МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Платформенно-независимое программирование»

**КУРСОВАЯ РАБОТА**

Тема Разработка кроссплатформенного приложения для работы со списками на Qt и С++

Выполнил студент /Цуканова И.В./

подпись инициалы, фамилия

Курс третий Группа ПИбд-32

Направление /специальность 09.03.04 «Программная инженерия»

Руководитель

должность, ученая степень, ученое звание фамилия, имя, отчество

Дата сдачи:

« » 2024 г.

Дата защиты:

« » 2024 г. Оценка:

Ульяновск

2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Платформенно-независимое программирование»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студенту ПИбд-32 Цуканова И.В.

группа фамилия, инициалы

Тема работы Разработка кроссплатформенного приложения для работы со списками на Qt и С++

Срок сдачи законченной работы « » 2024 г.

**Исходные данные к работе:** описание задания по теме, утвержденной распоряжением деканата ФИСТ

**Рекомендуемые источники:** курс лекций по дисциплине «Платформенно-независимое программирование», методические указания к лабораторным работам по дисциплине «Платформенно-независимое программирование», интернет-источники.

**Содержание пояснительной записки** (перечень подлежащих разработке вопросов) Введение.

Первая глава. Проектирование программной системы.

Вторая глава. Разработка программной системы.

Третья глава. Руководство пользователя.

**Перечень графического материала** (с точным указанием обязательных чертежей)

Схемы: схема взаимодействия между слоями приложения, блок-схемы.

Скриншоты разработанного программного продукта.

Руководитель / /

должность подпись инициалы, фамилия

« » 2024г

Студент / Цуканова И.В./

подпись инициалы, фамилия

« » 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ**

**руководителя на курсовую работу**

Студента Цукановой Ирины Валерьевны

фамилия, имя и отчество

Факультет ИСТ группа ПИбд-32 курс третий Дисциплина «Платформенно-независимое программирование»

Тема работы Разработка кроссплатформенного приложения для работы со списками на Qt и С++

Отмечаются следующие моменты: актуальность темы исследования; соответствие содержания и структуры курсовой работы ее теме; степень разработанности проблемы, наиболее интересно исследованные вопросы. Оценивается степень самостоятельности и инициативы студента; умение пользоваться различными источниками информации; уровень его теоретической подготовки; умение анализировать научные материалы, делать практические выводы; знание основных концепций, научной и специальной литературы по избранной теме. Содержится оценка проекта (работы) руководителем.

Руководитель / /

должность, ученая степень, ученое звание подпись инициалы, фамилия

« » 2024г

Оглавление

[Введение 2](#_Toc156213115)

[Глава 1. Проектирование программной системы 5](#_Toc156213116)

[Глава 1.1 Математическое обеспечение. 5](#_Toc156213117)

[Глава 1.2 Интерфейс пользователя. 12](#_Toc156213118)

[Глава 1.3 Архитектура приложения. 13](#_Toc156213119)

[Глава 2. Разработка программной системы 14](#_Toc156213120)

[Глава 3. Руководство пользователя 18](#_Toc156213121)

[Вывод 35](#_Toc156213122)

[Исходный код 36](#_Toc156213123)

# Введение

Разработка кроссплатформенного приложения для работы со списками на Qt и C++ предоставляет ряд преимуществ и особенностей. Вот ключевые аспекты:

Qt Framework:

* + 1. Кроссплатформенность - Qt предоставляет кроссплатформенные инструменты, позволяющие разрабатывать приложения, которые могут работать на различных операционных системах, таких как Windows, Linux, и macOS.
    2. Графический интерфейс (GUI) - Qt обеспечивает гибкую систему для создания GUI. Вы можете использовать Qt Designer для визуального создания интерфейса приложения и управления списками.
    3. Сигналы и слоты - Механизм сигналов и слотов Qt облегчает взаимодействие между различными компонентами приложения, такими как списки и другие элементы GUI, а также эффективную связь между логикой приложения и графическим интерфейсом.
    4. Qt Containers - Qt предоставляет собой богатый набор контейнеров, включая QList, QVector, и другие, что облегчает управление данными в списках.

C++ :

* + 1. Эффективность - Использование C++ обеспечивает высокую производительность приложения, что особенно важно для обработки больших объемов данных в списках.
    2. Стандартные библиотеки - C++ STL (Standard Template Library) предоставляет стандартные контейнеры, такие как std::vector и std::list, которые могут быть использованы вместе с Qt для управления данными.

Совместное использование

* + 1. Интеграция Qt и C++ - Qt может легко интегрироваться с кодом на C++, позволяя использовать преимущества обеих технологий.
    2. Многоплатформенная сборка - Используйте средства сборки, такие как CMake, для создания кроссплатформенных сборок, которые поддерживают разные операционные системы.
    3. Работа с данными - Использование Qt-контейнеров вместе с возможностями C++ для обработки данных в списках, обеспечивая эффективное и кроссплатформенное управление данными.

Особенности ручной реализации односвязного списка списка. Во-первых, эта реализация связанного списка с использованием структур данных. Во-вторых, Каждый элемент списка хранит данные и указатель на следующий элемент. Также позволяет динамически добавлять и удалять элементы.

Реализация списка на основе массива – использование массива с возможностью изменения размера. Позволяет динамически изменять размер списка при добавлении/удалении элементов. Динамический массив, обеспечивает быстрый доступ к элементам.

Реализация списка на основе дэка(двусторонняя очередь). Дек является двусторонней структурой данных, что означает, что вы можете добавлять и удалять элементы как с начала, так и с конца. Это обеспечивает эффективность операций вставки и удаления на обоих концах.

Ручная реализация двусвязного списка предполагает создание собственного класса для узла списка и класса для самого списка.

Выбор определенной реализации зависит от требований по производительности, доступу и изменению данных. Стандартные контейнеры предоставляют удобство и широкий функционал, но ручная реализация может быть полезна для специфических сценариев.

Особенности списка:

* Упорядоченность: Список представляет собой упорядоченную коллекцию элементов. Это означает, что порядок элементов в списке имеет значение, и они располагаются в определенной последовательности.
* Динамичность: Список может изменяться по мере выполнения операций добавления, удаления и изменения элементов. Это позволяет эффективно управлять данными в процессе выполнения программы.
* Гибкость: Списки поддерживают различные методы, что делает их мощным инструментом для обработки данных.
* Размер: Список может содержать любое количество элементов, в том числе и ноль. Это позволяет представлять как небольшие, так и большие коллекции данных.
* Использование в петлях: Списки часто используются в циклах для обработки каждого элемента последовательности, что делает их важным элементом в программировании.

Существует несколько методов оценки сложности алгоритмов, которые позволяют оценить, насколько эффективно алгоритм решает задачу.

* + 1. Асимптотическая нотация:

O-нотация (Big O): описывает верхнюю границу роста алгоритма. Включает тету.

Ω-нотация (Big Omega): показывает нижнюю границу роста алгоритма. Такая сложность, эффективней которой алгоритм никогда не работает. Не включает тету.

Θ-нотация (Theta): объединяет O-нотацию и Ω-нотацию, указывая точную границу роста алгоритма.

* + 1. Анализ времени выполнения:

Измерение количества базовых операций, которые выполняет алгоритм, в зависимости от размера входных данных.

Оценка лучшего, среднего и худшего случаев выполнения алгоритма.

* + 1. Пространственная сложность:

Оценка объема памяти, необходимого для выполнения алгоритма в зависимости от размера входных данных.

* + 1. Сложность по времени и пространству в совокупности:

Комбинированная оценка, учитывающая как время выполнения, так и использование памяти алгоритмом.

# Глава 1. Проектирование программной системы

Глава 1.1 Математическое обеспечение.

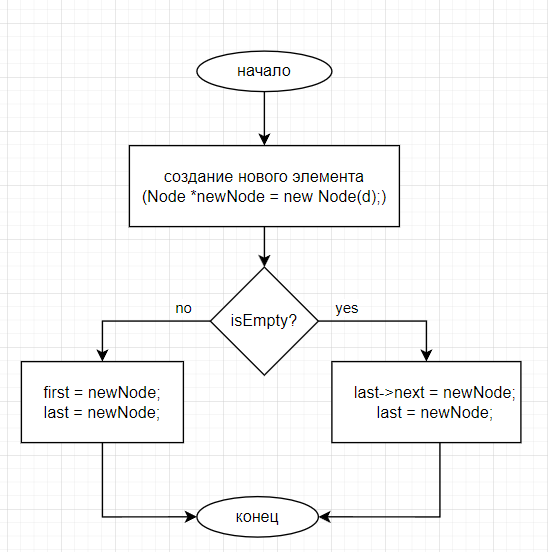
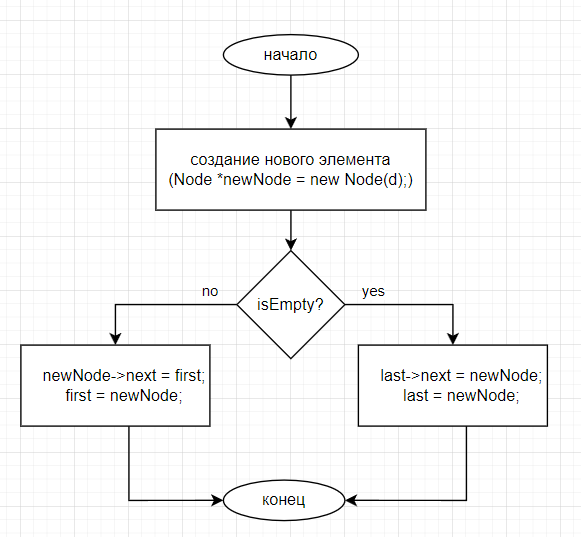
Блок схемы операций для LinkedList

Рисунок 1.1 pushBack



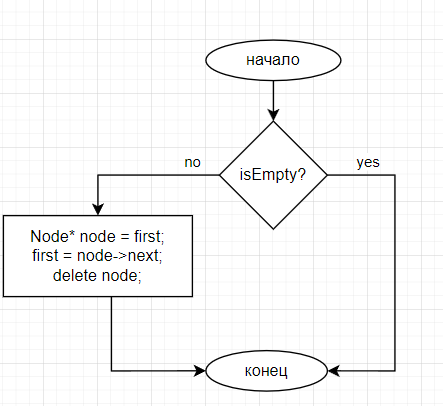
Рисунок 1.2 pushFront

Рисунок 1.3 removeLast

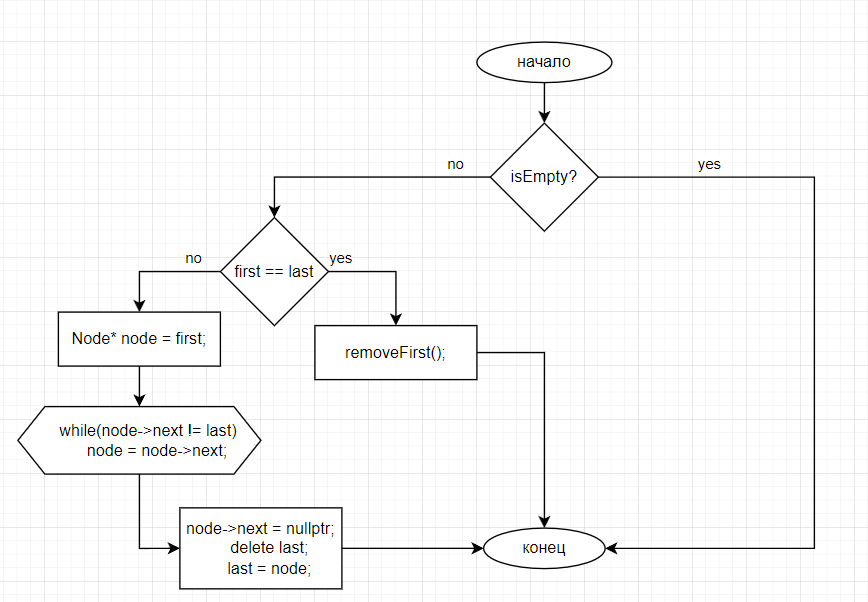
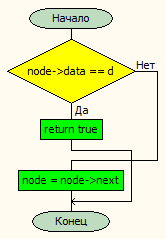
Рисунок 1.4 removeFirst

