

return obj.getInfo();

}

System::Void Device::MyForm::button2\_Click(System::Object^ sender, System::EventArgs^ e) {

Device::MyForm^ myForm = this;

String^ info;

label21->Visible = false;

button10->Visible = false;

if (Device::MyForm::getIndex1() \* 10 + Device::MyForm::getIndex2(Device::MyForm::getIndex1()) >= 10) {

lucky++;

button9->Visible = true;

label20->Visible = true;

richTextBox2->Visible = true;

label12->Visible = true;

}

setLucky(System::Convert::ToString(lucky));

Продолжение рис. П3.16

switch (Device::MyForm::getIndex1() \* 10 + Device::MyForm::getIndex2(Device::MyForm::getIndex1())) {

case 0:

case 1:

case 2:

case 3:

case 4: {

textBox1->Visible = true;

label11->Visible = true;

button3->Visible = true;

}

break;

case 10: info = authorization(Mobile\_1, q\_mob);

richTextBox3->Text = info + "\nИнформация записана в файл Mobile.txt";

break;

case 11: info = authorization(Mobile\_2, q\_mob);

richTextBox3->Text = info + "\nИнформация записана в файл Mobile.txt";

break;

case 12: info = authorization(Mobile\_3, q\_mob);

richTextBox3->Text = info + "\nИнформация записана в файл Mobile.txt";

break;

case 13: info = authorization(Mobile\_4, q\_mob);

richTextBox3->Text = info + "\nИнформация записана в файл Mobile.txt";

break;

case 14: info = authorization(Mobile\_5, q\_mob);

richTextBox3->Text = info + "\nИнформация записана в файл Mobile.txt";

break;

case 20: info = authorization(Router\_1, q\_rout);

richTextBox3->Text = info + "\nИнформация записана в файл Modem.txt";

break;

case 21: info = authorization(Router\_2, q\_rout);

richTextBox3->Text = info + "\nИнформация записана в файл Modem.txt";

break;

case 22: info = authorization(Router\_3, q\_rout);

richTextBox3->Text = info + "\nИнформация записана в файл Modem.txt";

break;

case 30: info = authorization(Switch\_1, q\_sw);

richTextBox3->Text = info + "\nИнформация записана в файл Switch.txt";

break;

case 31: info = authorization(Switch\_2, q\_sw);

richTextBox3->Text = info + "\nИнформация записана в файл Switch.txt";

break;

case 32: info = authorization(Switch\_3, q\_sw);

richTextBox3->Text = info + "\nИнформация записана в файл Switch.txt";

break;

case 40: info = authorization(Hub\_1, q\_hub);

richTextBox3->Text = info + "\nИнформация записана в файл Hub.txt";

break;

case 41: info = authorization(Hub\_2, q\_hub);

richTextBox3->Text = info + "\nИнформация записана в файл Hub.txt";

break;

case 42: info = authorization(Hub\_3, q\_hub);

richTextBox3->Text = info + "\nИнформация записана в файл Hub.txt";

break;

case 50: info = authorization(USB\_1, q\_fl);

richTextBox3->Text = info + "\nИнформация записана в файл Flash.txt";

break;

case 51: info = authorization(USB\_2, q\_fl);

richTextBox3->Text = info + "\nИнформация записана в файл Flash.txt";

break;

case 52: info = authorization(USB\_3, q\_fl);

richTextBox3->Text = info + "\nИнформация записана в файл Flash.txt";

break;

case 53: info = authorization(USB\_4, q\_fl);

richTextBox3->Text = info + "\nИнформация записана в файл Flash.txt";

break;

case 60: info = authorization(HDD\_1, q\_hdd);

richTextBox3->Text = info + "\nИнформация записана в файл HDD.txt";

break;

case 61: info = authorization(HDD\_2, q\_hdd);

richTextBox3->Text = info + "\nИнформация записана в файл HDD.txt";

break;

case 62: info = authorization(HDD\_3, q\_hdd);

richTextBox3->Text = info + "\nИнформация записана в файл HDD.txt";

break;

case 70: info = authorization(SSD\_1, q\_ssd);

