**Учреждение образования «Барановичский технологический колледж» Белкоопсоюза**

Отметка о допуске к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отметка за защиту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Курсовой проект**

по учебному предмету «Основы алгоритмизации и программирования»

Тема: «Разработка класса для реализации различных видов сортировки массива (выбором, пузырьком, перестановкой и т.д.) на языке программирования Visual С++»

**Исполнитель:**

учащийся II курса группы 283

Пилипчук Илья Игоревич

Фамилия, имя, отчество

**Руководитель:**

Преподаватель 2 категории

Ученое звание, ученая степень, должность

Камленок И.А.

Барановичи 2024

Барановичи 2024

**Учреждение образования «Барановичский технологический колледж» Белкоопсоюза**

УТВЕРЖДАЮ

Председатель предметной

(цикловой) комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

**ЗАДАНИЕ**

**по курсовому проекту (курсовой работе)**

Обучающемуся \_\_\_\_\_\_\_\_\_\_\_\_\_\_Пилипчук Илья Игоревич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, собственное имя, отчество)

Курса \_\_II\_\_ учебной группы \_\_\_\_283\_\_\_\_

по учебному предмету, модулю «Основы алгоритмизации и программирования»

Тема курсового проекта (курсовой работы) «Разработка класса для реализации различных видов сортировки массива (выбором, пузырьком, перестановкой и т.д.) на языке программирования Visual С++»

Исходные данные \_Visual Studio\_

Содержание курсового проекта (курсовой работы)

Пояснительная записка

|  |  |
| --- | --- |
| Содержание разделов | Срок выполнения |
| Введение |  |
| 1. Теоретическая часть |  |
| 1.1 Постановка задачи |  |
| 1.2 Объектно-ориентированный анализ и проектирование системы на языке UML |  |
| 2 Практический раздел |  |
| 2.1 Описание объектов системы, их свойств и методов (диаграмма классов) |  |
| 2.2 Инструкция пользователю |  |
| 2.3 Тестирование программного продукта |  |
| Заключение |  |
| Список используемых источников |  |
| Приложение |  |

Графическая (практическая) часть курсового проекта (курсовой работы)

Лист – \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Лист – \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель

курсового проекта

(курсовой работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись) (подпись) (инициалы (инициал собственного имени), фамилия)

Подпись обучающегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «БАРАНОВИЧСКИЙ ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ» БЕЛКООПСОЮЗА**

РЕЦЕНЗИЯ

на курсовой проект

Учащегося \_\_\_Пилипчука Ильи Игоревича\_

специальности 5-04-0612-02 «Разработка и сопровождение программного обеспечения информационных систем»

Курс \_\_\_2\_\_\_группа \_\_\_283\_\_\_\_

Учебный предмет «Основы алгоритмизации и программирования»

Рецензент Камленок И. А.

**Текст рецензии:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Подпись преподавателя-рецензента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc170304466)

[**1** **ТЕОРЕТИЧЕСКАЯ РАЗДЕЛ** 6](#_Toc170304467)

[**1.1.** **Постановка задачи** 6](#_Toc170304468)

[**1.2.** **Объектно-ориентированный анализ и проектирование системы на языке UML** 6](#_Toc170304469)

[**2.** **ПРАКТИЧЕСКАЯ РАЗДЕЛ** 9](#_Toc170304470)

[**2.1.** **Краткая характеристика инструментальной программной среды** 9](#_Toc170304471)

[**2.2.** **Описание объектов системы, их свойств и методов (диаграмма классов)** 10](#_Toc170304472)

[**2.3.** **Инструкция пользователю** 16](#_Toc170304473)

[**2.4.** **Тестирование программного продукта** 19](#_Toc170304474)

[**ЗАКЛЮЧЕНИЕ** 22](#_Toc170304475)

[**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 23](#_Toc170304476)

[**ПРИЛОЖЕНИЕ** 25](#_Toc170304477)

# **ВВЕДЕНИЕ**

В современном мире, где информация становится все более ценным ресурсом, способность быстро и эффективно обрабатывать большие объемы данных становится критически важной. Алгоритмы сортировки играют ключевую роль в этом процессе, поскольку они используются во многих областях, включая поиск данных, машинное обучение, оптимизацию баз данных и многое другое. Это особенно важно в контексте растущего объема данных, которые требуют обработки в современных приложениях. Разработка класса для реализации различных видов сортировки на C++ является актуальной задачей, так как это помогает улучшить производительность и эффективность программ, обеспечивая быстрый и надежный доступ к данным. Это также отражает тенденцию к использованию объектно-ориентированного подхода в программировании, который обеспечивает большую гибкость и повторное использование кода.

