Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

КОНСТРУКТОР ЭКЗАМЕНАЦИОННЫХ БИЛЕТОВ

Пояснительная записка

к курсовой работе по дисциплине

«Основы алгоритмизации и программирования»

**КП9К.939315.104.029ПЗ**

Руководитель В.А. Кабариха

Учащийся гр. А.М. Усов

310901

Минск 2024

**Содержание**

Введение 3

1 Требования к программе 4

2 Конструирование программы 10

2.1 Разработка модульной структуры программы 10

2.2 Выбор способа организации данных 12

2.3 Разработка перечня пользовательских функций программы 12

3. Разработка алгоритмов работы программы 14

3.1 Алгоритм функции main 14

3.2 Алгоритм функции сортировки () 15

3.3 Алгоритм функции поиска () 14

4 Описание работы программы 17

4.1 Авторизация 14

4.2 Модуль администратора 15

4.3 Модуль пользователя 14

4.4 Исключительные ситуации 14

Заключение 23

Список используемых источников 24

Приложение: Листинг кода с комментариями 25

5 Перечень графического материала 20

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

**КП9К.939315.104.029ПЗ**

Разраб.

Усов

Провер.

Василькова

Реценз.

Н. Контр.

Утверд.

Конструктор экзаменационных билетов

Пояснительная записка

Лит.

Листов

35

БГУИР

Введение

Курсовая работа «Конструктор экзаменационных билетов» предназначена для упрощенного и систематизированного создания экзаменационных билетов используя готовые шаблоны как самих билетов, так и вопросов к ним. А так же должна быть реализована возможность отправки готовых билетов на печать

При реализации курсовой работы «Конструктор экзаменационных билетов» были поставлены следующие задачи:

* изучить различные варианты вопросов;
* изучить аналоги для выявления нужного функционала
* создать систему для возможности прохождения тестов
* изучить возможности языка и среды разработки в сфере экспорта данных
* реализация понятного и простого интерфейса.

При разработке курсовой работы «Конструктор экзаменационных билетов» были проанализированы аналоги программных средств: «Экзаменатор», «Мой экзамен», «Билеты ПДД». На основе аналогов собирались функциональные возможности программного средства и создавалась курсовая работа «Конструктор экзаменационных билетов».

Курсовая работа «Конструктор экзаменационных билетов» ориентирована на учебные заведения и преподавателей, желающих автоматизировать процесс подготовки и проведения экзаменов, а также помочь с проведением самостоятельных работ через готовую систему с рейтингом проходивших тест учеников и упрочение оценивания их знаний.

Для работы над курсовой работой «Конструктор экзаменационных билетов» используется IDE (интегрированная среда разработки) Microsoft Visual Studio. Ее преимуществами являются ее популярность, следовательно, большое количество различной документации, также относительно удобная структура интерфейса, поддержка разных языков программирования, большое количество плагинов и дополнений, доступных для скачивания. Отдельно нужно отметить, что IDE позволяет удобно создать текстовый файл и начать работу с ним во время написания кода программы. В Microsoft Visual Studio заранее существует большое количество удобных и популярных библиотек, доступных для подключения в проекте.

С развитием технологий упрощается процесс подготовки и проведения экзаменов. Поэтому тема курсовой работы является актуальной и может быть использована в учебных заведениях для автоматизации подготовки экзаменационных билетов.

1 Постановка задачи

Курсовая работа посвящена созданию программного средства «Конструктор экзаменационных билетов». Целью курсового проекта является исследование различных методов создания экзаменационных билетов и упрощение процесса их подготовки и анализа.

Возможности и функции программы включают:

* Создание тестов по заготовленным шаблонам
* Создание билетов на основе созданного теста
* Прохождение ранее созданного теста / рандомного билета
* Просмотр рейтинга в виде таблицы для каждого теста
* Экспорт заранее созданных тестов на печать

Основное назначение программного средства «Конструктор экзаменационных билетов» - облегчение процесса подготовки экзаменов, улучшение качества подготовки билетов и их анализа, а также позволяет пользователям углубить свои знания в этой области.