Продолжение рис. П3.16

richTextBox3->Text = info + "\nИнформация записана в файл SSD.txt";

break;

case 71: info = authorization(SSD\_2, q\_ssd);

richTextBox3->Text = info + "\nИнформация записана в файл SSD.txt";

break;

case 72: info = authorization(SSD\_3, q\_ssd);

richTextBox3->Text = info + "\nИнформация записана в файл SSD.txt";

break;

}

Device::MyForm::setTraffic(System::Convert::ToString(traffic));

Device::MyForm::setTime(System::Convert::ToString(times));

}

System::Void Device::MyForm::button3\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ info;

switch (Device::MyForm::getIndex1() \* 10 + Device::MyForm::getIndex2(Device::MyForm::getIndex1())) {

case 0: {

try {

if (WiFi\_1.getPassword() == marshal\_as<string>(textBox1->Text)) {

label12->Visible = true;

label13->Visible = false;

button9->Visible = true;

label20->Visible = true;

richTextBox2->Visible = true;

lucky++;

setLucky(System::Convert::ToString(lucky));

info = authorization(WiFi\_1, q\_wifi);

richTextBox3->Text = info + "\nИнформация записана в файл WiFi.txt";

}

else {

throw MyException("Пароль неверный!");

}

}

catch (MyException& ex) {

label12->Visible = false;

label13->Visible = true;

button9->Visible = false;

label20->Visible = false;

richTextBox2->Visible = false;

unlucky++;

setUnlucky(System::Convert::ToString(unlucky));

}

}

break;

case 1: {

try {

if (WiFi\_2.getPassword() == marshal\_as<string>(textBox1->Text)) {

label12->Visible = true;

label13->Visible = false;

button9->Visible = true;

label20->Visible = true;

richTextBox2->Visible = true;

lucky++;

setLucky(System::Convert::ToString(lucky));

info = authorization(WiFi\_2, q\_wifi);

richTextBox3->Text = info + "\nИнформация записана в файл WiFi.txt";

}

else {

throw MyException("Пароль неверный!");

}

}

catch (MyException& ex) {

label12->Visible = false;

label13->Visible = true;

button9->Visible = false;

label20->Visible = false;

richTextBox2->Visible = false;

unlucky++;

setUnlucky(System::Convert::ToString(unlucky));

}

}

break;

Продолжение рис. П3.16

case 2: {

try {

if (WiFi\_3.getPassword() == marshal\_as<string>(textBox1->Text)) {

label12->Visible = true;

label13->Visible = false;

button9->Visible = true;

label20->Visible = true;

richTextBox2->Visible = true;

lucky++;

setLucky(System::Convert::ToString(lucky));

info = authorization(WiFi\_3, q\_wifi);

richTextBox3->Text = info + "\nИнформация записана в файл WiFi.txt";

}

else {

throw MyException("Пароль неверный!");

}

}

catch (MyException& ex) {

label12->Visible = false;

label13->Visible = true;

button9->Visible = false;

label20->Visible = false;

richTextBox2->Visible = false;

unlucky++;

setUnlucky(System::Convert::ToString(unlucky));

}

}

break;

case 3: {

try {

if (WiFi\_4.getPassword() == marshal\_as<string>(textBox1->Text)) {

label12->Visible = true;

label13->Visible = false;

button9->Visible = true;

label20->Visible = true;

richTextBox2->Visible = true;

lucky++;

setLucky(System::Convert::ToString(lucky));

info = authorization(WiFi\_4, q\_wifi);

richTextBox3->Text = info + "\nИнформация записана в файл WiFi.txt";

}

else {

throw MyException("Пароль неверный!");

}

}

catch (MyException& ex) {

label12->Visible = false;

label13->Visible = true;

button9->Visible = false;

label20->Visible = false;

richTextBox2->Visible = false;

unlucky++;

setUnlucky(System::Convert::ToString(unlucky));