Рисунок 1.5 contains

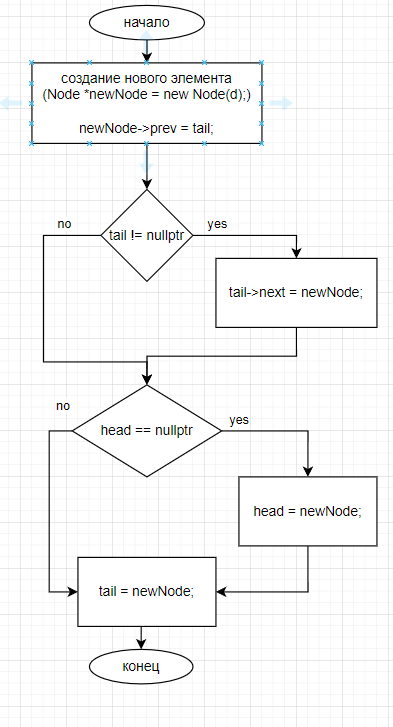
Блок схемы операций для DoubleLinkedList

Рисунок 1.6 pushBack

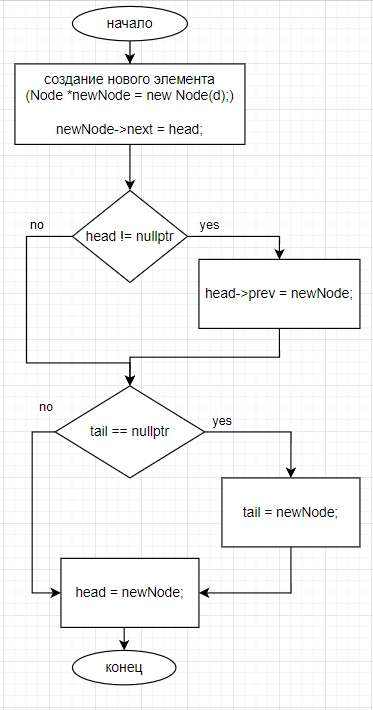


Рисунок 1.7 pushFront

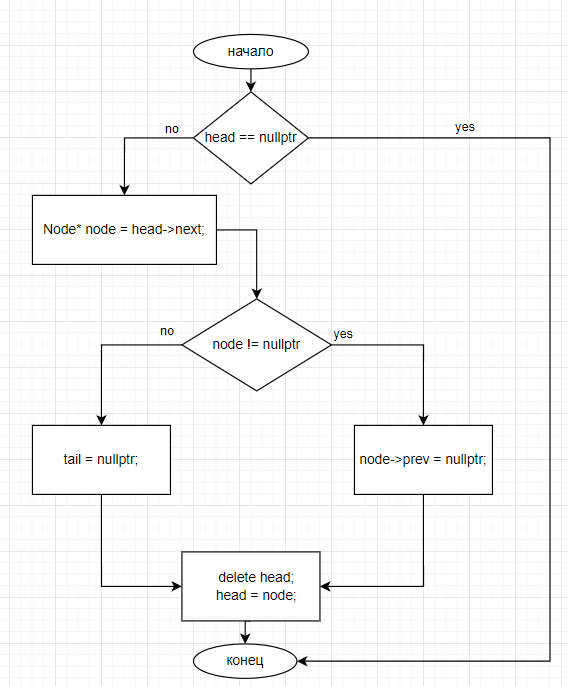


Рисунок 1.8 deleteFront

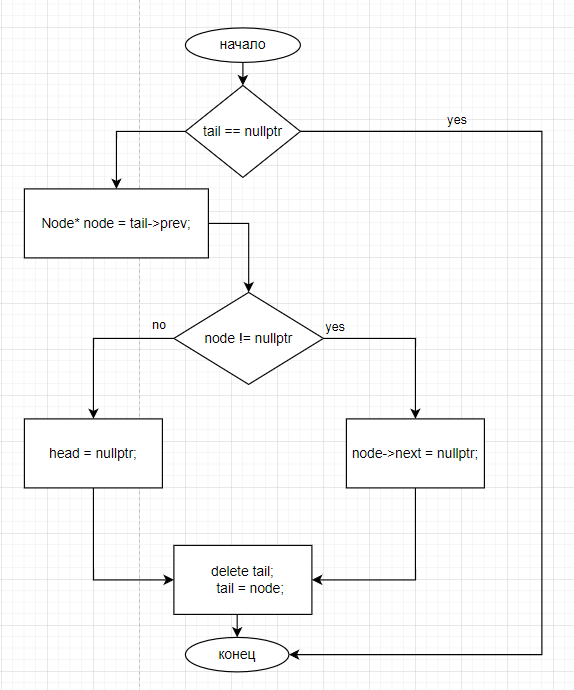
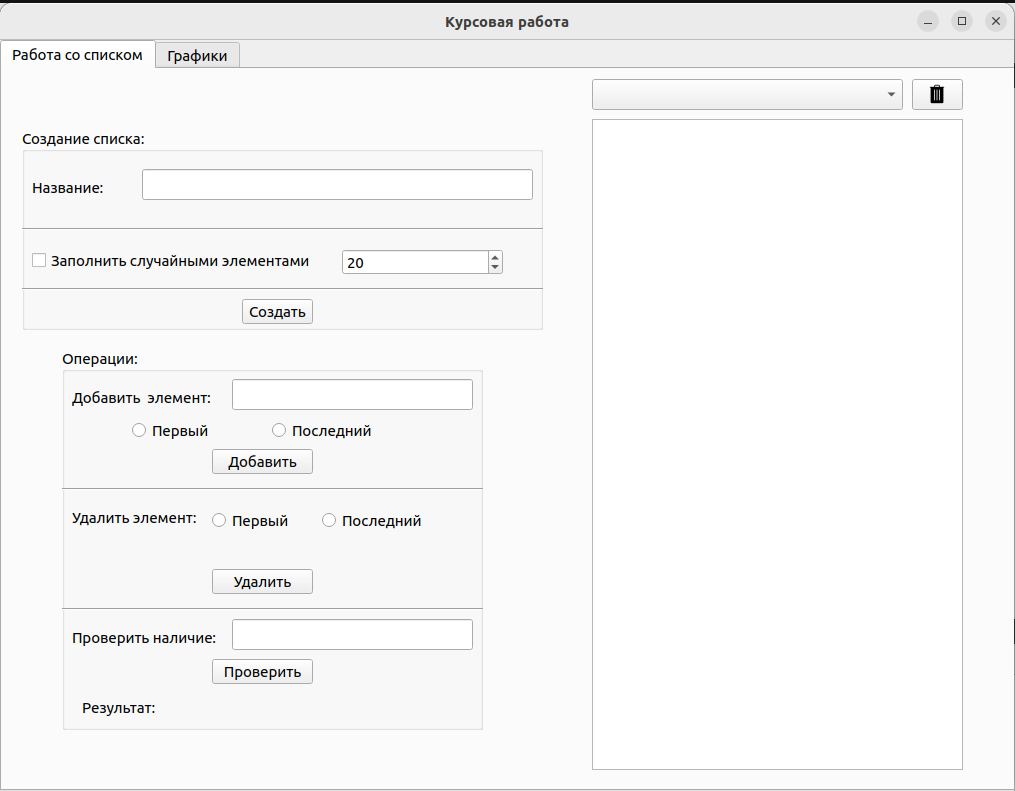


Рисунок 1.8 deleteBack

Глава 1.2 Интерфейс пользователя.

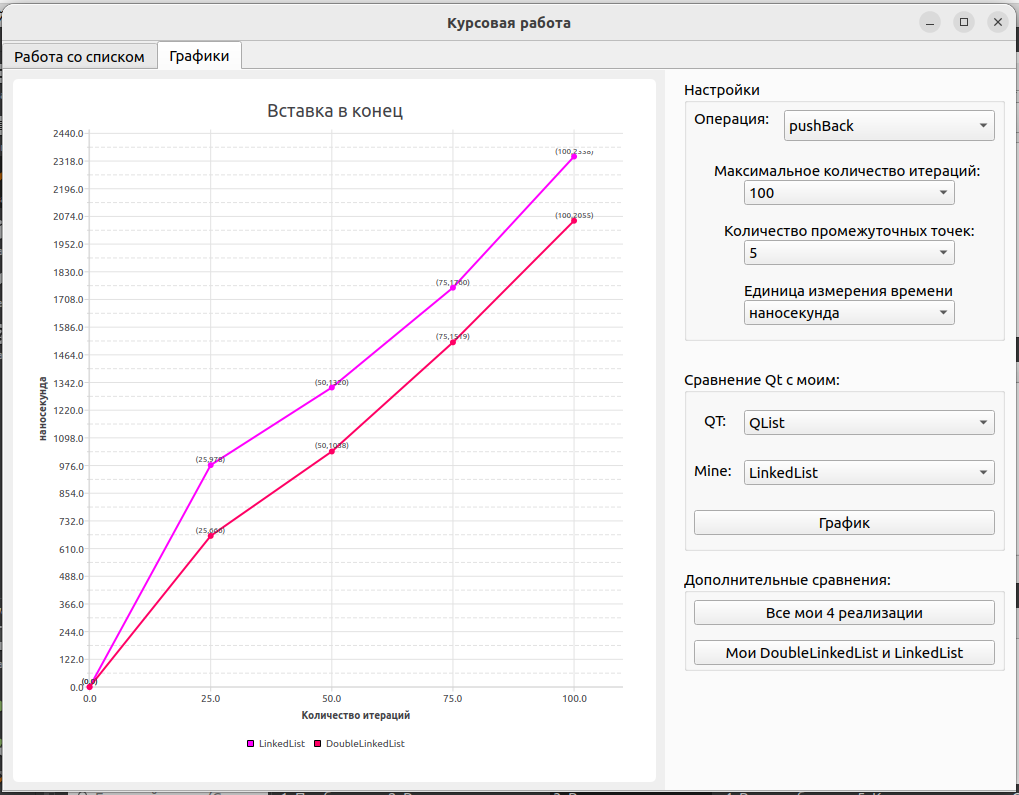
Разработанное приложение имеет одну экранную форму с двумя вкладками – «Работа со списком», «Графики».

Первая вкладка предоставляет пользователю проведения различных операций над списками – создание (с возможностью заполнения списка случайными элементами), перемещение между созданными списками, удаление списка, добавление элемента в конец списка, добавление элемента в начало списка, удаление элемента из конца списка, удаление элемента из начала списка, проверка элемента на наличие в списке.

Рисунок 1.2.1 Вкладка «Работа со списком»

Вторая вкладка предоставляет пользователю формирование различных графиков (по оси абсцисс – количество итераций, по оси ординат – время выполнения соответствующего количества итераций) с выбранными настройками. Настроить можно вид операции, максимальное количество итераций выбранной операции, количество промежуточных точек, отображаемых на графике, единицу измерения времени. Графики сравнений могут быть следующими:

* Сравнение одной из четырех моих реализация с QLinkedList или QList.
* Сравнение всех четырех моих реализаций.
* Сравнение моих реализаций DoubleLinkedList и LinkedList.

Рисунок 1.2.2 Вкладка «Графики»

Глава 1.3 Архитектура приложения.

Главный файл main.cpp является входной точкой, он открывает главную экранную форму приложения.

В приложении один интерфейс - IList, прописанный в заголовочном файле ilist.h, от которого наследуются классы реализаций списков. Всего реализаций четыре:

* + 1. LinkedList – класс реализации односвязного списка. Заголовочный файл – linkedlist.h, файл исходного кода – linkedlist.cpp.
    2. DoubleLinkedList – класс реализации двусвязного списка. Заголовочный файл – doublelinkedlist.h, файл исходного кода – doublelinkedlist.cpp.
    3. ArrayList – класс реализации списка на основе массива. Заголовочный файл – arraylist.h, файл исходного кода – arraylist.cpp.
    4. DequeList – класс реализации списка на основе дэка(двусторонняя очередь). Заголовочный файл – dequelist.h, файл исходного кода – dequelist.cpp.

Файл mainwindow.cpp содержит логику, связанную с интерфейсом программы, выводы графиков, осуществления замеров времени, проведения итераций, работы с списком.

# Глава 2. Разработка программной системы

Разработан интерфейс IList, от которого наследуются реализации списка, с следующими методами:

* pushBack – метод добавления элемента в конец списка.
* pushFront – метод добавление элемента в начало списка.
* removeFirst – метод удаление элемента из начала списка.
* removeLast – метод удаления элемента из конца списка.
* contains – метод проверки нахождения элемента в списке.
* print – метод вывода списка.
* clear – метод удаления всех элементов списка (очистка списка).