1.1 Описание предметной области

Основная задача курсовой работы заключается в разработке программного средства «Конструктор экзаменационных билетов».

Предметная область – это сфера образования и подготовки к экзаменам, которая имеет существенное значение и непосредственное отношение к функционалу разрабатываемого программного средства.

Учебные компьютерные программы, такие как «Конструктор экзаменационных билетов», позволяют моделировать процесс создания и проверки билетов, являясь эффективным инструментом для обучения. В то же время, такая программа позволяет визуализировать абстрактные объекты и процессы, которые в реальном мире не существуют, а являются моделями реальных объектов и явлений в области образования.

Процесс представления предметной области задачи в виде набора объектов, обменивающихся сообщениями, называется объектной декомпозицией. В рамках этого процесса основные элементы задачи делятся на классы. Для каждого класса определяются его свойства и поведение, а также взаимодействие с другими классами.

В разрабатываемой системе главными объектами являются экзаменационные билеты и информация, содержащаяся в них. Пользователь запускает программу, выбирает предмет, тему и уровень сложности билетов, после чего программа генерирует экзаменационные билеты для дальнейшего анализа и использования.

1.2 Обзор существующих аналогов

Основная задача курсовой работы заключается в разработке программного средства "Конструктор экзаменационных билетов".

Предметная область – это сфера подготовки и проведения экзаменов, имеющая существенное значение для функционирования программы.

Учебные компьютерные программы, такие как "Конструктор экзаменационных билетов", позволяют моделировать и визуализировать процесс создания и анализа билетов. Это предоставляет возможность визуализировать и управлять абстрактными объектами и процессами, которые в реальном мире не существуют, но являются моделями реальных объектов и процессов в образовательной сфере.

При объектной декомпозиции процесс представления предметной области задачи преобразуется в набор объектов, обменивающихся сообщениями. В рамках этого процесса основные элементы задачи классифицируются, для каждого класса определяются его свойства и поведение, а также взаимодействие с другими классами.

В разрабатываемой системе главными объектами являются экзаменационные билеты и информация, содержащаяся в них. Пользователь запускает программу, выбирает тему и уровень сложности заданий, после чего программа генерирует экзаменационные билеты для дальнейшего анализа и использования.

Аналоги курсовой работы "Конструктор экзаменационных билетов" были изучены для формирования набора функций и особенностей работы, которые должны быть представлены в разрабатываемой программе. Некоторые из этих аналогов включают программы и ресурсы, такие как "Экзаменатор", "Мой экзамен", "Билеты ПДД" и другие. Изучение этих аналогов позволило идентифицировать их основные функции, преимущества и недостатки, что помогло в формировании требований к разрабатываемому программному средству.

1. "Экзаменатор":

Плюсы:

- Широкий выбор тем и предметов для создания экзаменационных билетов.

- Возможность настройки уровня сложности и количества вопросов в билете.

- Автоматическая генерация билетов на основе заданных параметров.

- Функция сохранения и печати созданных билетов.

Минусы:

- Ограниченные возможности редактирования и форматирования вопросов и ответов.

- Отсутствие функции импорта собственных вопросов и заданий.

- Нет возможности создания мультимедийных вопросов (с изображениями, аудио или видео).

2. "Мой экзамен":

Плюсы:

- Удобный интерфейс для создания и редактирования вопросов и билетов.

- Возможность добавления изображений и формул в вопросы и ответы.

- Функция случайного выбора вопросов из базы данных при генерации билетов.

- Возможность экспорта билетов в различные форматы (PDF, Word, Excel).

Минусы:

- Ограниченный выбор настроек для уровня сложности и количества вопросов в билете.

- Отсутствие функции автоматической оценки результатов экзамена.

- Нет возможности совместной работы нескольких пользователей над созданием билетов.

3. "Билеты ПДД":

Плюсы:

- Специализированное приложение для создания билетов по правилам дорожного движения.