}

}

break;

case 4: {

try {

if (WiFi\_5.getPassword() == marshal\_as<string>(textBox1->Text)) {

label12->Visible = true;

label13->Visible = false;

button9->Visible = true;

label20->Visible = true;

richTextBox2->Visible = true;

lucky++;

setLucky(System::Convert::ToString(lucky));

info = authorization(WiFi\_5, q\_wifi);

richTextBox3->Text = info + "\nИнформация записана в файл WiFi.txt";

}

else {

throw MyException("Пароль неверный!");

}

Продолжение рис. П3.16

}

catch (MyException& ex) {

label12->Visible = false;

label13->Visible = true;

button9->Visible = false;

label20->Visible = false;

richTextBox2->Visible = false;

unlucky++;

setUnlucky(System::Convert::ToString(unlucky));

}

}

break;

}

Device::MyForm::setTraffic(System::Convert::ToString(traffic));

Device::MyForm::setTime(System::Convert::ToString(times));

}

System::Void Device::MyForm::button4\_Click(System::Object^ sender, System::EventArgs^ e) {

lucky = 0;

unlucky = 0;

traffic = 0;

times = 0;

setLucky(System::Convert::ToString(lucky));

setUnlucky(System::Convert::ToString(unlucky));

Device::MyForm::setTraffic(System::Convert::ToString(traffic));

Device::MyForm::setTime(System::Convert::ToString(times));

}

System::Void Device::MyForm::button5\_Click(System::Object^ sender, System::EventArgs^ e) {

Device::MyForm^ myForm = this;

if (Device::MyForm::getIndex3() == 4) {

textBox6->Visible = true;

label19->Visible = true;

button7->Visible = true;

button8->Visible = false;

}

else {

textBox6->Visible = false;

label19->Visible = false;

button7->Visible = false;

}

ofstream result;

result.open("result.txt");

string s;

switch (Device::MyForm::getIndex1() \* 10 + Device::MyForm::getIndex3()) {

case 0: {

if (q\_wifi.getSize() <= 20) {

s << q\_wifi;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_wifi;

}

break;

case 1: q\_wifi.print\_NewOld(result, myForm);

break;

case 2: q\_wifi.print\_AZ(result, myForm);

break;

case 3: q\_wifi.print\_ZA(result, myForm);

break;

case 10: {

if (q\_mob.getSize() <= 20) {

s << q\_mob;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_mob;

}

break;

case 11: q\_mob.print\_NewOld(result, myForm);

break;

case 12: q\_mob.print\_AZ(result, myForm);

break;

case 13: q\_mob.print\_ZA(result, myForm);

break;

case 20: {

if (q\_rout.getSize() <= 20) {

Продолжение рис. П3.16

s << q\_rout;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_wifi;

}

break;

case 21: q\_rout.print\_NewOld(result, myForm);

break;

case 22: q\_rout.print\_AZ(result, myForm);

break;

case 23: q\_rout.print\_ZA(result, myForm);

break;

case 30: {

if (q\_sw.getSize() <= 20) {

s << q\_sw;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_sw;

}

break;

case 31: q\_sw.print\_NewOld(result, myForm);

break;

case 32: q\_sw.print\_AZ(result, myForm);

break;

case 33: q\_sw.print\_ZA(result, myForm);

break;

case 40: {

if (q\_hub.getSize() <= 20) {

s << q\_hub;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_hub;

}

break;

case 41: q\_hub.print\_NewOld(result, myForm);

break;

case 42: q\_hub.print\_AZ(result, myForm);

break;

case 43: q\_hub.print\_ZA(result, myForm);

break;

case 50: {

if (q\_fl.getSize() <= 20) {

s << q\_wifi;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_fl;

}

break;

case 51: q\_fl.print\_NewOld(result, myForm);

break;

case 52: q\_fl.print\_AZ(result, myForm);

break;

case 53: q\_fl.print\_ZA(result, myForm);

break;

case 60: {

if (q\_hdd.getSize() <= 20) {

s << q\_hdd;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_hdd;

}

break;

case 61: q\_hdd.print\_NewOld(result, myForm);

break;

case 62: q\_hdd.print\_AZ(result, myForm);

break;

case 63: q\_hdd.print\_ZA(result, myForm);

break;

case 70: {

if (q\_ssd.getSize() <= 20) {

s << q\_ssd;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << q\_ssd;