Объектом исследования являются алгоритмы сортировки, которые являются фундаментальными элементами компьютерной науки и используются во многих областях информационных технологий. Они служат основой для многих операций обработки данных и являются ключевыми для эффективного функционирования многих систем. Предметом исследования является разработка класса для реализации этих алгоритмов на C++. Это включает в себя изучение различных алгоритмов сортировки, таких как сортировка выбором, пузырьком, перестановкой и т.д., и их реализацию в виде класса на C++. Это также включает в себя анализ эффективности этих алгоритмов и их применимости в различных контекстах. Это важно для понимания того, как эти алгоритмы могут быть использованы для решения реальных задач и как они могут быть оптимизированы для улучшения производительности.

Целью данного проекта является разработка и реализация класса для различных видов сортировки массива на языке программирования Visual C++, а также анализ и сравнение эффективности этих алгоритмов. Это включает в себя разработку детального плана разработки, написание кода, проведение тестов и анализ результатов.

Для достижения этой цели ставятся следующие задачи:

* изучить основные алгоритмы сортировки,
* разработать класс для реализации этих алгоритмов на C++,
* и сравнить эффективность различных алгоритмов сортировки.

Это включает в себя изучение теории алгоритмов сортировки, разработку дизайна класса, написание кода для реализации алгоритмов, проведение тестов для проверки корректности и эффективности реализации, а также анализ результатов для определения наиболее эффективных алгоритмов.

Работа базируется на теории алгоритмов и структур данных, а также на методологии объектно-ориентированного программирования. Это включает в себя изучение и применение принципов проектирования классов, инкапсуляции, наследования и полиморфизма. Это также включает в себя изучение и применение методов анализа и синтеза для разработки и оптимизации алгоритмов сортировки. Это обеспечивает теоретическую основу для разработки класса и позволяет применить наиболее эффективные методы и подходы для решения задачи.

В работе используются методы анализа и синтеза для изучения и разработки алгоритмов сортировки, методы тестирования программного обеспечения для проверки корректности и эффективности реализации, а также методы сравнительного анализа для оценки производительности различных алгоритмов сортировки. Это включает в себя использование различных инструментов и техник для проведения тестов и анализа результатов, включая инструменты для профилирования кода, тестирования производительности и анализа сложности алгоритмов.

Теоретическая значимость работы заключается в систематизации знаний о различных алгоритмах сортировки и методах их реализации на C++. Это включает в себя изучение теории алгоритмов сортировки, анализ различных методов их реализации и сравнение их эффективности. Практическая значимость заключается в разработке универсального класса для реализации этих алгоритмов, который может быть использован в различных приложениях для обработки и анализа данных. Это может включать в себя базы данных, системы управления данными, научные исследования, машинное обучение и многое другое. Это также может способствовать улучшению производительности и эффективности этих приложений, что в свою очередь может привести к улучшению качества услуг и продуктов, предлагаемых пользователям.

# **ТЕОРЕТИЧЕСКАЯ РАЗДЕЛ**

* 1. **Постановка задачи**

Разработка класса для реализации различных видов сортировки массива (выбором, пузырьком, перестановкой и т.д.) на языке программирования Visual С++.

* Сортировка выбором (Selection Sort)
* Пузырьковая сортировка (Bubble Sort)
* Сортировка вставками (Insertion Sort)
* Быстрая сортировка (Quick Sort)
* Сортировка слиянием (Merge Sort)
* Сортировка перестановкой (Shell Sort)
  1. **Объектно-ориентированный анализ и проектирование системы на языке UML**

Объектно-ориентированный анализ и проектирование (ООАП) с использованием языка UML (Unified Modeling Language) – это методология, применяемая для разработки сложных программных систем. Она включает несколько этапов: анализ требований, проектирование системы и её реализацию. Вот основные аспекты и понятия:

1. Объектно-ориентированный анализ (ООА):

* Идентификация объектов: Определение основных объектов системы, их атрибутов и методов.
* Анализ требований: Сбор и документирование требований к системе от всех заинтересованных сторон.
* Создание моделей использования (Use Case Models): Описание функциональных возможностей системы с точки зрения пользователя.