- Большая база данных вопросов и ответов по ПДД.

- Возможность выбора региона и категории транспортного средства для генерации билетов.

- Функция симуляции экзамена с оценкой результатов.

Минусы:

- Узкая специализация (только билеты по ПДД), не подходит для создания билетов по другим темам.

- Ограниченные возможности редактирования и добавления собственных вопросов.

- Нет функции экспорта билетов в другие форматы.

Изучение этих аналогов позволило выявить ключевые функции и особенности, которые должны быть учтены при разработке вашей программы "Конструктор экзаменационных билетов":

1. Широкий выбор тем и предметов для создания билетов.

2. Гибкие настройки уровня сложности и количества вопросов в билете.

3. Удобный интерфейс для редактирования и форматирования вопросов и ответов.

4. Функция импорта собственных вопросов и заданий.

5. Автоматическая генерация билетов на основе заданных параметров.

6. Возможность экспорта билетов в различные форматы (PDF, Word, Excel).

7. Функция автоматической оценки результатов экзамена.

1.3 Функциональное назначение

Курсовой проект "Конструктор экзаменационных билетов" предназначен для автоматизации процесса создания и управления экзаменационными билетами. Программа должна предоставлять пользователям удобные инструменты для генерации билетов по различным темам и предметам, а также возможности для настройки уровня сложности и количества вопросов в каждом билете.

Компьютерная версия "Конструктора экзаменационных билетов" должна обеспечивать интуитивно понятный интерфейс и автоматизацию процессов создания, редактирования и анализа билетов. Приложение предназначено для оказания помощи преподавателям, экзаменаторам и другим специалистам в сфере образования, позволяя им эффективно разрабатывать и управлять экзаменационными материалами.

Основные задачи курсового проекта "Конструктор экзаменационных билетов":

- Предоставление удобных инструментов для создания и редактирования экзаменационных билетов по различным темам и предметам.

- Обеспечение гибких настроек уровня сложности и количества вопросов в каждом билете.

- Реализация функций автоматической генерации билетов на основе заданных параметров.

- Предоставление возможностей для анализа и оценки результатов экзаменов.

Основные функции и возможности программы:

* Создание и редактирование вопросов и ответов для экзаменационных билетов.
* Настройка уровня сложности и количества вопросов в каждом билете.
* Автоматическая генерация билетов на основе заданных параметров (тема, уровень сложности, количество вопросов и т.д.).
* Импорт и экспорт данных, позволяющие пользователям загружать собственные вопросы и сохранять билеты в различных форматах (PDF, Word, Excel).
* Функция автоматической оценки результатов экзамена на основе ответов пользователя.
* Генерация отчетов и статистики по результатам экзаменов.
* Удобный и интуитивно понятный пользовательский интерфейс.

Программа "Конструктор экзаменационных билетов" должна обеспечивать следующие функции:

- Информационная функция – предоставление пользователям доступа к базе данных вопросов и ответов по различным темам и предметам.

- Обучающая функция – помощь преподавателям в создании эффективных экзаменационных материалов, способствующих проверке знаний и навыков учащихся.

- Аналитическая функция – предоставление инструментов для анализа результатов экзаменов и оценки эффективности экзаменационных билетов.

Пользователи должны иметь возможность сохранять и загружать свои наборы билетов, а также экспортировать для дальнейшего использования. Программа должна обеспечивать стабильную работу и защиту данных пользователей.

# 

# 2 Проектирование задачи

## 

## 2.1 Алгоритм работы программы

Алгоритм – это набор инструкций, описывающих порядок действий исполнителя для достижения определенного результата. В контексте разработки программного обеспечения алгоритм представляет собой последовательность шагов, которые программа должна выполнить для решения поставленной задачи.

Требуется разработать курсовой проект "Конструктор экзаменационных билетов". Этот программный продукт предназначен для автоматизации процесса создания, редактирования и управления экзаменационными билетами по различным темам и предметам. Программное средство может быть использовано преподавателями, экзаменаторами и другими специалистами в сфере образования.