Реализации списка представлены в четырех классах, наследуемых от интерфейса IList. В каждом классе своя реализация всех виртуальных методов интерфейса IList.

Классы реализации:

1. ArrayList.   
   Методы:
   1. Все методы интерфейса IList.
   2. add – метод добавление элемента по индексу.
   3. remove – метод удаление элемента по индексу.
   4. isEmpty – метод проверки пустой ли список.
   5. isNeedToResize – метод проверка нужно ли расширять массив.
   6. resize – метод расширения массива.

Поля:

* 1. data – массив.
  2. list\_size – текущий размер списка.
  3. arr\_size – текущий размер массива.

1. DequeList.

Методы:

* 1. Все методы интерфейса IList.

Поля:

* 1. deque– дэк.

1. DoubleLinkedList.

Методы:

* 1. Все методы интерфейса IList.
  2. isEmpty – метод проверки пустой ли список.

Поля:

* 1. head – первый элемент.
  2. taile – последний элемент.

1. LinkedList.

Методы:

* 1. Все методы интерфейса IList.
  2. toQList – метод, возвращающий текущий список как QList содержащий в себе тип QString.
  3. isEmpty – метод проверки пустой ли список.

Поля:

* 1. first – первый элемент.
  2. last – последний элемент.

MainWindow - класс логики интерфейса и диаграмм.

Методы MainWindow:

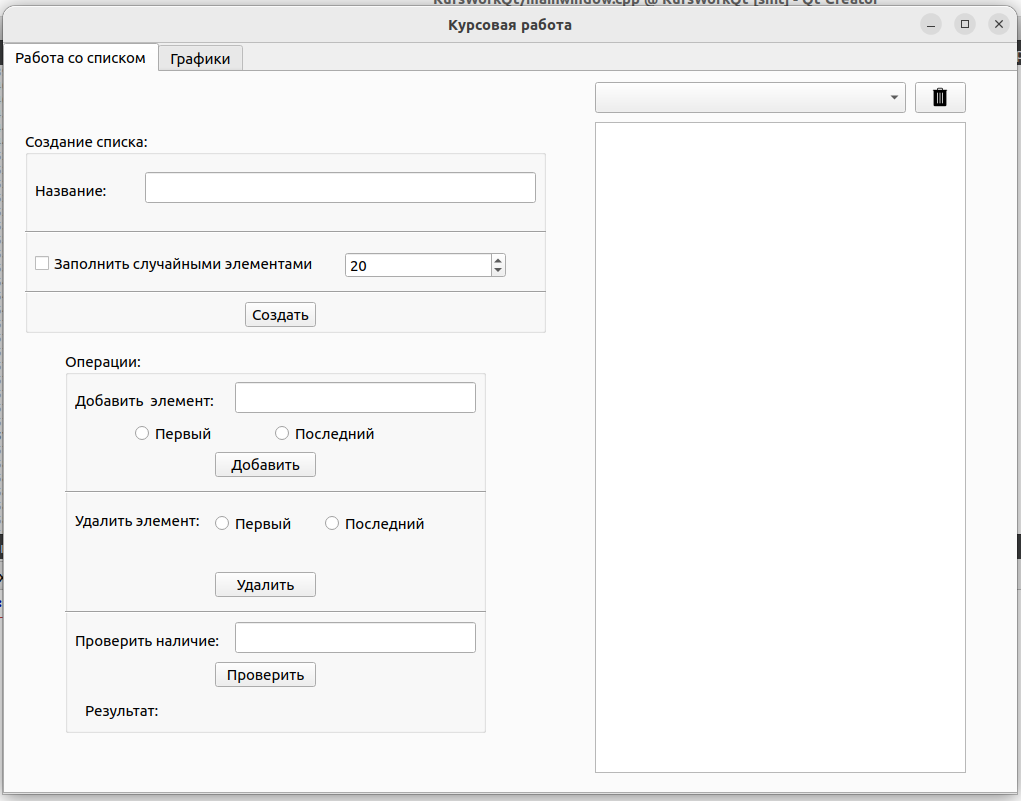
* сonfigure – метод, задающий общие настройки для диаграммы.
* configureSeries – метод, задающий общие настройки для серий.
* test – метод, в котором происходит подсчет времени выполнения итераций для реализаций списка.
* testQt – метод, в котором происходит подсчет времени выполнения итераций для Qt списков.
* setChartProps – метод, задающий частные настройки для диаграммы.
* setSeriesProps – метод, задающий частные настройки для серии.
* setAxisProps – метод, задающий частные настройки для осей диаграммы.
* on\_pushButton\_clicked – метод (слот) обработки нажатия на кнопку, отрисовка графика сравнения всех четырех реализаций.
* on\_pushButton\_2\_clicked – метод (слот) обработки нажатия на кнопку, отрисовка графика сравнения реализаций DoubleLinkedList и LinkedList.
* on\_pushButton\_qt\_mine\_clicked – метод (слот) обработки нажатия на кнопку, отрисовка графика сравнения одной из четырёх реализаций с QList или QLinkedList.
* on\_comboBox\_maxIter\_currentTextChanged – метод(слот) обработки изменения текущего значения максимального количества итераций.
* on\_comboBox\_colIter\_currentTextChanged – метод (слот) обработки изменения текущего значения количества итераций.
* on\_comboBox\_ci\_currentTextChanged – метод (слот) обработки изменения текущего значения единицы измерения.
* on\_pushButton\_create\_clicked – метод (слот) обработки нажатия на кнопку, создание нового списка.
* on\_comboBox\_select\_currentTextChanged – метод (слот) обработки изменения текущего списка.
* on\_pushButton\_add\_clicked – метод (слот) обработки нажатия на кнопку, добавление нового элемента в список.
* on\_pushButton\_del\_clicked – метод (слот) обработки нажатия на кнопку, удаление элемента из списка.
* on\_pushButton\_contains\_clicked – метод (слот) обработки нажатия на кнопку, проверка нахождения элемента в списке
* on\_pushButton\_del\_list\_clicked – метод (слот) обработки нажатия на кнопку, удаление списка.

Поля MainWindow:

* ui – графияеское окно.
* chart – диаграмма.
* fontChartTitle – шрифт для заголовка.
* fontAxis – шрифт для осей.
* fontPointLabels – шрифт для подписей.
* maxIter – максимальное количество итераций.
* colIter – количество промежуточных точек.
* del – значение для определения единицы измерения времени.
* methods – словарь методов.
* qt – словарь Qt списков.
* mine – словарь реализаций.
* ci – словарь единиц измерения времени.
* lists – словарь с листами.

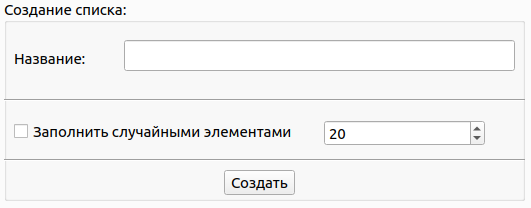
# Глава 3. Руководство пользователя

При запуске программы открывается следующее окно (Рисунок 3.1):

Рисунок 3.1

В нем представлен функционал работы с списком – создание списка, переход между созданными списками, удаление списка, добавление элемента в начало или конец списка, удаление элемента из начала списка или из конца, проверка наличия элемента в списке.

Функционал создания списка находятся в блоке с названием «Создание списка:» (Рисунок 3.2).

Рисунок 3.2

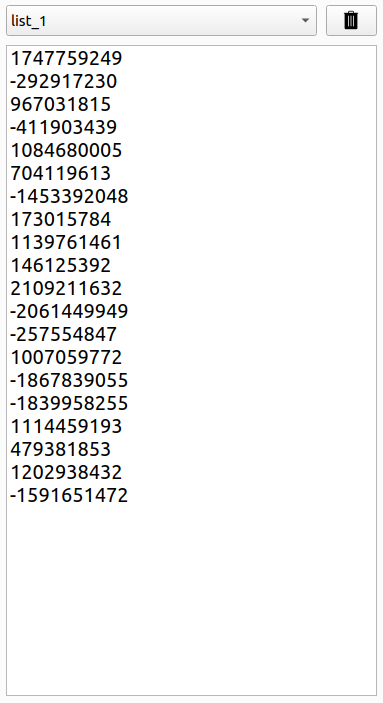
При создании списка необходимо указать его название – поле для ввода названия находится рядом с заголовком «Название:» (Рисунок 3.3).

Рисунок 3.3

Если название будет пустым, то список не создастся, также название списка должно быть уникальным. По желанию можно заполнить список случайными элементами – целыми числами, могут быть и отрицательными, и положительными – для этого нужно поставить галочку слева от заголовка «Заполнить случайными элементами» (Рисунок 3.4). По умолчанию количество добавленных элементов 20, это число можно изменить, выбрав нужное количество в поле справа от заголовка «Заполнить случайными элементами» (Рисунок 3.4).

Рисунок 3.4

Далее для создания списка необходимо нажать на кнопку «Создать», расположенную в самом низу блока «Создание списка:». В случае, если все действия были выполнены корректно в правой части вкладки «Работа со списком» будет отображен созданный лист со всеми его элементами (Рисунок 3.5).

Рисунок 3.5

Над блоком с элементами листа расположен переключатель между созданными листами и кнопка удаления (Рисунок 3.6).

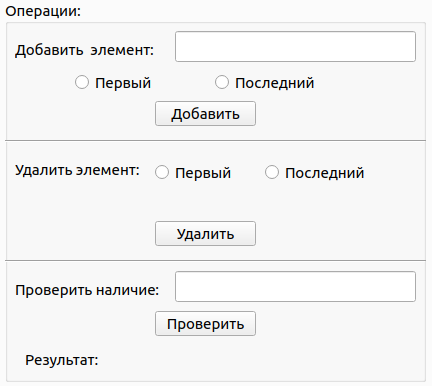
Рисунок 3.6

При нажатии на переключатель, будут отображены названия всех существующих списков (Рисунок 3.7). Для переключения на другой список необходимо нажать на его название.

Рисунок 3.7

Для удаления списка необходимо выбрать нужный список в переключателе и нажать на кнопку удаления справа от переключателя.

Функционал добавление элемента в начало или конец списка, удаление элемента из начала списка или из конца и проверка наличия элемента в списке представлен в блоке «Операции:» (Рисунок 3.8).

Рисунок 3.8

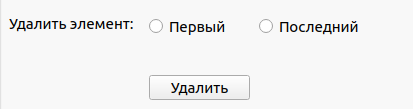
Для добавления элемента в список необходимо ввести добавляемый элемент в поле слева от заголовка «Добавить элемент:» (Рисунок 3.9). Добавляемый элемент должен быть целочисленным, иначе добавления не произойдет.

Рисунок 3.9

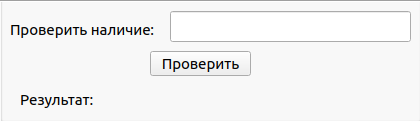
Далее необходимо выбрать, будет ли элемент добавлен в начало списка или в его конец, для этого надо выбрать один из вариантов – «Первый» или «Последний» – под полем для ввода элемента (Рисунок 3.10). После нажать на кнопку «Добавить» и в случае, если все действия были выполнены корректно, в список в нужную позицию добавится введенный элемент.

Рисунок 3.10

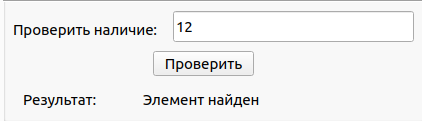
Для удаления элемента из списка необходимо выбрать с какой позиции будет удален элемент – «Первый» для удаления первого элемента или «Последний» для удаления последнего элемента. И затем нажать на кнопку «Удалить» (Рисунок 3.11). В случае, если все действия были выполнены корректно, из списка будет удален один элемент.

Рисунок 3.11

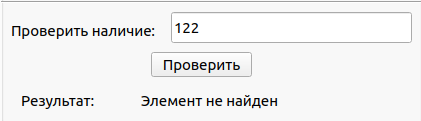
Для проверки наличия элемента в списке сначала необходимо ввести проверяемый элемент в поле рядом с заголовком «Проверить наличие:» (Рисунок 3.12). И затем нажать на кнопку «Проверить». Результат проверки будет высвечен справа от заголовка «Результат:» (Рисунок 3.12).