}

Продолжение рис. П3.16

break;

case 71: q\_ssd.print\_NewOld(result, myForm);

break;

case 72: q\_ssd.print\_AZ(result, myForm);

break;

case 73: q\_ssd.print\_ZA(result, myForm);

break;

}

result.close();

}

System::Void Device::MyForm::button6\_Click(System::Object^ sender, System::EventArgs^ e) {

Device::MyForm^ myForm = this;

ofstream result;

result.open("result.txt");

string s;

if (Device::MyForm::getIndex3() == 4) {

textBox6->Visible = true;

label19->Visible = true;

button7->Visible = false;

button8->Visible = true;

}

else {

textBox6->Visible = false;

label19->Visible = false;

button8->Visible = false;

}

switch (Device::MyForm::getIndex3()) {

case 0: {

if (v.getSize() == 0) {

richTextBox1->Text = "Подключений не было";

}

else if (v.getSize() <= 20) {

s << v;

richTextBox1->Text = marshal\_as<String^>(s);

}

else result << v;

}

break;

case 1: v.print\_NewOld(result, myForm);

break;

case 2: v.print\_AZ(result, myForm);

break;

case 3: v.print\_ZA(result, myForm);

break;

}

result.close();

}

System::Void Device::MyForm::button7\_Click(System::Object^ sender, System::EventArgs^ e) {

Device::MyForm^ myForm = this;

switch (Device::MyForm::getIndex1()) {

case 0: {

q\_wifi.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

case 1: {

q\_mob.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

case 2: {

q\_rout.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

case 3: {

q\_sw.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

case 4: {

q\_hub.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

case 5: {

q\_fl.search(marshal\_as<string>(textBox6->Text), myForm);

}

Продолжение рис. П3.16

break;

case 6: {

q\_hdd.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

case 7: {

q\_ssd.search(marshal\_as<string>(textBox6->Text), myForm);

}

break;

}

}

System::Void Device::MyForm::button8\_Click(System::Object^ sender, System::EventArgs^ e) {

Device::MyForm^ myForm = this;

v.search(marshal\_as<string>(textBox6->Text), myForm);

}

System::Void Device::MyForm::button10\_Click(System::Object^ sender, System::EventArgs^ e) {

Device::MyForm^ myForm = this;

v.broadcast(richTextBox2->Text, myForm);

label21->Visible = true;

}

Текст модуля Container.h представлен на рис. П3.17.

#pragma once

using namespace std;

class Container {

protected:

void\*\* data;

size\_t size;

size\_t capacity;

void resize(size\_t new\_capacity) {

void\*\* new\_data = new void\* [new\_capacity];

for (size\_t i = 0; i < size; ++i) {

new\_data[i] = data[i];

}

delete[] data;

data = new\_data;

capacity = new\_capacity;

}

public:

Container() {

data = nullptr;

size = 0;

capacity = 0;

}

virtual ~Container() {

delete[] data;

}

};

Рис. П3.17. Текст модуля Container.h

Текст модуля Vector.h представлен на рис. П3.18.

#pragma once

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <algorithm>

#include <msclr/marshal\_cppstd.h>

#include "Container.h"

#include "DataDevices\_i.h"

using namespace std;

using namespace msclr::interop;

template <class T>

class Vector : private Container {

public:

Vector() {

capacity = 10;

data = new void\* [capacity];

size = 0;

}

Vector(const Vector& other) {

capacity = other.capacity;

size = other.size;

data = new void\* [capacity];

for (size\_t i = 0; i < size; ++i) {

data[i] = new T(\*static\_cast<T\*>(other.data[i]));

}

}

~Vector() {

clear();

}

void clear() {

for (size\_t i = 0; i < size; ++i) {

delete static\_cast<T\*>(data[i]);

}

delete[] data;

data = nullptr;

size = 0;

capacity = 0;

}

bool empty() const {

return size == 0;

}

void add(const T& value) {

if (size == capacity) {

resize(capacity \* 2);

}

data[size++] = new T(value);

}

T get() {

if (empty()) {

throw out\_of\_range("Vector is empty");