Алгоритм работы программы "Конструктор экзаменационных билетов" можно представить следующим образом:

1. Запуск самой программы и вывидение окна регистрации

2. В Окне регистрации пользователь выбирает 1 из трех пунктов: продолжить как гость , авторизоваться и зарегистрироваться

3. В зависимости от выбора продолжить программу на определенных правах доступа

4.Вобор создать тест, выбрать существующий для дольнейшей работы с ним, и для администратора: редактировать рейтинг, редактировать пользователей

5. Во время создания теста происходит выбор из готовых шаблонов вопросов , таких как вопрос с одиночным ответом, вопрос с несколькими вариантами ответов

6. После завершения проходдения теста польхователем , его результат записывается в базу рейтинга текущего теста.

7. Завершение работы программы при выборе соответствующего пункта.

Полный алгоритм программы представлен в приложении А

## 

## 2.2 Выбор и обоснование инструментов разработки

Для реализации курсового проекта "Конструктор экзаменационных билетов" была выбрана среда разработки Microsoft Visual Studio 2022. Данная IDE предоставляет широкий набор инструментов и функций, необходимых для создания консольных приложений.

Microsoft Visual Studio 2022 обладает рядом преимуществ, которые делают ее подходящей для разработки "Конструктора экзаменационных билетов":

1. Интуитивно понятный и удобный интерфейс, облегчающий процесс разработки.

2. Поддержка различных языков программирования, включая C++, C# и другие.

3. Встроенный отладчик, позволяющий эффективно находить и исправлять ошибки в коде.

4. Возможность подключения сторонних библиотек и компонентов для расширения функциональности.

5. Интеграция с системами контроля версий, такими как Git, для удобной совместной работы над проектом.

Для разработки "Конструктора экзаменационных билетов" был выбран язык программирования C++. C++ является мощным и эффективным языком программирования, который предоставляет широкие возможности для создания консольных приложений.

Преимущества использования C++ для данного проекта:

1. Высокая производительность и эффективность, позволяющие создавать быстрые и оптимизированные приложения.

2. Поддержка объектно-ориентированного программирования, позволяющая создавать модульный и легко расширяемый код.

3. Возможность низкоуровневого управления памятью и ресурсами, что дает полный контроль над работой приложения.

4. Наличие стандартной библиотеки шаблонов (STL), предоставляющей готовые решения для многих задач.

5. Широкое сообщество разработчиков и обширная документация, облегчающие поиск информации и решение проблем.

Несмотря на то, что "Конструктор экзаменационных билетов" будет разрабатываться как консольное приложение, Visual Studio 2022 предоставляет удобные инструменты для работы с текстовым вводом и выводом. Среда разработки позволяет легко организовать взаимодействие с пользователем через командную строку, обрабатывать ввод данных и отображать результаты.

Кроме того, Visual Studio 2022 предоставляет инструменты для работы с файлами и базами данных, что важно для реализации функций хранения и управления вопросами и билетами в "Конструкторе экзаменационных билетов". Среда разработки позволяет легко создавать и редактировать файлы данных, а также взаимодействовать с базами данных с помощью соответствующих библиотек и API.

Таким образом, выбор Microsoft Visual Studio 2022 и языка программирования C++ для разработки курсового проекта "Конструктор экзаменационных билетов" в виде консольного приложения обоснован их мощными возможностями, эффективностью и наличием необходимых инструментов для работы с текстовым вводом/выводом, файлами.

# 

# 3 Программная реализация

# 3.1 Описание алгоритмов решения задачи

Алгоритм – набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результа. В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Независимые инструкции могут выполняться в произвольном порядке, параллельно, если это позволяют используемые исполнители.

Программа начинаеися с ввода диапазона начального и конечного числовых значений. Далее пользователь выбирает любой из трёх предложенных алгоритмов генериррования чисел. После чего получает ряд сгенерированных чисел параллельно с их подробным анализом.