Рисунок 3.12

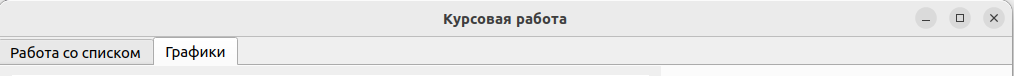
В случае, если элемент был найден в списке, появится надпись «Элемент найден» (Рисунок 3.14).

Рисунок 3.14

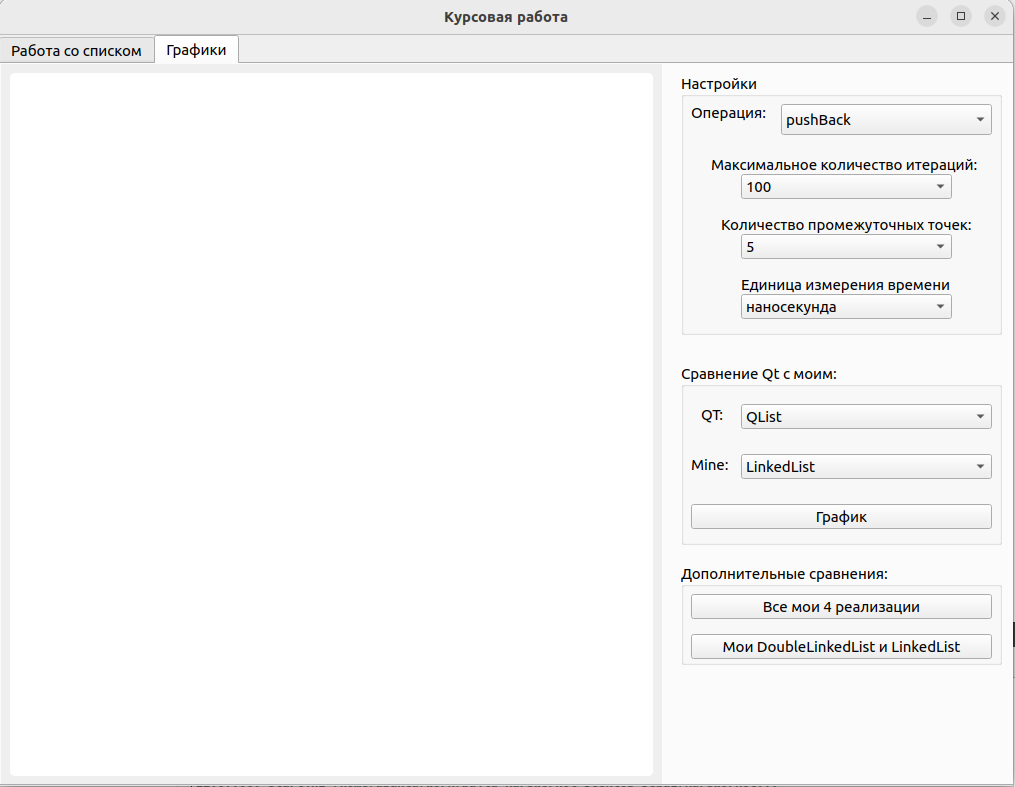
В случае, если элемент не был найден в списке, появится надпись «Элемент не найден» (Рисунок 3.15).

Рисунок 3.15

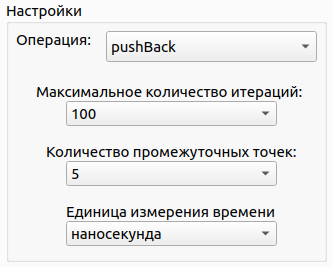
Для перехода на вкладку с функционалом графиков необходимо выбрать вариант «Графики» в левом верхнем углу экранной формы приложения (Рисунок 3.16).

Рисунок 3.16

Далее будет отображено следующее окно (Рисунок 3.17).

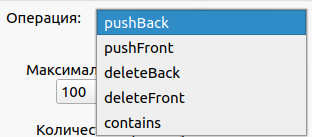
Рисунок 3.17

В блоке «Настройки» (Рисунок 3.18) представлены такие настройки, влияющие на отображаемый график.

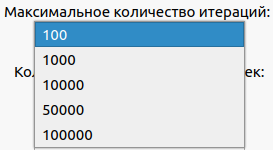
Рисунок 3.18

Всего доступны четыре настройки:

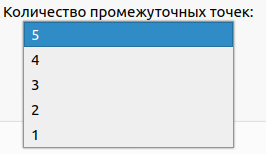
1. Выбор операции (Рисунок 3.19) – добавление в конец списка(pushBack), добавление в начало списка(pushFront), удаление из конца списка(deleteBack), удаление из начала списка(deleteFront), проверка на наличие элемента в списке(contains).

Рисунок 3.19

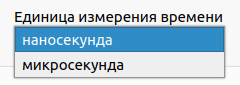
1. Выбор максимального количества итераций (Рисунок 3.20).

Рисунок 3.20

1. Выбор количества промежуточных точек (Рисунок 3.21).

Рисунок 3.21

1. Выбор единицы измерения (Рисунок 3.22).

Рисунок 3.22

Для получения графика сравнения одной из четырех реализаций списка – LinkedList, DoubleLinkedList, ArrayList, DequeList (Рисунок 3.23) – с представленными списками в Qt – QList или QLinkedList (Рисунок 3.24) – в блоке «Сравнение Qt с моим» (Рисунок 3.25) необходимо выбрать, что с чем будет сравниваться, после чего нажать на кнопку «График». Если все действия были выполнены правильно, то в левой части вкладки будет вырисован график (Рисунок 3.26).

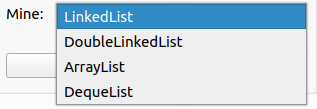
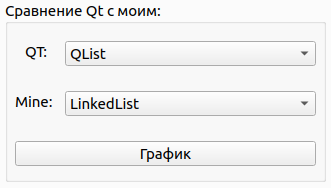
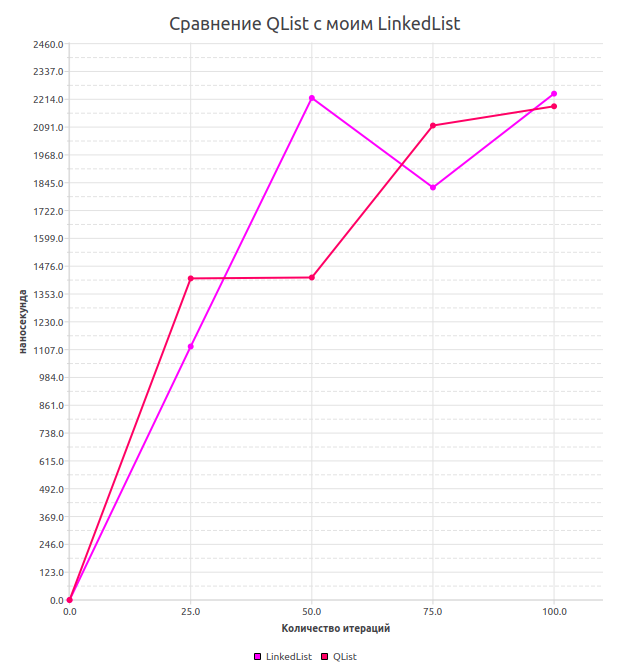
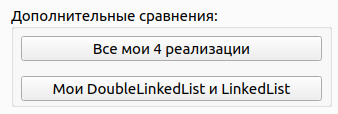
Рисунок 3.23

Рисунок 3.24

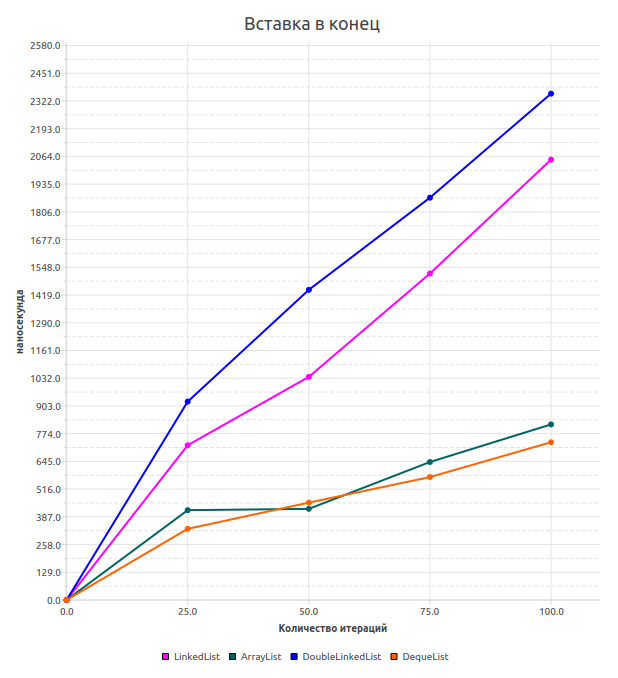
Рисунок 3.25

Рисунок 3.26

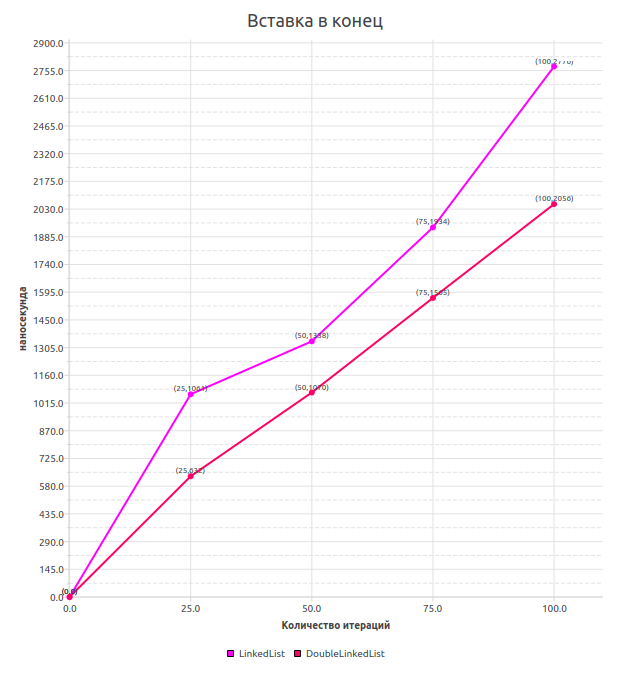
В блоке «Дополнительные сравнения»(Рисунок 3.27) представлена возможность создания дополнительных двух вариантов графиков.

Рисунок 3.27

Вариант «Все мои 4 реализации» представляет график сравнения всех четырех реализаций списка (Рисунок 3.28).

Рисунок 3.28

Вариант «Мои DoubleLinkedList и LinkedList» представляет график сравнения реализаций на связных списках – первый двусвязный, второй односвязный (Рисунок 3.29).