}

T value = \*static\_cast<T\*>(data[--size]);

delete static\_cast<T\*>(data[size]);

return value;

}

T& operator[](size\_t index) {

if (index >= size) {

throw out\_of\_range("Index out of range");

}

return \*static\_cast<T\*>(data[index]);

}

size\_t getSize() const {

return size;

}

friend ostream& operator<<(ostream& os, const Vector& vec) {

Рис. П3.18. Текст модуля Vector.h

Продолжение рис. П3.18

for (size\_t i = 0; i < vec.size; ++i) {

T value = \*reinterpret\_cast<T\*>(vec.data[i]);

os << value->getName() << endl;

}

return os;

}

friend string& operator<<(string& s, const Vector& vec) {

stringstream ss;

for (size\_t i = 0; i < vec.size; ++i) {

T value = \*reinterpret\_cast<T\*>(vec.data[i]);

ss << value->getName() << "\n";

}

s = ss.str();

return s;

}

void print\_NewOld(ofstream& result, Device::MyForm^ myForm) {

myForm->setClear();

if (size == 0) {

myForm->setText("Подключений не было");

}

for (int i = size - 1; i >= 0; --i) {

T value = \*reinterpret\_cast<T\*>(data[i]);

if (value != nullptr) {

if (size <= 20) {

myForm->setText(marshal\_as<System::String^>(value->getName()));

}

else {

result << value->getName() << endl;

}

}

}

if (size > 20) {

myForm->setText("История подключений записана в файл result.txt");

}

}

void arr(string\* networks) {

for (size\_t i = 0; i < size; ++i) {

T value = \*reinterpret\_cast<T\*>(data[i]);

if (value != nullptr) {

networks[i] = value->getName();

}

}

}

void print\_AZ(ofstream& result, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

string\* networks = new string[size];

arr(networks);

sort(networks, networks + n);

if (n == 0) {

myForm->setText("Подключений не было");

}

else {

for (int i = 0; i < n; ++i) {

if (size <= 20) {

myForm->setText(marshal\_as<System::String^>(networks[i]));

}

else {

result << networks[i] << endl;

}

}

if (size > 20) {

myForm->setText("История подключений записана в файл result.txt");

}

}

delete[] networks;

}

void print\_ZA(ofstream& result, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

Продолжение рис. П3.18

string\* networks = new string[size];

arr(networks);

sort(networks, networks + n, greater<string>());

if (n == 0) {

myForm->setText("Подключений не было");

}

else {

for (int i = 0; i < n; ++i) {

if (size <= 20) {

myForm->setText(marshal\_as<System::String^>(networks[i]));

}

else {

result << networks[i] << endl;

}

}

if (size > 20) {

myForm->setText("История подключений записана в файл result.txt");

}

}

delete[] networks;

}

void search(const string& name, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

int found = 0;

for (int i = 0; i < n; ++i) {

T value = \*reinterpret\_cast<T\*>(data[i]);

if (value != nullptr && value->getName() == name) {

myForm->setText(marshal\_as<System::String^>(value->getName()));

found++;

}

}

if (found == 0) {

myForm->setText("Устройство не найдено");

}

}

void broadcast(String^ info, Device::MyForm^ myForm) {

T value = \*reinterpret\_cast<T\*>(data[size - 1]);

value->setData(marshal\_as<string>(info));

}

};

Текст модуля Queue.h представлен на рис. П3.19.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <msclr\marshal.h>

#include <msclr\marshal\_cppstd.h>

#include <msclr\marshal\_windows.h>

#include "DataDevices.h"

#include "WiFi\_router.h"

#include "Mobile\_antenna.h"

#include "Modem.h"

#include "Switch.h"

#include "Hub.h"

#include "Flash.h"

#include "HDD.h"

#include "SSD.h"

#include "MyForm.h"

using namespace std;

using namespace msclr::interop;

Рис. П3.19. Текст модуля Queue.h

Продолжение рис. П3.19

template <class T>

class Queue {

private:

struct Node { //узел очереди

T data;

Node\* next;

};

Node\* head;