Алгоритм проверки чисел на чётность:

Ряд сгенерированных чисел записывается в массив, где каждое число это элемент массива. Далее идёт простая математическая проверка на чётность. Если число делится на 2 без остатка, оно является чётным. В противном случае число будет являться нечётным.

В ячейки будет выводиться количество чётных и нечётных чисел,а также будут выводиться сами эти числа.

Алгоритм нахождения максимального и минимального чисел:

Ряд сгенерированных чисел записывается в массив, где каждое число это элемент массива. Вводятся определённые переменные, отвечающие за минимальное и максимальное число. Будет производиться прохождение по всему массиву чисел. Если максимальный элемент меньше элемента массива, то оно присваивается этому лементу массива. Аналогично будет с поиском минимального элемента.

В ячейки выводятся сами минимальные и максимальные числа.

Алгоритм поиска простых и составных чисел:

Ряд сгенерированных чисел записывается в массив, где каждое число это элемент массива. Если данный элемент массива делится только на единицу и на само себя, то оно является простым. Во всех остальных случаях число будет составным.

В ячейки будет выводиться количество простых и составных чисел,а также будут выводиться сами эти числа.

Алгоритм поиска положительных и отрицательных чисел:

Ряд сгенерированных чисел записывается в массив, где каждое число это элемент массива. Если элемент массива больше или равен нулю, он будет являиться положительным. В противном случае, число будет отрицательным.

В ячейки будет выводиться количество положительных и отрицательных чисел,а также будут выводиться сами эти числа.

Алгоритм поиска повторяющихся чисел:

Ряд сгенерированных чисел записывается в массив, где каждое число это элемент массива. Далее идёт прохождение по уже заполненным элементам массива, где сравнивается каждый с введенным значением.

Таким образом, если находится совпадение, то в ячейку выводятся повторяющиеся числа.

Алгоритм проверки однозначных и двузначных чисел:

Ряд сгенерированных чисел записывается в массив, где каждое число это элемент массива. Если элемент попадает в диапазон однозначных чисел, а это от -9 до 9 включительно, то он является однозначным. Если же элемент попадает в диапазон двузначных чисел от -99 до 99 включительно, но при этом не включая диапазон однозначных чисел, то он является двузначным.

В ячейки будут выводиться однозначные и двузначные числа.

# 3.2 Проектирование интерфейса приложения

Рассмотрим сам термин интерфейс. Интерфейс – это совокупность средств и правил, обеспечивающих взаимодействие устройств вычислительной системы и программ, а также взаимодействие с человеком. Основные задачи, для решения которых он предназначен:

* ввод и отображение информации (звук, изображение);
* управление отдельными приложениями;
* обмен данными с другими устройствами;
* взаимодействие с операционной системой.

Пользовательский интерфейс объединяет в себе все элементы и компоненты программы, которые способны оказать влияние на взаимодействие пользователя с программным обеспечением.

Интерфейс должен быть простым и интуитивно понятным рядовому пользователю. Все надписи должны быть чётко видны, элементы не должны перекрывать друг друга.

Приложение было решено разрабатывать в ярких и светлых тонах. Макет интерфейса главного окна представлен ниже (см. рисунок 3.2).

На форме располагаются следующие функциональные элементы:

* TextBox;
* Label;
* Button.

textBox4

textBox5

textBox31

textBox30

textBox33

textBox32

label26

label25

label22

textBox8

button1

textBox20

textBox21

textBox1

button4

textBox25

label4

label15

label9

label7

textBox16

textBox6

textBox6

textBox10

textBox34

textBox11

textBox35

textBox36

textBox37

textBox22

textBox38

textBox9

textBox24

textBox27

textBox26

textBox9

textBox39

textBox22

label49

label8

label8

textBox18

textBox17

textBox43

textBox46

textBox19

textBox40

textBox47

textBox48

textBox41

textBox42

textBox44

textBox45

Рисунок 3.2 – Макет главного окна программы

# 4 Тестирование

Тестирование – процесс проверки программного обеспечения с целью выявления и предотвращения дефектов, оценки соответствия требованиям заказчика.