Рисунок 3.29

Глава 3.2 Оценки сложности

Оценки сложности методов для реализации списка – LinkedList:

1. pushBack – O(1)
2. pushFront – O(1)
3. removeFirst – O(1)
4. removeLast – O(n)
5. contains – O(n)

Оценки сложности методов для реализации списка – DoubleLinkedList:

1. pushBack – O(1)
2. pushFront – O(1)
3. removeFirst – O(1)
4. removeLast – O(1)
5. contains – O(n)

Оценки сложности методов для реализации списка – ArrayList:

1. pushBack – O(n)
2. pushFront – O(2n)
3. removeFirst – O(n)
4. removeLast – O(1)
5. contains – O(n)

Оценки сложности методов для реализации списка – LinkedList:

1. pushBack – O(1)
2. pushFront – O(1)
3. removeFirst – O(1)
4. removeLast – O(1)
5. contains – O(n)

Глава 3.3 Результаты (графики)

График результатов сравнения всех четырех реализаций по операции вставки в конец с максимальным числом итераций 10000 и пятью промежуточными точками (Рисунок 3.3.1):

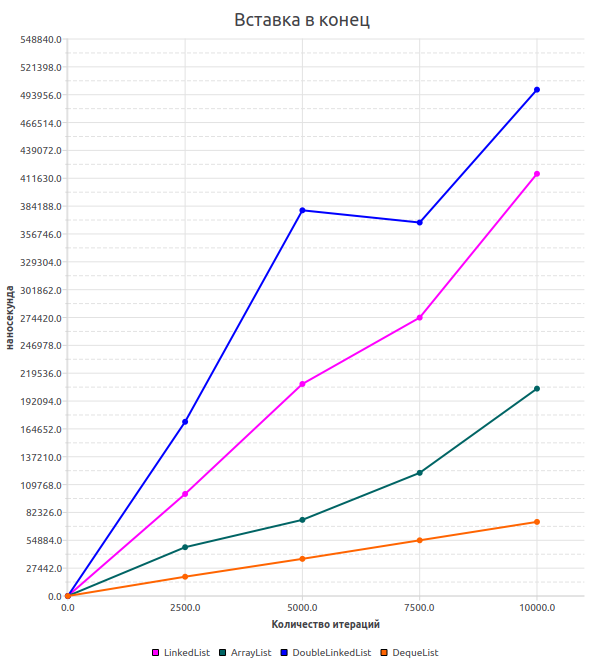
Рисунок 3.3.1

График результатов сравнения всех четырех реализаций по операции вставки в начало с максимальным числом итераций 100 и пятью промежуточными точками (Рисунок 3.3.2):

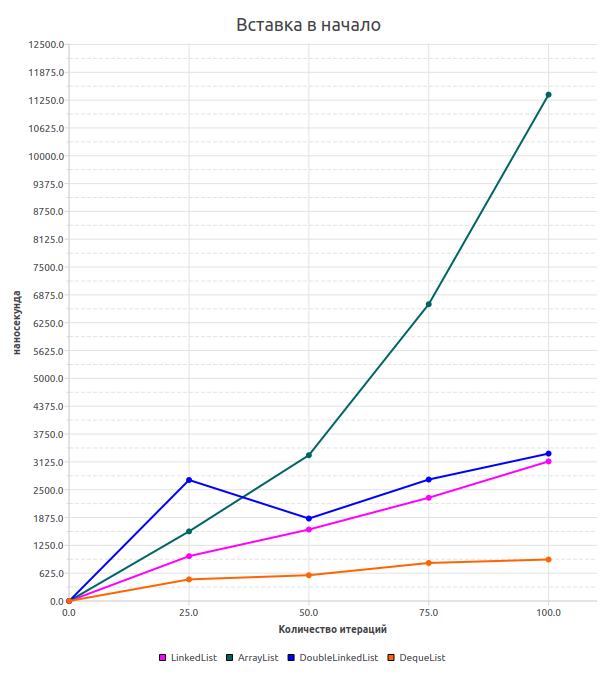
Рисунок 3.3.2

График результатов сравнения всех четырех реализаций по операции удаления с конца с максимальным числом итераций 100 и пятью промежуточными точками (Рисунок 3.3.3):

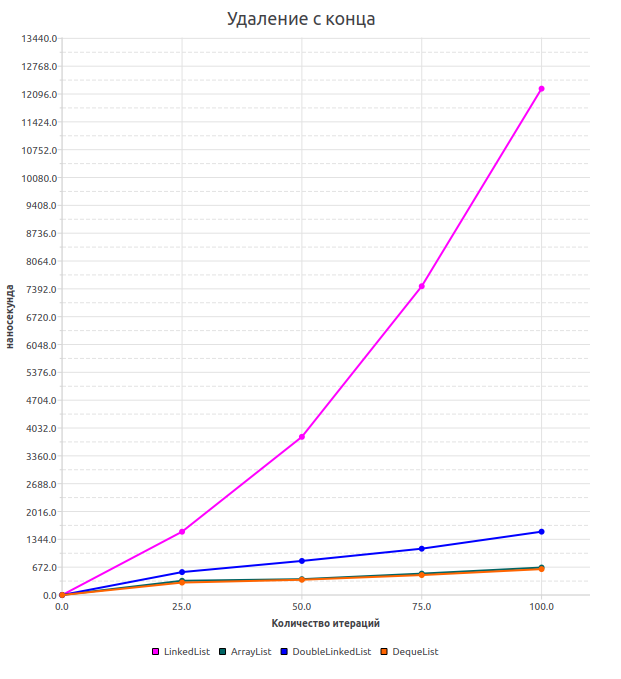
Рисунок 3.3.3

График результатов сравнения всех четырех реализаций по операции удаления с начала с максимальным числом итераций 100 и пятью промежуточными точками (Рисунок 3.3.4):

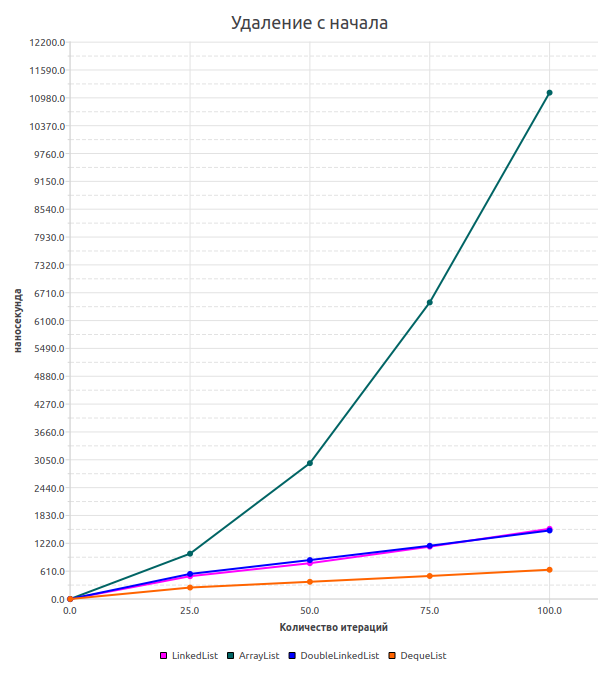
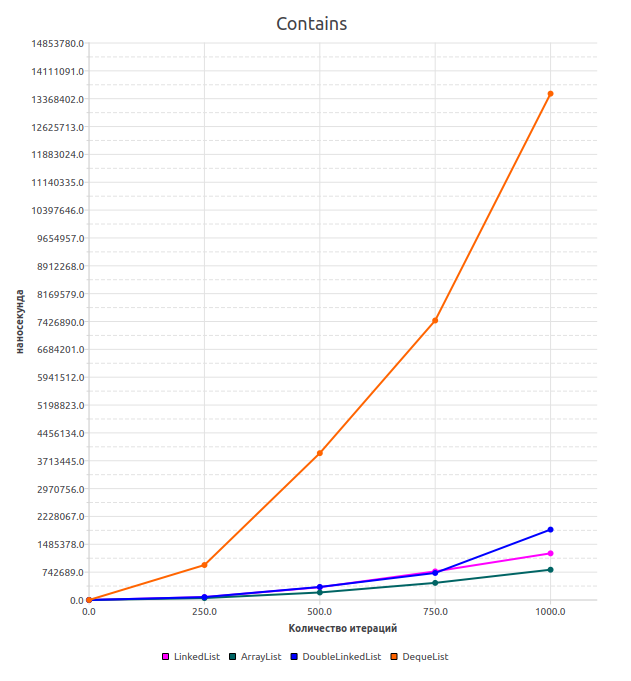
Рисунок 3.3.4

График результатов сравнения всех четырех реализаций по операции проверки элемента в списке с максимальным числом итераций 1000 и пятью промежуточными точками (Рисунок 3.3.5):

Рисунок 3.3.5

# Вывод

В ходе выполнения курсовой работы было успешно разработано кроссплатформенное приложение на базе фреймворка Qt и языка программирования C++. Проект представляет собой функциональное приложение, предназначенное для работы со списками данных, а также построения графиков сравнения ручных реализаций списка на основе массива, дека(двусторонней очереди), ручной реализации односвязного списка и двусвязного списка с различными настройками.

Одной из целей работы было создание удобного и интуитивно понятного интерфейса для пользователя. Используя возможности Qt, удалось реализовать графический интерфейс, который обеспечивает простоту в использовании и доступность функций работы со списками, а также визуализации графиков с различными настройками. Основной функционал приложения включает в себя операции добавления, удаления, поиска элементов в списках, построение графиков сравнения.

Одним из значимых моментов является кроссплатформенность приложения, что позволяет пользователям использовать его на различных операционных системах без необходимости изменений в исходном коде. Это достигнуто благодаря использованию Qt, который обеспечивает высокую степень переносимости.

В процессе разработки было уделено внимание тестированию приложения для обеспечения его стабильной работы и предотвращения возможных ошибок. Этот подход способствует повышению качества программного продукта.

Таким образом, разработанное кроссплатформенное приложение на Qt и C++ представляет собой эффективный инструмент для работы со списками данных, обеспечивая пользователю удобство использования, производительность и переносимость на различные платформы.

# Исходный код

QT += core gui charts printsupport opengl

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

CONFIG += c++11

QT\_QPA\_PLATFORM= wayland

# You can make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

arraylist.cpp \

deque.cpp \

dequelist.cpp \

doublelinkedlist.cpp \

form.cpp \

linkedlist.cpp \

main.cpp \

mainwindow.cpp

HEADERS += \

Node.h \

arraylist.h \

deque.h \

dequelist.h \

doublelinkedlist.h \

form.h \

linkedlist.h \

mainwindow.h

FORMS += \

form.ui \

mainwindow.ui

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

#include "arraylist.h"

ArrayList::ArrayList()

{

arr\_size = 100;

list\_size = 0;

data = new int[arr\_size];

}

ArrayList::~ArrayList()

{

delete []data;

}

void ArrayList::pushBack(int d)

{

add(list\_size, d);

}

void ArrayList::pushFront(int d)

{

add(0, d);

}

void ArrayList::removeFirst()

{

remove(0);

}

void ArrayList::removeLast()

{

remove(list\_size - 1);

}

int ArrayList::contains(int d)

{

for(int i = 0; i < list\_size; i++)

if(data[i] == d)

return i;

return -1;

}

void ArrayList::print()

{

qDebug() << "===========Print ArrayList============";

if(isEmpty()){

qDebug() << "List is empty";

}

for(int i = 0; i < list\_size; i++)

qDebug() << data[i];

qDebug() << "========End Of Print ArrayList========";

qDebug() << "";

}

void ArrayList::clear()

{

list\_size = 0;

}

void ArrayList::add(int index, int d)

{

if(isNeedToResize())

resize();

for(int i = list\_size; i >= index; i--)

data[i + 1] = data[i];

data[index] = d;

list\_size++;

}

void ArrayList::remove(int index)

{

for(int i = index; i < list\_size; i++)

data[i] = data[i + 1];

list\_size--;

}

bool ArrayList::isEmpty()

{

return list\_size == 0;

}

bool ArrayList::isNeedToResize()

{

return arr\_size == list\_size;

}

void ArrayList::resize()

{

int \*newArr = new int[arr\_size\*arr\_size];

for(int i = 0; i < arr\_size; i++)

newArr[i] = data[i];

data = newArr;

arr\_size \*= arr\_size;

}

#ifndef ARRAYLIST\_H

#define ARRAYLIST\_H

#include <QDebug>

#include <cmath>

class ArrayList

{

public:

ArrayList();

~ArrayList();

void pushBack(int d);

void pushFront(int d);

void removeFirst();

void removeLast();

int contains(int d);

void print();

void clear();

private:

int \*data;

int list\_size; //текущий размер списка

int arr\_size; // текущий размер массива

void add(int index, int d);

void remove(int index);

bool isEmpty();

bool isNeedToResize();

void resize();

};

#endif // ARRAYLIST\_H

#include "deque.h"

Deque::Deque(int capacity)

{

arr = new int[capacity];

this->capacity = capacity;

count = 0;

front = -1;

rear = -1;

}

Deque::~Deque()

{

delete[] arr;

}

void Deque::insertFront(int data)

{

if(isFull()){

qDebug() << "Очередь заполнена";

return;

}

if(front == -1){

front = 0;

rear = 0;

}

else if (front == 0)

front = capacity - 1;

else

front--;

arr[front] = data;

count++;

}

void Deque::insertRear(int data)

{

if (isFull()) {

qDebug() << "Очередь заполнена";

return;

}

if (front == -1) {

front = 0;

rear = 0;

}

else if (rear == capacity - 1)

rear = 0;

else

rear++;

arr[rear] = data;

count++;

}

void Deque::deleteFront()

{

if (isEmpty()) {

qDebug() << "Очередь пуста";

return;

}

if (front == rear) {

front = -1;

rear = -1;

}

else if (front == capacity - 1)

front = 0;

else

front++;

count--;

}

void Deque::deleteRear()

{

if (isEmpty()) {

qDebug() << "Очередь пуста";

return;

}

if (front == rear) {

front = -1;

rear = -1;