Node\* tail;

int size = 0;

public:

Queue() { //конструктор

head = NULL;

tail = NULL;

}

Queue(const Queue& other) { //копирующий конструктор

Node\* current = other.head;

while (current) {

add(new T(\*current->data));

current = current->next;

}

}

bool empty() { //проверка очереди на пустоту

return head == NULL;

}

void add(T value) { //добавление элемента в очередь

Node\* newNode = new Node;

newNode->data = value;

newNode->next = NULL;

if (empty()) {

head = tail = newNode;

}

else {

tail->next = newNode;

tail = newNode;

}

size++;

}

string del() { //взятие и удаление элемента из очереди

if (!empty()) {

string d = head->data.DataDevices::getName();

Node\* tmp = head;

head = head->next;

delete(tmp);

size--;

return d;

}

}

void nullQueue() { //обнуление очереди

Node\* tmp;

while (!empty()) {

tmp = head;

head = head->next;

delete(tmp);

}

}

bool operator >= (const Queue& other) {

return this->size >= other.size;

}

bool operator < (const Queue& other) {

return this->size < other.size;

}

int getSize() { //геттер для размера очереди

Продолжение рис. П3.19

return size;

}

friend ostream& operator<<(ostream& del, Queue& q) {

while (!q.empty())

del << q.del() << endl;

return del;

}

friend string& operator<<(std::string& s, Queue& q) {

stringstream ss;

while (!q.empty()) {

ss << q.del() << "\n";

}

s = ss.str();

return s;

}

void arr(string\* a, int& i) {

while (!empty()) {

a[i] = del();

i++;

}

}

void print\_NewOld(ofstream& result, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

string s;

string\* networks = new string[size];

int i = 0;

arr(networks, i);

i--;

if (n == 0) myForm->setText("Подключений не было");

for (i; i >= 0; i--) {

if (n <= 20) {

myForm->setText(marshal\_as<System::String^>(networks[i]));

}

else result << networks[i] << endl;

}

if (n > 20) myForm->setText("История подключений записана в файл result.txt");

}

void print\_AZ(ofstream& result, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

string s;

string\* networks = new string[size];

int i = 0;

arr(networks, i);

for (int i = 0; i < n; i++)

for (int j = i + 1; j < n; j++) {

if (networks[i] > networks[j]) {

swap(networks[i], networks[j]);

}

}

if (n == 0) myForm->setText("Подключений не было");

for (int i = 0; i < n; i++) {

if (n <= 20) myForm->setText(marshal\_as<System::String^>(networks[i]));

else result << networks[i] << endl;

}

if (n > 20) myForm->setText("История подключений записана в файл result.txt");

delete[] networks;

}

void print\_ZA(ofstream& result, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

string s;

string\* networks = new string[size];

int i = 0;

arr(networks, i);

for (int i = 0; i < n; i++)

for (int j = i + 1; j < n; j++) {

if (networks[i] < networks[j]) {

Продолжение рис. П3.19

swap(networks[i], networks[j]);

}

}

if (n == 0) myForm->setText("Подключений не было");

for (int i = 0; i < n; i++) {

if (n <= 20) myForm->setText(marshal\_as<System::String^>(networks[i]));

else

result << networks[i] << endl;

}

if (n > 20) myForm->setText("История подключений записана в файл result.txt");

delete[] networks;

}

void search(string name, Device::MyForm^ myForm) {

myForm->setClear();

int n = size;

string\* networks = new string[size];

int i = 0;

arr(networks, i);

int found = 0;

for (int i = 0; i < n; i++) {

if (networks[i] == name) {

myForm->setText(marshal\_as<System::String^>(networks[i]));

found++;

}

}

if (found == 0) myForm->setText("Устройство не найдено");

delete[] networks;

}

~Queue() { //деструктор

}

};

Текст модуля Exception.h представлен на рис. П3.20.

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

class MyException : public exception

{

private:

string message\_error;

public:

MyException(string error) {

this->message\_error = error;

}

const char\* what() const noexcept {

return message\_error.c\_str();

}

};

Рис. П3.20. Текст модуля Exception.h

# Приложение 4