1. Объектно-ориентированное проектирование (ООП):

* Создание статических моделей: Построение классов, определение их атрибутов, методов и связей (ассоциации, агрегации, композиции, наследование).
* Динамические модели: Описание поведения системы с помощью диаграмм последовательностей, взаимодействий, состояний и активностей.
* Архитектурные модели: Разработка общей архитектуры системы, включая модули, компоненты и их взаимодействие.

1. Использование UML:

* Диаграммы классов: Отображают классы, их атрибуты, методы и связи между ними.
* Диаграммы объектов: Показаны конкретные экземпляры классов и их связи.
* Диаграммы последовательностей: Отображают взаимодействие объектов в хронологическом порядке.
* Диаграммы состояний: Показывают возможные состояния объектов и переходы между этими состояниями.
* Диаграммы активности: Описывают поток управления и данные внутри системы.
* Диаграммы компонентов и развёртывания: Отображают физическую структуру системы, её компоненты и распределение на аппаратных ресурсах.

Процесс ООАП с UML:

1. Сбор и анализ требований: Создание моделей использования для понимания функциональных требований.
2. Построение концептуальной модели: Определение основных понятий и связей в предметной области.
3. Проектирование системы: Разработка детализированных моделей классов и объектов.
4. Детализация поведения: Использование диаграмм последовательностей и состояний для описания динамики системы.
5. Создание архитектурной модели: Определение компонентов системы и их взаимодействия.

Методология ООАП с UML помогает структурировать процесс разработки программного обеспечения, улучшая его управляемость и поддерживаемость.

Объектно-ориентированное программирование (ООП) — это парадигма программирования, основанная на концепции объектов, которые могут содержать данные и код для их обработки. Основные принципы ООП включают инкапсуляцию, наследование, полиморфизм и абстракцию:

1. Инкапсуляция:

* Объединение данных (полей) и методов (функций), которые работают с этими данными, в единый объект.

1. Наследование:

* Способность одного класса (наследника) использовать свойства и методы другого класса (родителя).

1. Полиморфизм:

* Способность объектов различных классов обрабатывать сообщения одного и того же типа по-разному. [14]

# **ПРАКТИЧЕСКАЯ РАЗДЕЛ**

* 1. **Краткая характеристика инструментальной программной среды**

Visual Studio — это интегрированная среда разработки (IDE), разработанная компанией Microsoft, которая поддерживает множество языков программирования и предназначена для создания приложений различного типа. Вот основные характеристики Visual Studio:

1. Поддержка различных языков:
   * Включает поддержку C#, C++, Python, JavaScript, TypeScript, и других языков.
2. Многофункциональность:
   * Поддерживает разработку веб-приложений, настольных приложений, мобильных приложений, игр и облачных сервисов.
3. Редактор кода:
   * Включает функции подсветки синтаксиса, автозаполнения кода, рефакторинга и интеллектуального анализа кода.
4. Отладка и тестирование:
   * Встроенные инструменты для отладки, включая пошаговую отладку, точки останова, и анализ памяти.
   * Поддержка модульного тестирования и интеграционного тестирования.
5. Интеграция с системами контроля версий:
   * Поддерживает Git, Team Foundation Version Control (TFVC) и другие системы управления версиями.
6. Инструменты для совместной работы:
   * Включает функции для совместной разработки, такие как Live Share, позволяющая нескольким разработчикам одновременно работать над одним проектом.
7. Расширяемость:
   * Поддерживает множество расширений, доступных через Visual Studio Marketplace, для добавления дополнительных функций и интеграции с другими инструментами.
8. Интеграция с Azure:
   * Облегчает разработку, развертывание и мониторинг приложений, работающих в облаке Microsoft Azure. [15]
   1. **Описание объектов системы, их свойств и методов (диаграмма классов)**
9. array<int>^ allElements
   * Тип: Массив целых чисел
   * Описание: Хранит все элементы DataGridView в виде одномерного массива.
10. DataGridView dataGridView1
    * Тип: DataGridView
    * Свойства:
      + RowCount: Количество строк
      + ColumnCount: Количество столбцов
    * Описание: Используется для отображения и редактирования сетки значений.
11. TextBox textBox1, textBox2, ..., textBox14
    * Тип: TextBox
    * Свойства:
      + Text: Текстовое содержание
    * Описание: Поля ввода и отображения отсортированных массивов.
12. CheckBox checkBox1, checkBox2, ..., checkBox12
    * Тип: CheckBox
    * Свойства:
      + Checked: Состояние флажка
    * Описание: Выбор алгоритма сортировки для строк и столбцов.
13. Random rand