}

else if (rear == 0)

rear = capacity - 1;

else

rear--;

count--;

}

bool Deque::contains(int data)

{

if (isEmpty()) {

qDebug() << "Очередь пуста";

return false;

}

int stop = rear;

if(stop == capacity)

stop = 0;

int temp;

bool result = false;

while(front != stop){

temp = getFront();

deleteFront();

insertRear(temp);

if (temp == data)

result = true;

}

return result;

}

int Deque::getFront()

{

if(isEmpty()){

qDebug() << "Очередь пуста";

return -1;

}

return arr[front];

}

int Deque::getRear()

{

if(isEmpty()){

qDebug() << "Очередь пуста";

return -1;

}

return arr[rear];

}

void Deque::print()

{

if (isEmpty()) {

qDebug() << "Очередь пуста";

return;

}

int stop = rear + 1;

if(stop == capacity)

stop = 0;

int temp;

while(front != stop){

temp = getFront();

qDebug() << temp;

deleteFront();

insertRear(temp);

}

}

void Deque::clear()

{

count = 0;

front = -1;

rear = -1;

}

bool Deque::isFull()

{

return count == capacity;

}

bool Deque::isEmpty()

{

return count == 0;

}

#ifndef DEQUE\_H

#define DEQUE\_H

#include <QDebug>

class Deque

{

public:

Deque(int capacity = 1000);

~Deque();

void insertFront(int data);

void insertRear(int data);

void deleteFront();

void deleteRear();

bool contains(int data);

int getFront();

int getRear();

void print();

void clear();

private:

int \*arr;

int capacity; // максимальная емкость очереди

int count; // текущий размер очереди

int front; // первый элемент очереди

int rear; // последний элемент очереди

bool isFull();

bool isEmpty();

};

#endif // DEQUE\_H

#include "dequelist.h"

DequeList::DequeList()

{

deque = new Deque();

}

DequeList::DequeList(int dequeSize)

{

deque = new Deque(dequeSize);

}

DequeList::~DequeList()

{

delete deque;

}

void DequeList::pushBack(int d)

{

deque->insertRear(d);

}

void DequeList::pushFront(int d)

{

deque->insertFront(d);

}

void DequeList::removeFirst()

{

deque->deleteFront();

}

void DequeList::removeLast()

{

deque->deleteRear();

}

bool DequeList::contains(int d)

{

return deque->contains(d);

}

void DequeList::print()

{

qDebug() << "===========Print DequeList============";

deque->print();

qDebug() << "========End Of Print DequeList========";

qDebug() << "";

}

void DequeList::clear()

{

deque->clear();

}

#ifndef DEQUELIST\_H

#define DEQUELIST\_H

#include "deque.h"

class DequeList

{

public:

DequeList();

DequeList(int dequeSize);

~DequeList();

void pushBack(int d);

void pushFront(int d);

void removeFirst();

void removeLast();

bool contains(int d);

void print();

void clear();

private:

Deque \*deque;

};

#endif // DEQUELIST\_H

#include "doublelinkedlist.h"

DoubleLinkedList::DoubleLinkedList()

{

head = tail = nullptr;

}

void DoubleLinkedList::pushBack(int d)

{

Node\* newNode = new Node(d);

newNode->prev = tail;

if (tail != nullptr)

tail->next = newNode;

if (head == nullptr)

head = newNode;

tail = newNode;

}

void DoubleLinkedList::pushFront(int d)

{

Node\* newNode = new Node(d);

newNode->next = head;

if(head != nullptr)

head->prev = newNode;

if(tail == nullptr)

tail = newNode;

head = newNode;

}

void DoubleLinkedList::removeFirst()

{

if (head == nullptr)

return;

Node\* node = head->next;

if (node != nullptr)

node->prev = nullptr;

else

tail = nullptr;

delete head;

head = node;

}

void DoubleLinkedList::removeLast()

{

if (tail == nullptr) return;

Node\* node = tail->prev;

if (node != nullptr)

node->next = nullptr;

else

head = nullptr;

delete tail;

tail = node;

}

bool DoubleLinkedList::contains(int d)

{

Node\* node = head;

while(node){

if(node->data == d)

return true;

node = node->next;

}

return false;

}

void DoubleLinkedList::print()

{

qDebug() << "===========Print DoubleLinkedList============";

if(isEmpty()){

qDebug() << "List is empty";

}

Node \*current = head;

while(current){

qDebug() << current->data;

current = current->next;

}

qDebug() << "========End Of Print DoubleLinkedList========";

qDebug() << "";

}

void DoubleLinkedList::clear()

{

if(isEmpty())

return;

Node\* current = head;

Node\* next = nullptr;

while (current != NULL)

{

next = current->next;

free(current);

current = next;

}

head = nullptr;

}

bool DoubleLinkedList::isEmpty()

{

return head == nullptr && tail == nullptr;

}

#include "doublelinkedlist.h"

DoubleLinkedList::DoubleLinkedList()

{

head = tail = nullptr;

}

void DoubleLinkedList::pushBack(int d)

{

Node\* newNode = new Node(d);

newNode->prev = tail;

if (tail != nullptr)

tail->next = newNode;

if (head == nullptr)

head = newNode;

tail = newNode;

}

void DoubleLinkedList::pushFront(int d)

{

Node\* newNode = new Node(d);

newNode->next = head;

if(head != nullptr)

head->prev = newNode;

if(tail == nullptr)

tail = newNode;

head = newNode;

}

void DoubleLinkedList::removeFirst()

{

if (head == nullptr)

return;

Node\* node = head->next;

if (node != nullptr)

node->prev = nullptr;

else

tail = nullptr;

delete head;

head = node;

}

void DoubleLinkedList::removeLast()

{

if (tail == nullptr) return;

Node\* node = tail->prev;

if (node != nullptr)

node->next = nullptr;

else

head = nullptr;

delete tail;

tail = node;

}

bool DoubleLinkedList::contains(int d)

{

Node\* node = head;

while(node){

if(node->data == d)

return true;

node = node->next;

}

return false;

}

void DoubleLinkedList::print()

{

qDebug() << "===========Print DoubleLinkedList============";

if(isEmpty()){

qDebug() << "List is empty";

}

Node \*current = head;

while(current){

qDebug() << current->data;

current = current->next;

}

qDebug() << "========End Of Print DoubleLinkedList========";

qDebug() << "";

}

void DoubleLinkedList::clear()

{

if(isEmpty())

return;

Node\* current = head;

Node\* next = nullptr;

while (current != NULL)

{

next = current->next;

free(current);

current = next;

}

head = nullptr;

}

bool DoubleLinkedList::isEmpty()

{

return head == nullptr && tail == nullptr;

}

#include "linkedlist.h"

LinkedList::LinkedList()

{

first = nullptr;

}

void LinkedList::pushBack(int d)

{

Node \*newNode = new Node(d);

if(isEmpty()){

first = newNode;

last = newNode;

return;

}

last->next = newNode;

last = newNode;

}

void LinkedList::pushFront(int d)

{

Node \*newNode = new Node(d);

if(isEmpty()){

first = newNode;

last = newNode;

return;

}

newNode->next = first;

first = newNode;

}

void LinkedList::removeFirst()

{

if(isEmpty())

return;

Node\* node = first;

first = node->next;

delete node;

}

void LinkedList::removeLast()

{

if(isEmpty())

return;

if(first == last){

removeFirst();

return;

}

Node\* node = first;

while(node->next != last)

node = node->next;

node->next = nullptr;

delete last;

last = node;

}

bool LinkedList::contains(int d)

{

Node\* node = first;

while(node){

if(node->data == d)

return true;

node = node->next;

}

return false;

}

void LinkedList::print()

{

qDebug() << "===========Print LinkedList============";

if(isEmpty()){

qDebug() << "List is empty";

}

Node \*current = first;

while(current){

qDebug() << current->data;

current = current->next;

}

qDebug() << "========End Of Print LinkedList========";

qDebug() << "";

}

void LinkedList::clear()

{

if(isEmpty())

return;

Node\* current = first;

Node\* next = nullptr;

while (current != nullptr)

{

next = current->next;

free(current);

current = next;

}

first = nullptr;

}

QList<QString> \*LinkedList::toQList()

{

QList<QString> \*list = new QList<QString>();

Node \*current = first;

while(current){

list->append(QString::number(current->data));

current = current->next;

}

return list;

}

bool LinkedList::isEmpty()

{

return first == nullptr;

}

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

#include <QDebug>

#include <QList>

#include "Node.h"

class LinkedList

{

public:

LinkedList();

// ~LinkedList();

void pushBack(int d);

void pushFront(int d);

void removeFirst();

void removeLast();

bool contains(int d);

void print();

void clear();

QList<QString> \*toQList();

private:

Node \*first;

Node \*last;

bool isEmpty();

};

#endif // LINKEDLIST\_H

#include "mainwindow.h"

#include <QApplication>

#include <QtWidgets>

#include <QtCharts>

#include <QGraphicsWidget>

#include <QGraphicsView>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

chart = new QChart();

configure();

}

MainWindow::~MainWindow()

{

delete ui;

delete chart;

}

void MainWindow::configure()

{

maxIter = 100;

colIter = 4;

del = 1;

fontChartTitle.setPixelSize(18);

// config title

chart->setTitleFont(fontChartTitle);

chart->setTitleBrush(Qt::black);

chart->setTitle("test chart");

// config legend

chart->legend()->setAlignment(Qt::AlignBottom);

// config fonts

fontAxis.setPixelSize(10);

fontPointLabels.setPixelSize(8);

}

void MainWindow::setChartProps(QChart \*chart, QString name)

{

chart->setTitleFont(fontChartTitle);

chart->setTitle(name);

chart->legend()->setAlignment(Qt::AlignBottom);

chart->legend()->setFont(fontAxis);

chart->setAnimationOptions(QChart::AllAnimations);

}

void MainWindow::setSeriesProps(QLineSeries \*series, QString name, QColor color, bool labelsVisible)

{

series->setPointsVisible(true);

series->setPointLabelsVisible(labelsVisible);

series->setPointLabelsFormat("(@xPoint,@yPoint)");

series->setPointLabelsFont(fontPointLabels);

series->setPointLabelsClipping(false);

QPen pen(color);

pen.setWidth(2);

series->setPen(pen);

series->setName(name);

series->append(0,0);

}

void MainWindow::setAxisProps(QValueAxis \*axis, QString name, int interval, int minorTickCount)

{

axis->setLabelsFont(fontAxis);

axis->setTickType(QValueAxis::TicksDynamic);

axis->setTickAnchor(0);

axis->setTitleText(name);

axis->setTickInterval(interval);

axis->setMinorTickCount(minorTickCount);

}

template<typename T>

void MainWindow::test(T\* list, QValueAxis \*axisX, QValueAxis \*axisY, QLineSeries \*series)

{

QElapsedTimer timer;

int maxX = 0, maxY = axisY->max();

int offset = maxIter / colIter;

int temp = offset;

int time = 0;

switch (methods[ui->comboBox->currentText()])

{

case Method::PushBack:

while(temp <= maxIter){

timer.restart();

for(int i = 0; i < temp; i++)

list->pushBack(i);

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::PushFront:

while(temp <= maxIter){

timer.restart();

for(int i = 0; i < temp; i++)

list->pushFront(i);

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::DeleteFirst:

while(temp <= maxIter){

for(int i = 0; i < temp; i++)

list->pushBack(i);

timer.restart();

for(int i = 0; i < temp; i++)

list->removeFirst();

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::DeleteLast:

while(temp <= maxIter){

for(int i = 0; i < temp; i++)

list->pushBack(i);

timer.restart();

for(int i = 0; i < temp; i++)

list->removeLast();

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::Contains:

while(temp <= maxIter){

for(int i = 0; i < temp; i++)

list->pushFront(i);

timer.restart();

for(int i = 0; i < temp; i++)

list->contains(i);

time = timer.elapsed();

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

}

axisX->setRange(0, maxX);

axisY->setRange(0, maxY);

}

template<typename T>

void MainWindow::testQt(T \*list, QValueAxis \*axisX, QValueAxis \*axisY, QLineSeries \*series)

{

QElapsedTimer timer;

int maxX = 0, maxY = axisY->max();

int offset = maxIter / colIter;

int temp = offset;

int time = 0;

switch (methods[ui->comboBox->currentText()]) {

case Method::PushBack:

while(temp <= maxIter){

timer.restart();

for(int i = 0; i < temp; i++)

list->push\_back(i);