* Тип: Random^
* Описание: Генератор случайных чисел.

Методы

1. button1\_Click
   * Параметры: (System::Object^ sender, System::EventArgs^ e)
   * Описание: Инициализация DataGridView случайными значениями и их сохранение в allElements.
2. FillTextBox

* Параметры: (TextBox^ textBox, array<int>^ arr)
* Описание: Заполнение TextBox элементами массива в виде строки.

1. SortBubbleRows/Columns

* Параметры: (array<int>^ arr, int rows/cols, TextBox^ outputBox)
* Описание: Сортировка строк/столбцов массива методом пузырьковой сортировки и отображение результата.

1. SortInsertionRows/Columns

* Параметры: (array<int>^ arr, int rows/cols, TextBox^ outputBox)
* Описание: Сортировка строк/столбцов массива методом вставками и отображение результата.

1. SortRows/ColumnsQuick

* Параметры: (array<int>^ arr, int rows/cols, TextBox^ outputBox)
* Описание: Сортировка строк/столбцов массива методом быстрой сортировки и отображение результата.

1. SortRows/ColumnsMerge

* Параметры: (array<int>^ arr, int rows/cols, TextBox^ outputBox)
* Описание: Сортировка строк/столбцов массива методом сортировки слиянием и отображение результата.

1. SortRows/ColumnsShell

* Параметры: (array<int>^ arr, int rows/cols, TextBox^ outputBox)
* Описание: Сортировка строк/столбцов массива методом Шелла и отображение результата.

1. SortRow/ColumnsSelection

* Параметры: (array<int>^ arr, int rows/cols, TextBox^ outputBox)
* Описание: Сортировка строк/столбцов массива методом выбора и отображение результата.

1. BubbleSort

* Параметры: (array<int>^ arr)
* Описание: Пузырьковая сортировка массива.

1. SelectionSort

* Параметры: (array<int>^ arr)
* Описание: Сортировка массива методом выбора.

1. InsertionSort

* Параметры: (array<int>^ arr)
* Описание: Сортировка массива методом вставок.

1. QuickSort

* Параметры: (array<int>^ arr, int low, int high)
* Описание: Быстрая сортировка массива.

1. Partition

* Параметры: (array<int>^ arr, int low, int high)
* Описание: Разделение массива для быстрой сортировки.

1. MergeSort

* Параметры: (array<int>^ arr, int l, int r)
* Описание: Сортировка массива методом слияния.

1. Merge

* Параметры: (array<int>^ arr, int l, int m, int r)
* Описание: Слияние массивов для сортировки методом слияния.

1. ShellSort

* Параметры: (array<int>^ arr, int size)
* Описание: Сортировка массива методом Шелла.

Обработчики событий

1. button3\_Click
   * Описание: Сортировка массива методом пузырьковой сортировки для строк и столбцов, отображение результата.
2. button2\_Click
   * Описание: Сортировка массива методом выбора для строк и столбцов, отображение результата.
3. button4\_Click
   * Описание: Сортировка массива методом вставок для строк и столбцов, отображение результата.
4. button5\_Click
   * Описание: Сортировка массива методом быстрой сортировки для строк и столбцов, отображение результата.
5. button6\_Click
   * Описание: Сортировка массива методом слияния для строк и столбцов, отображение результата.
6. button7\_Click
   * Описание: Сортировка массива методом Шелла для строк и столбцов, отображение результата.

На диаграмме на рисунке 2.1 показано что пользователь может: запуск программы, ввод размера массива, выбор сортировки, выбор сортировки (по столбикам или по строкам), сортировка, завершение программы.

Рисунок 2.1 – Диаграмма классов пользователя

В блок схеме на рисунке 2.2 демонстрируется: запуск программы, ввод размера массива, создать массив, выбор сортировки, выбор сортировки (по столбикам или по строкам), сортировка, завершение программы.



Рисунок 2.2 – Блок схема

На рисунке 2.3 показана диаграмма классов Sorting, которая содержит в себе такие методы: BubbleSort, InsertionSort, MergeSort, QuickSort, SelectionSort, ShellSort, SortBubbleColumns, SortBubbleRows, SortColumnsSelection, SortRowSelection, SortInsertionRows, SortInsertionColumns, SortRowsMerge, SortColumnsMerge, SortRowsQuick, SortColumnsQuick, SortRowsShell, SortColumnsShell.

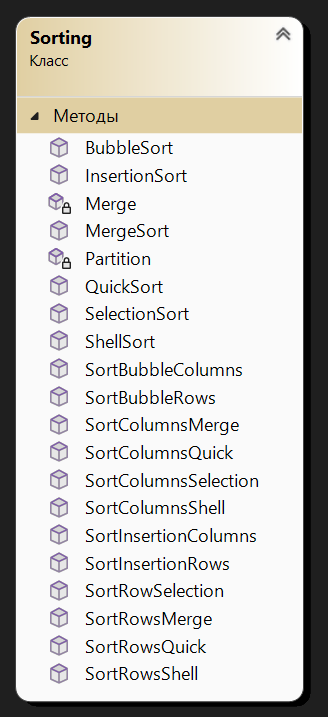


Рисунок 2.3 – Диаграмма классов

* 1. **Инструкция пользователю**

Запуск программы:

* + Запустите исполняемый файл программы. Откроется главное окно приложения с таблицей (dataGridView), различными вкладками, кнопками и флажками.

Ввод данных в таблицу:

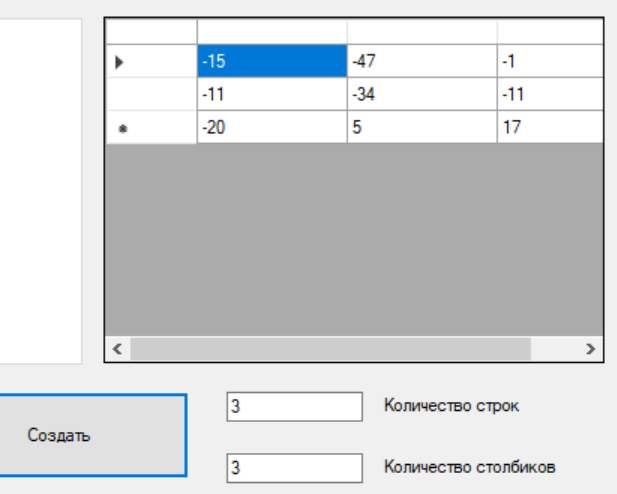
* + Введите данные, которые вы хотите отсортировать, в таблицу. Таблица поддерживает любое количество строк и столбцов как на рисунке 2.4.

Рисунок 2.4 – Ввод данных

Выбор метода сортировки:

* Перейдите на вкладку, соответствующую нужному методу сортировки:
  + Вкладка "BubbleSort" для пузырьковой сортировки.
  + Вкладка "SelectionSort" для сортировки методом выбора.
  + Вкладка "InsertionSort" для сортировки методом вставок.
  + Вкладка "QuickSort" для быстрой сортировки.
  + Вкладка "MergeSort" для сортировки методом слияния.
  + Вкладка "ShellSort" для сортировки методом Шелла.

Настройка параметров сортировки:

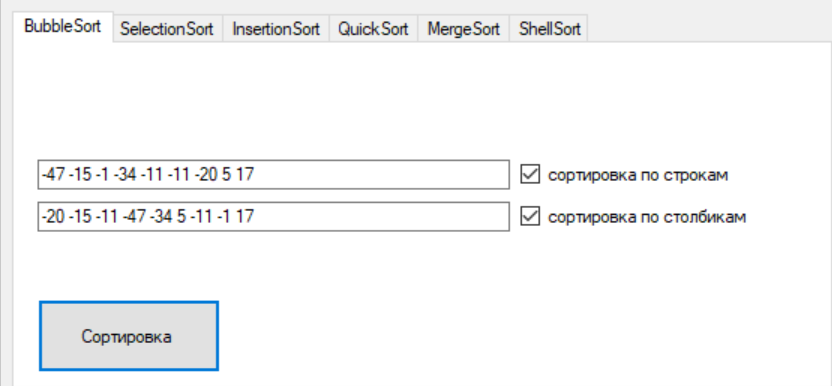
* Установите флажки напротив параметров сортировки, которые вы хотите применить на выбранной вкладке:
  + checkBox1 для пузырьковой сортировки строк.
  + checkBox2 для пузырьковой сортировки столбцов.
  + checkBox3 для сортировки строк методом выбора.
  + checkBox4 для сортировки столбцов методом выбора.
  + checkBox5 для сортировки строк методом вставок.
  + checkBox6 для сортировки столбцов методом вставок.
  + checkBox8 для быстрой сортировки строк.
  + checkBox7 для быстрой сортировки столбцов.
  + checkBox9 для сортировки строк методом слияния.
  + checkBox10 для сортировки столбцов методом слияния.
  + checkBox12 для сортировки строк методом Шелла.
  + checkBox11 для сортировки столбцов методом Шелла.
* Запуск сортировки:
  + Нажмите кнопку "Сортировка" на выбранной вкладке для запуска выбранного метода сортировки как на рисунке 2.5.

Рисунок 2.5 – Выбор сортировки и её результат

Просмотр результатов:

* После выполнения сортировки результаты будут отображены в соответствующих текстовых полях (textBox) на той же вкладке:
  + textBox3 и textBox4 для результатов пузырьковой сортировки.
  + textBox5 и textBox6 для результатов сортировки методом выбора.
  + textBox7 и textBox8 для результатов сортировки методом вставок.
  + textBox9 и textBox10 для результатов быстрой сортировки.
  + textBox11 и textBox12 для результатов сортировки методом слияния.
  + textBox13 и textBox14 для результатов сортировки методом Шелла.

Описание интерфейса и функций программы

Главное окно:

* Содержит таблицу (dataGridView1), в которую вводятся данные для сортировки.
* Вкладки для различных методов сортировки: "BubbleSort", "SelectionSort", "InsertionSort", "QuickSort", "MergeSort", "ShellSort".
* Кнопки для выполнения сортировки: "Сортировка".
* Флажки (checkBox) для выбора метода сортировки и области применения (строки или столбцы).
* Текстовые поля (textBox) для отображения результатов сортировки.

Элементы управления:

* Таблица (dataGridView1): используется для ввода данных, которые необходимо отсортировать.
* Флажки (checkBox): позволяют выбрать метод сортировки и область применения (строки или столбцы).
* Кнопки: запускают соответствующие методы сортировки:
  + button3 для пузырьковой сортировки.
  + button2 для сортировки методом выбора.
  + button4 для сортировки методом вставок.
  + button5 для быстрой сортировки.
  + button6 для сортировки методом слияния.
  + button7 для сортировки методом Шелла.
* Текстовые поля (textBox): отображают результаты выполнения сортировки.
  1. **Тестирование программного продукта**

Тестирование программного продукта представлена на таблице 2.1.

Таблица 2.1 – Тестирование программного продукта

|  |  |  |
| --- | --- | --- |
| №  п\п | Название тестового случая | Тестовый случай пройдён? |
| 1. | Отображение приложение | Да |
| 2. | Кнопка «Создать», «Сортировка» | Да |
| 3. | Отображение отсортированного массива | Да |
| 4. | Тестирование метода сортировки Bubble | Да |
| 5. | Тестирование метода сортировки Insertion | Да |
| 6. | Тестирование метода сортировки Merge | Да |
| 7. | Тестирование метода сортировки Quick | Да |
| 8. | Тестирование метода сортировки Shell | Да |
| 9. | Тестирование метода сортировки Selection | Да |

1. Модульное тестирование:

* Проверка отдельных функций сортировки (BubbleSort, SelectionSort, InsertionSort, QuickSort, MergeSort, ShellSort) на корректность работы с различными наборами данных.
* Тестирование вспомогательных функций, таких как сортировка строк и столбцов.