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::PushFront :

while(temp <= maxIter){

timer.restart();

for(int i = 0; i < temp; i++)

list->push\_front(i);

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::DeleteFirst:

while(temp <= maxIter){

for(int i = 0; i < temp; i++)

list->push\_back(i);

timer.restart();

for(int i = 0; i < temp; i++)

list->pop\_front();

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::DeleteLast:

while(temp <= maxIter){

for(int i = 0; i < temp; i++)

list->push\_back(i);

timer.restart();

for(int i = 0; i < temp; i++)

list->pop\_back();

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

case Method::Contains:

while(temp <= maxIter){

for(int i = 0; i < temp; i++)

list->push\_back(i);

timer.restart();

for(int i = 0; i < temp; i++)

list->contains(i);

time = timer.nsecsElapsed() / del;

series->append(temp, time);

list->clear();

if(temp > maxX)

maxX = temp;

if(time > maxY)

maxY = time;

temp += offset;

}

break;

}

axisX->setRange(0, maxX);

axisY->setRange(0, maxY);

}

void MainWindow::on\_pushButton\_clicked()

{

LinkedList linkedList;

ArrayList arrayList;

DoubleLinkedList doubleLinkedList;

DequeList dequeList(maxIter);

QChart \*chart = new QChart();

QValueAxis \*axisX = new QValueAxis();

QValueAxis \*axisY = new QValueAxis();

QLineSeries \*series1 = new QLineSeries();

QLineSeries \*series2 = new QLineSeries();

QLineSeries \*series3 = new QLineSeries();

QLineSeries \*series4 = new QLineSeries();

setChartProps(chart, "Вставка в конец");

setSeriesProps(series1, "LinkedList", QColor(255,0,255),false);

setSeriesProps(series2, "ArrayList", QColor(0,100,100),false);

setSeriesProps(series3, "DoubleLinkedList", QColor(0,0,255),false);

setSeriesProps(series4, "DequeList", QColor(255,100,0),false);

test(&arrayList,axisX,axisY, series2);

test(&doubleLinkedList, axisX,axisY, series3);

test(&dequeList, axisX,axisY, series4);

test(&linkedList, axisX,axisY, series1);

axisX->setRange(0, axisX->max() + axisX->max()\*0.1);

setAxisProps(axisX, "Количество итераций", maxIter / colIter, 0);

axisY->setRange(0, axisY->max() + axisY->max()\*0.1);

setAxisProps(axisY, ui->comboBox\_ci->currentText(), axisY->max() / 20, 1);

chart->addSeries(series1);

chart->addSeries(series2);

chart->addSeries(series3);

chart->addSeries(series4);

chart->addAxis(axisX, Qt::AlignBottom);

chart->addAxis(axisY, Qt::AlignLeft);

series1->attachAxis(axisX);

series1->attachAxis(axisY);

series2->attachAxis(axisX);

series2->attachAxis(axisY);

series3->attachAxis(axisX);

series3->attachAxis(axisY);

series4->attachAxis(axisX);

series4->attachAxis(axisY);

ui->chartView->setChart(chart);

ui->chartView->setRenderHint(QPainter::Antialiasing);

}

void MainWindow::on\_pushButton\_2\_clicked()

{

LinkedList linkedList;

DoubleLinkedList doubleLinkedList;

QChart \*chart = new QChart();

QValueAxis \*axisX = new QValueAxis();

QValueAxis \*axisY = new QValueAxis();

QLineSeries \*series1 = new QLineSeries();

QLineSeries \*series2 = new QLineSeries();

setChartProps(chart, "Вставка в конец");

setSeriesProps(series1, "LinkedList", QColor(255,0,255),true);

setSeriesProps(series2, "DoubleLinkedList", QColor(255,0,100),true);

test(&linkedList, axisX,axisY, series1);

test(&doubleLinkedList, axisX,axisY, series2);

axisX->setRange(0, axisX->max() + axisX->max()\*0.1);

setAxisProps(axisX, "Количество итераций", maxIter / colIter, 0);

axisY->setRange(0, axisY->max() + axisY->max()\*0.05);

setAxisProps(axisY, ui->comboBox\_ci->currentText(), (axisY->max() / 20), 1);

chart->addSeries(series1);

chart->addSeries(series2);

chart->addAxis(axisX, Qt::AlignBottom);

chart->addAxis(axisY, Qt::AlignLeft);

series1->attachAxis(axisX);

series1->attachAxis(axisY);

series2->attachAxis(axisX);

series2->attachAxis(axisY);

ui->chartView->setChart(chart);

ui->chartView->setRenderHint(QPainter::Antialiasing);

}

void MainWindow::on\_pushButton\_qt\_mine\_clicked()

{

try {

LinkedList linkedList;

ArrayList arrayList;

DoubleLinkedList doubleLinkedList;

DequeList dequeList(maxIter);

QLinkedList<int> QLinkedList;

QList<int> QList;

QChart \*chart = new QChart();

QValueAxis \*axisX = new QValueAxis();

QValueAxis \*axisY = new QValueAxis();

QLineSeries \*series1 = new QLineSeries();

QLineSeries \*series2 = new QLineSeries();

QString chartName = "";

switch (qt[ui->comboBox\_qt->currentText()]) {

case 0:

chartName += "Сравнение QLinkedList";

setSeriesProps(series2, "QLinkedList", QColor(255,0,100),false);

testQt(&QLinkedList,axisX,axisY, series2);

break;

case 1:

chartName += "Сравнение QList";

setSeriesProps(series2, "QList", QColor(255,0,100),false);

testQt(&QList, axisX,axisY, series2);

break;

}

switch (mine[ui->comboBox\_mine->currentText()]) {

case 0:

chartName += " c моим LinkedList";

setSeriesProps(series1, "LinkedList", QColor(255,0,255),false);

test(&linkedList,axisX,axisY, series1);

break;

case 1:

chartName += " c моим ArrayList";

setSeriesProps(series1, "ArrayList", QColor(255,0,255),false);

test(&arrayList,axisX,axisY, series1);

break;

case 2:

chartName += " c моим DoubleLinkedList";

setSeriesProps(series1, "DoubleLinkedList", QColor(255,0,255),false);

test(&doubleLinkedList, axisX,axisY, series1);

break;

case 3:

chartName += " c моим DequeList";

setSeriesProps(series1, "DequeList", QColor(255,0,255),false);

test(&dequeList, axisX,axisY, series1);

break;

}

setChartProps(chart, chartName);

axisX->setRange(0, axisX->max() + axisX->max()\*0.1);

setAxisProps(axisX, "Количество итераций", maxIter / colIter, 0);

axisY->setRange(0, axisY->max() + axisY->max()\*0.1);

setAxisProps(axisY, ui->comboBox\_ci->currentText(), (axisY->max() / 20), 1);

chart->addSeries(series1);

chart->addSeries(series2);

chart->addAxis(axisX, Qt::AlignBottom);

chart->addAxis(axisY, Qt::AlignLeft);

series1->attachAxis(axisX);

series1->attachAxis(axisY);

series2->attachAxis(axisX);

series2->attachAxis(axisY);

ui->chartView->setChart(chart);

ui->chartView->setRenderHint(QPainter::Antialiasing);

} catch (...) {

return;

}

}

void MainWindow::on\_comboBox\_maxIter\_currentTextChanged(const QString &arg1)

{

maxIter = arg1.toInt();

}

void MainWindow::on\_comboBox\_colIter\_currentTextChanged(const QString &arg1)

{

colIter = arg1.toInt();

}

void MainWindow::on\_comboBox\_ci\_currentTextChanged(const QString &arg1)

{

del = ci[arg1];

}

void MainWindow::on\_pushButton\_create\_clicked()

{

ui->listWidget->clear();

QString name = ui->lineEdit\_list\_name->text();

LinkedList \*list = new LinkedList();

if(lists.contains(name) || name.isEmpty()){

delete list;

list = nullptr;

return;

}

if(ui->checkBox){

int kol = ui->spinBox->value();

for(int i = 0; i < kol; i++){

int el = QRandomGenerator::global()->generate();

list->pushBack(el);

ui->listWidget->addItem(QString::number(el));

}

}

lists.insert(name, list);

ui->comboBox\_select->addItem(name);

ui->lineEdit\_list\_name->clear();

ui->checkBox->setChecked(false);

ui->spinBox->setValue(20);

}

void MainWindow::on\_comboBox\_select\_currentTextChanged(const QString &arg1)

{

ui->listWidget->clear();

QList<QString> \*list = lists[arg1]->toQList();

for( int i = 0; i < list->count(); i++)

ui->listWidget->addItem(list->value(i));

}

void MainWindow::on\_pushButton\_add\_clicked()

{

int el = ui->lineEdit\_add->text().toInt();

QString list = ui->comboBox\_select->currentText();

if(ui->radioButton\_add\_first->isChecked())

lists[list]->pushFront(el);

else if(ui->radioButton\_add\_last->isChecked())

lists[list]->pushBack(el);

ui->lineEdit\_add->clear();

on\_comboBox\_select\_currentTextChanged(list);

}

void MainWindow::on\_pushButton\_del\_clicked()

{

QString list = ui->comboBox\_select->currentText();

if(ui->radioButton\_del\_first->isChecked())

lists[list]->removeFirst();

else if(ui->radioButton\_del\_last->isChecked())

lists[list]->removeLast();

on\_comboBox\_select\_currentTextChanged(list);

}

void MainWindow::on\_pushButton\_contains\_clicked()

{

int el = ui->lineEdit\_contains->text().toInt();

if(lists[ui->comboBox\_select->currentText()]->contains(el))

ui->label\_contains->setText("Элемент найден");

else

ui->label\_contains->setText("Элемент не найден");

ui->lineEdit\_contains->clear();

}

void MainWindow::on\_pushButton\_del\_list\_clicked()

{

ui->listWidget->clear();

lists.remove(ui->comboBox\_select->currentText());

ui->comboBox\_select->removeItem(ui->comboBox\_select->currentIndex());

}

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include "linkedlist.h"

#include "arraylist.h"

#include "doublelinkedlist.h"

#include "dequelist.h"

#include "form.h"

#include <QMainWindow>

#include <QElapsedTimer>

#include <QtCharts>

#include <QRgb>

#include <QPainter>

#include <QLineSeries>

#include <QObject>

#include <QtGlobal>

#include <QLinkedList>

#include <QList>

#include <QRandomGenerator>

#include <cmath>

enum class Method {PushBack, PushFront, DeleteFirst, DeleteLast, Contains};

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_qt\_mine\_clicked();

void on\_comboBox\_maxIter\_currentTextChanged(const QString &arg1);

void on\_comboBox\_colIter\_currentTextChanged(const QString &arg1);

void on\_comboBox\_ci\_currentTextChanged(const QString &arg1);

void on\_pushButton\_create\_clicked();

void on\_comboBox\_select\_currentTextChanged(const QString &arg1);

void on\_pushButton\_add\_clicked();

void on\_pushButton\_del\_clicked();

void on\_pushButton\_contains\_clicked();

void on\_pushButton\_del\_list\_clicked();

private:

Ui::MainWindow \*ui;

QChart \*chart;

QFont fontChartTitle;

QFont fontAxis;

QFont fontPointLabels;

int maxIter;

int colIter;

int del;

QMap<QString, Method> methods{

{"pushBack", Method::PushBack},

{"pushFront", Method::PushFront},

{"deleteFirst", Method::DeleteFirst},

{"deleteLast", Method::DeleteLast},

{"contains", Method::Contains}

};

QMap<QString, int> qt{

{"QLinkedList", 0},

{"QList", 1}

};

QMap<QString, int> mine{

{"LinkedList", 0},

{"ArrayList", 1},

{"DoubleLinkedList", 2},

{"DequeList", 3}

};

QMap<QString, int> ci{

{"миллисекунда", 1000000},

{"микросекунда", 1000},

{"наносекунды", 1},

};

QMap<QString, LinkedList\*> lists;

void configure();

void configureSeries();

template<typename T>

void test(T\* list, QValueAxis \*axisX, QValueAxis \*axisY, QLineSeries \*series);

template<typename T>

void testQt(T\* list, QValueAxis \*axisX, QValueAxis \*axisY, QLineSeries \*series);

void setChartProps(QChart \*chart, QString name);

void setSeriesProps(QLineSeries \*series, QString name, QColor color, bool labelsVisible);

void setAxisProps(QValueAxis \*axis, QString name, int interval, int minorTickCount);