1. Интеграционное тестирование:

* Проверка взаимодействия между различными модулями программы.
* Тестирование корректности передачи данных из таблицы в алгоритмы сортировки и обратно.

1. Функциональное тестирование:

* Проверка всех функций пользовательского интерфейса, включая ввод данных, выбор методов сортировки и отображение результатов.
* Проверка работы всех кнопок, флажков и текстовых полей.

1. Тестирование производительности:

* Оценка времени выполнения сортировки для больших объемов данных.
* Проверка устойчивости программы при работе с большими таблицами.

Представление результатов тестирования

1. Модульное тестирование:

* Все основные функции сортировки прошли проверку на корректность. Алгоритмы сортировки правильно обрабатывали различные наборы данных, включая пустые массивы, отсортированные массивы, массивы с одинаковыми элементами и случайные массивы.

1. Интеграционное тестирование:

* Тестирование показало, что данные корректно передаются из таблицы в функции сортировки и обратно. Все методы сортировки успешно интегрированы с пользовательским интерфейсом.

1. Функциональное тестирование:

* Все элементы интерфейса (кнопки, флажки, текстовые поля) работают корректно. Пользователь может вводить данные, выбирать методы сортировки и получать правильные результаты.
* Программа успешно обрабатывает все тестовые сценарии, включая сортировку строк и столбцов различными методами.

1. Тестирование производительности:

* Программа демонстрирует приемлемую производительность при работе с таблицами до 1000 элементов. Время выполнения сортировки остается в разумных пределах для всех методов.

Анализ ошибок и способы их устранения

1. Ошибка: Проблемы с отображением больших массивов данных:

* Описание: При сортировке больших массивов данных результаты могли отображаться некорректно.
* Решение: Оптимизирован алгоритм отображения данных в таблице. Улучшена производительность интерфейса при работе с большими объемами данных.

1. Ошибка: Неправильное обновление интерфейса:

* Описание: В некоторых случаях интерфейс не обновлялся после сортировки.
* Решение: Обновлен механизм обновления интерфейса. Теперь результаты сортировки корректно отображаются после каждого выполнения алгоритма.

1. Ошибка: Нестабильная работа при больших объемах данных:

* Описание: При очень больших объемах данных программа могла зависать.
* Решение: Проведена оптимизация алгоритмов сортировки. Улучшена работа с памятью для предотвращения зависаний.

Тестирование программного продукта показало, что программа работает корректно и стабильно при различных сценариях использования. Проведенные исправления ошибок и оптимизации позволили улучшить производительность и надежность программы. [16]

# **ЗАКЛЮЧЕНИЕ**

В данном проекте была разработана и реализована программа для сортировки данных в таблице различными методами. Основной целью проекта было создание удобного и функционального инструмента, позволяющего пользователю сортировать данные по строкам и столбцам с использованием нескольких алгоритмов сортировки.

Достижения проекта:

1. Разработка пользовательского интерфейса:

* Создан удобный и интуитивно понятный интерфейс, позволяющий пользователю легко вводить данные, выбирать метод сортировки и просматривать результаты.

1. Реализация различных алгоритмов сортировки:

* В программе реализованы и протестированы шесть основных алгоритмов сортировки: пузырьковая сортировка, сортировка выбором, сортировка вставками, быстрая сортировка, сортировка слиянием и сортировка Шелла.

1. Обработка данных и отображение результатов:

* Программа успешно обрабатывает данные из таблицы, передает их в соответствующие алгоритмы сортировки и корректно отображает отсортированные данные.

1. Тестирование и отладка:

* Проведено всестороннее тестирование программы, включающее модульное, интеграционное, функциональное и тестирование производительности. Выявленные ошибки были исправлены, а программа оптимизирована для работы с большими объемами данных.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. 7 способов сортировки массивов на примере С++ с иллюстрациями, Дата доступа: 24.06.2024 [Электронный ресурс] - <https://proglib.io/p/7-sposobov-sortirovki-massivov-na-primere-s-s-illyustraciyami-2022-04-20>
2. Описание алгоритмов сортировки и сравнение их производительности , Дата доступа: 24.06.2024 [Электронный ресурс] - <https://habr.com/ru/articles/335920/>
3. C++: реализация алгоритмов сортировки, Дата доступа: 24.06.2024 [Электронный ресурс] - <https://spravochnick.ru/informatika/c_realizacii_algoritmov_sortirovki/>
4. Алгоритмы сортировки на С++, Дата доступа: 24.06.2024 [Электронный ресурс] - <https://www.matematicus.ru/cplusplus/algoritmy-sortirovki-na-c>
5. Руководство по классическим приложениям (Windows Forms .NET), Дата доступа: 24.06.2024 [Электронный ресурс] <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?v>
6. Генерация случайных чисел в диапазоне [a,b], Дата доступа: 24.06.2024 [Электронный ресурс] <https://ru.stackoverflow.com/questions/41317/Генерация-случайных-чисел-в-диапазоне-a-b>
7. Урок №71. Генерация случайных чисел, Дата доступа: 24.06.2024 [Электронный ресурс] <https://ravesli.com/urok-71-generatsiya-sluchajnyh-chisel-funktsii-srand-i-rand/>
8. Принципы пользовательского интерфейса, Дата доступа: 24.06.2024 [Электронный ресурс] <https://learn.microsoft.com/ru-ru/windows/win32/appuistart/-user-interface-principles>
9. Тестирование программного продукта, Дата доступа: 24.06.2024 [Электронный ресурс] <https://www.tadviser.ru/index.php/Статья:Тестирование_программного_продукта>
10. Виды тестирования программного обеспечения (100 примеров), Дата доступа: 24.06.2024 [Электронный ресурс] <https://www.guru99.com/ru/types-of-software-testing.html>
11. Алгоритмы сортировки: что, зачем и почему?, Дата доступа: 24.06.2024 [Электронный ресурс] <https://itproger.com/news/algoritmi-sortirovki-chto-zachem-i-pochemu>
12. [Электронный ресурс] <https://thecode.media/sort/>
13. Объектно-ориентированный анализ и проектирование (ООАД), Дата доступа: 24.06.2024 [Электронный ресурс] <https://www.geeksforgeeks.org/object-oriented-analysis-and-design/>
14. Документация по интегрированной среде разработки Visual Studio, Дата доступа: 24.06.2024 [Электронный ресурс] <https://learn.microsoft.com/ru-ru/visualstudio/ide/?view=vs-2022>
15. Software Testing Techniques with Test Case Design Examples, Дата доступа: 24.06.2024 [Электронный ресурс] https://www.guru99.com/software-testing-techniques.html

# **ПРИЛОЖЕНИЕ**

Код всех сортировок

private: void BubbleSort(array<int>^ arr) {

int n = arr->Length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

private: void SelectionSort(array<int>^ arr) {

int n = arr->Length;

for (int i = 0; i < n - 1; i++) {

int min\_idx = i;

for (int j = i + 1; j < n; j++) {

if (arr[j] < arr[min\_idx]) {

min\_idx = j;

}

}

int temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

private: void InsertionSort(array<int>^ arr) {

int n = arr->Length;

for (int i = 1; i < n; i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

private: void QuickSort(array<int>^ arr, int low, int high) {

if (low < high) {

int pi = Partition(arr, low, high);

QuickSort(arr, low, pi - 1);

QuickSort(arr, pi + 1, high);

}

}

private: int Partition(array<int>^ arr, int low, int high) {

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return (i + 1);

}

private: void MergeSort(array<int>^ arr, int l, int r) {

if (l < r) {

int m = l + (r - l) / 2;

MergeSort(arr, l, m);

MergeSort(arr, m + 1, r);

Merge(arr, l, m, r);

}

}

private: void Merge(array<int>^ arr, int l, int m, int r) {

int n1 = m - l + 1;

int n2 = r - m;

array<int>^ L = gcnew array<int>(n1);

array<int>^ R = gcnew array<int>(n2);

for (int i = 0; i < n1; i++) {

L[i] = arr[l + i];

}

for (int j = 0; j < n2; j++) {

R[j] = arr[m + 1 + j];

}

int i = 0, j = 0;

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

private: void ShellSort(array<int>^ arr, int size) {

int gap = size / 2;

while (gap > 0) {

for (int i = gap; i < size; i++) {

int temp = arr[i];

int j;

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

arr[j] = arr[j - gap];

}

arr[j] = temp;

}

gap /= 2;

}

}