};

#endif // MAINWINDOW\_H

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>1014</width>

<height>751</height>

</rect>

</property>

<property name="windowTitle">

<string>Курсовая работа</string>

</property>

<property name="windowIcon">

<iconset>

<normaloff>icons8-список-100.png</normaloff>icons8-список-100.png</iconset>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QTabWidget" name="tabWidget">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>1021</width>

<height>751</height>

</rect>

</property>

<property name="currentIndex">

<number>0</number>

</property>

<widget class="QWidget" name="tab\_2">

<attribute name="title">

<string>Работа со списком</string>

</attribute>

<widget class="QGroupBox" name="groupBox\_4">

<property name="geometry">

<rect>

<x>20</x>

<y>60</y>

<width>521</width>

<height>201</height>

</rect>

</property>

<property name="title">

<string>Создание списка:</string>

</property>

<widget class="QPushButton" name="pushButton\_create">

<property name="geometry">

<rect>

<x>220</x>

<y>170</y>

<width>71</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>Создать</string>

</property>

</widget>

<widget class="QLabel" name="label\_9">

<property name="geometry">

<rect>

<x>10</x>

<y>50</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Название:</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_list\_name">

<property name="geometry">

<rect>

<x>120</x>

<y>40</y>

<width>391</width>

<height>31</height>

</rect>

</property>

</widget>

<widget class="Line" name="line">

<property name="geometry">

<rect>

<x>0</x>

<y>90</y>

<width>521</width>

<height>20</height>

</rect>

</property>

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

</widget>

<widget class="QCheckBox" name="checkBox">

<property name="geometry">

<rect>

<x>10</x>

<y>120</y>

<width>291</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Заполнить случайными элементами</string>

</property>

</widget>

<widget class="QSpinBox" name="spinBox">

<property name="geometry">

<rect>

<x>320</x>

<y>120</y>

<width>161</width>

<height>26</height>

</rect>

</property>

<property name="value">

<number>20</number>

</property>

</widget>

<widget class="Line" name="line\_3">

<property name="geometry">

<rect>

<x>0</x>

<y>150</y>

<width>521</width>

<height>20</height>

</rect>

</property>

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

</widget>

</widget>

<widget class="QGroupBox" name="groupBox\_6">

<property name="geometry">

<rect>

<x>60</x>

<y>280</y>

<width>421</width>

<height>381</height>

</rect>

</property>

<property name="title">

<string>Операции:</string>

</property>

<widget class="QLabel" name="label\_7">

<property name="geometry">

<rect>

<x>10</x>

<y>40</y>

<width>141</width>

<height>17</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Добавить элемент:</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_add">

<property name="geometry">

<rect>

<x>170</x>

<y>30</y>

<width>241</width>

<height>31</height>

</rect>

</property>

<property name="alignment">

<set>Qt::AlignJustify|Qt::AlignVCenter</set>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_add">

<property name="geometry">

<rect>

<x>150</x>

<y>100</y>

<width>101</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>Добавить</string>

</property>

</widget>

<widget class="QRadioButton" name="radioButton\_add\_first">

<property name="geometry">

<rect>

<x>70</x>

<y>70</y>

<width>112</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Первый</string>

</property>

</widget>

<widget class="QRadioButton" name="radioButton\_add\_last">

<property name="geometry">

<rect>

<x>210</x>

<y>70</y>

<width>112</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Последний</string>

</property>

</widget>

<widget class="QLabel" name="label\_10">

<property name="geometry">

<rect>

<x>10</x>

<y>160</y>

<width>141</width>

<height>17</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Удалить элемент:</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_del">

<property name="geometry">

<rect>

<x>150</x>

<y>220</y>

<width>101</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>Удалить</string>

</property>

</widget>

<widget class="QRadioButton" name="radioButton\_del\_last">

<property name="geometry">

<rect>

<x>260</x>

<y>160</y>

<width>112</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Последний</string>

</property>

</widget>

<widget class="QRadioButton" name="radioButton\_del\_first">

<property name="geometry">

<rect>

<x>150</x>

<y>160</y>

<width>112</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Первый</string>

</property>

</widget>

<widget class="Line" name="line\_2">

<property name="geometry">

<rect>

<x>0</x>

<y>250</y>

<width>421</width>

<height>20</height>

</rect>

</property>

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_contains">

<property name="geometry">

<rect>

<x>170</x>

<y>270</y>

<width>241</width>

<height>31</height>

</rect>

</property>

<property name="alignment">

<set>Qt::AlignJustify|Qt::AlignVCenter</set>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_contains">

<property name="geometry">

<rect>

<x>150</x>

<y>310</y>

<width>101</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>Проверить</string>

</property>

</widget>

<widget class="QLabel" name="label\_11">

<property name="geometry">

<rect>

<x>10</x>

<y>280</y>

<width>151</width>

<height>17</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Проверить наличие:</string>

</property>

</widget>

<widget class="QLabel" name="label\_res">

<property name="geometry">

<rect>

<x>20</x>

<y>350</y>

<width>91</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Результат:</string>

</property>

</widget>

<widget class="QLabel" name="label\_contains">

<property name="geometry">

<rect>

<x>140</x>

<y>350</y>

<width>271</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="Line" name="line\_4">

<property name="geometry">

<rect>

<x>0</x>

<y>130</y>

<width>421</width>

<height>20</height>

</rect>

</property>

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

</widget>

</widget>

<widget class="QListWidget" name="listWidget">

<property name="geometry">

<rect>

<x>590</x>

<y>50</y>

<width>371</width>

<height>651</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>15</pointsize>

</font>

</property>

</widget>

<widget class="QComboBox" name="comboBox\_select">

<property name="geometry">

<rect>

<x>590</x>

<y>10</y>

<width>311</width>

<height>31</height>

</rect>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_del\_list">

<property name="geometry">

<rect>

<x>910</x>

<y>10</y>

<width>51</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string/>

</property>

<property name="icon">

<iconset>

<normaloff>icons8-удалить-100.png</normaloff>icons8-удалить-100.png</iconset>

</property>

<property name="iconSize">

<size>

<width>20</width>

<height>20</height>

</size>

</property>

</widget>

</widget>

<widget class="QWidget" name="tab">

<attribute name="title">

<string>Графики</string>

</attribute>

<widget class="QChartView" name="chartView" native="true">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>661</width>

<height>721</height>

</rect>

</property>

</widget>

<widget class="QGroupBox" name="groupBox">

<property name="geometry">

<rect>

<x>680</x>

<y>300</y>

<width>321</width>

<height>181</height>

</rect>

</property>

<property name="title">

<string>Сравнение Qt с моим:</string>

</property>

<widget class="QComboBox" name="comboBox\_qt">

<property name="geometry">

<rect>

<x>60</x>

<y>40</y>

<width>251</width>

<height>25</height>

</rect>

</property>

<item>

<property name="text">

<string>QList</string>

</property>

</item>

<item>

<property name="text">

<string>QLinkedList</string>

</property>

</item>

</widget>

<widget class="QLabel" name="label\_5">

<property name="geometry">

<rect>

<x>20</x>

<y>40</y>

<width>31</width>

<height>20</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>QT:</string>

</property>

</widget>

<widget class="QLabel" name="label\_6">

<property name="geometry">

<rect>

<x>10</x>

<y>90</y>

<width>51</width>

<height>20</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Mine:</string>

</property>

</widget>

<widget class="QComboBox" name="comboBox\_mine">

<property name="geometry">

<rect>

<x>60</x>

<y>90</y>

<width>251</width>

<height>25</height>

</rect>

</property>

<item>

<property name="text">

<string>LinkedList</string>

</property>

</item>

<item>

<property name="text">

<string>DoubleLinkedList</string>

</property>

</item>

<item>

<property name="text">

<string>ArrayList</string>

</property>

</item>

<item>

<property name="text">

<string>DequeList</string>

</property>

</item>

</widget>

<widget class="QPushButton" name="pushButton\_qt\_mine">

<property name="geometry">

<rect>

<x>10</x>

<y>140</y>

<width>301</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>График</string>

</property>

</widget>

</widget>

<widget class="QGroupBox" name="groupBox\_3">

<property name="geometry">

<rect>

<x>680</x>

<y>500</y>

<width>321</width>

<height>101</height>

</rect>

</property>

<property name="title">

<string>Дополнительные сравнения:</string>

</property>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>

<x>10</x>

<y>30</y>

<width>301</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>Все мои 4 реализации</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_2">

<property name="geometry">

<rect>

<x>10</x>

<y>70</y>

<width>301</width>

<height>25</height>

</rect>

</property>

<property name="text">

<string>реализации node</string>

</property>

</widget>

</widget>

<widget class="QGroupBox" name="groupBox\_2">

<property name="geometry">

<rect>

<x>680</x>

<y>10</y>

<width>321</width>

<height>261</height>

</rect>

</property>

<property name="title">

<string>Настройки</string>

</property>

<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>10</x>

<y>30</y>

<width>91</width>

<height>17</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Операция:</string>

</property>

</widget>

<widget class="QComboBox" name="comboBox">

<property name="geometry">

<rect>

<x>100</x>

<y>30</y>

<width>211</width>

<height>31</height>

</rect>

</property>

<item>

<property name="text">

<string>pushBack</string>

</property>

</item>

<item>

<property name="text">

<string>pushFront</string>

</property>

</item>

<item>

<property name="text">

<string>deleteBack</string>

</property>

</item>

<item>

<property name="text">

<string>deleteFront</string>

</property>

</item>

<item>

<property name="text">

<string>contains</string>

</property>

</item>

<item>

<property name="text">

<string/>

</property>

</item>

</widget>

<widget class="QLabel" name="label\_2">

<property name="geometry">

<rect>

<x>30</x>

<y>80</y>

<width>271</width>

<height>20</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Максимальное количество итераций:</string>

</property>

</widget>

<widget class="QComboBox" name="comboBox\_maxIter">

<property name="geometry">

<rect>

<x>60</x>

<y>100</y>

<width>211</width>

<height>25</height>

</rect>

</property>

<item>

<property name="text">

<string>100</string>

</property>

</item>

<item>

<property name="text">

<string>1000</string>

</property>

</item>

<item>

<property name="text">

<string>10000</string>

</property>

</item>

<item>

<property name="text">

<string>50000</string>

</property>

</item>

<item>

<property name="text">

<string>100000</string>

</property>

</item>

</widget>

<widget class="QComboBox" name="comboBox\_colIter">

<property name="geometry">

<rect>

<x>60</x>

<y>160</y>

<width>211</width>

<height>25</height>

</rect>

</property>

<item>

<property name="text">

<string>5</string>

</property>

</item>

<item>

<property name="text">

<string>4</string>

</property>

</item>

<item>

<property name="text">

<string>3</string>

</property>

</item>

<item>

<property name="text">

<string>2</string>

</property>

</item>

<item>

<property name="text">

<string>1</string>

</property>

</item>

</widget>

<widget class="QLabel" name="label\_3">

<property name="geometry">

<rect>

<x>40</x>

<y>140</y>

<width>261</width>

<height>20</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Количество промежуточных точек:</string>

</property>

</widget>

<widget class="QLabel" name="label\_4">

<property name="geometry">

<rect>

<x>60</x>

<y>200</y>

<width>221</width>

<height>20</height>

</rect>

</property>

<property name="font">

<font>

<pointsize>11</pointsize>

</font>

</property>

<property name="text">

<string>Единица измерения времени</string>

</property>

</widget>

<widget class="QComboBox" name="comboBox\_ci">

<property name="geometry">

<rect>

<x>60</x>

<y>220</y>

<width>211</width>

<height>25</height>

</rect>

</property>

<item>

<property name="text">

<string>наносекунда</string>

</property>

</item>

<item>

<property name="text">

<string>микросекунда</string>

</property>

</item>

</widget>

</widget>

</widget>

</widget>

</widget>

</widget>

<customwidgets>

<customwidget>

<class>QChartView</class>

<extends>QWidget</extends>

<header>qchartview.h</header>

<container>1</container>

</customwidget>

</customwidgets>

<resources/>

<connections/>

</ui>

#ifndef NODE\_H

#define NODE\_H

struct Node{

int data;

Node \*prev,\*next;

Node(int \_data) : data(\_data), prev(nullptr), next(nullptr){}

};

#endif // NODE\_H