Баг (дефект) – ошибка в коде, которая может привести к тому, что программа будет работать не так, как ожидается.

В широком смысле, тестирование – это одна из техник контроля качества (Quality Control), которая включает планирование, составление тестов, непосредственно выполнение тестирования и анализ полученных результатов.

Некоторые ошибки проявляются после первого же запуска программы на выполнение. Они заметны «невооруженным глазом», и для их обнаружения не надо прибегать ни к каким специальным средствам. Некоторые ошибки проявляются в случайные моменты работы программы. С такими ошибками справиться труднее всего. Если вы не можете зафиксировать условия возникновения ошибки, вы не сможете понять причину и устранить ее. Иногда, чтобы добиться устойчивого проявления ошибки, перед каждым стартом программы приходится перезагружать среду разработки.

Всевозможные ошибки обычно подразделяют на три группы:

– синтаксические ошибки;

– ошибки времени выполнения программы;

– смысловые (логические) ошибки;

– рассмотрим каждую из выделенных групп по порядку.

Синтаксические ошибки являются самыми простыми, они устраняются уже на этапе компиляции. Причина их проста и однообразна – неправильная запись служебных слов, операторов.

Намного больше неприятностей доставляют ошибки времени выполнения. Они дают о себе знать прекращением выполнения программы. Чаще всего ошибка времени выполнения является признаком смысловой ошибки.

Смысловые (логические) ошибки – самые сложные и трудноуловимые. Они проявляются в том, что программа выполняет не то, что от нее предполагалось. Последствия смысловых ошибок могут быть самыми разными: это может быть что-то безобидное, например, неправильное содержимое окна, невыполнение или неверное выполнение команд пользователя, неправильное содержимое выходной информации и прочее, а может быть и достаточно серьезными – программа может досрочно завершиться с ошибкой времени выполнения и многое другое.

При исправлении ошибки самое главное – это не внести в программу новых ошибок.

Тестирование программного обеспечения имеет следующие уровни:

* компонентное – поиск дефектов и проверка функциональности (модулей, программ, объектов, классов), которые можно протестировать изолированно;
* интеграционное – проверяет взаимодействие между компонентами, взаимодействие различных частей системы (ОС, файловая система, аппаратное обеспечение);
* системное – тестирование поведения тестового объекта как целостной системы или продукта;
* приемочное, в свою очередь, подразделяется на пользовательское, эксплуатационное, контрактное и правовое, альфа и бета. Заключается в проверке работоспособности системы, частей системы или отдельных нефункциональных характеристик системы.

Также различают следующие типы тестирования:

* функциональное – проверяет, что система делает. Проводится тестирование внешнего поведения ПО (black–box testing);
* нефункциональное – проверяет, как система работает. Сюда входят нагрузочное тестирование, тестирование производительности, стресс–тестирование, тестирование удобства использования (в большинстве случаев black–box testing);
* структурное – тестирование, основанное на анализе внутренней структуры компонента или системы (white–box testing).

Тестирование изменений включает в себя подтверждение и регрессионное тестирование. Подтверждение – проверка того, что дефект действительно исправлен. Регрессионное тестирование – проверка, что ранее работающая корректно функциональность все еще работает.

После того, как ошибка зафиксирована, необходимо найти в исходном тексте программы то место, в котором она возникает. Другими словами, нужно локализировать ошибку. Первое, что приходит в голову, – протестировать программу по шагам с помощью отладчика, выполнить трассировку. Однако сделать это далеко не всегда возможно, особенно в больших и сложных программах. При исправлении ошибки самое главное – не внести в программу новых ошибок.

Тестирование программы проходило в ходе и после ее разработки.

Отладка программы осуществлялась встроенными средствами Microsoft Visual Studio 2019.

По результатам тестирования были выявлены некоторые ошибки в работе программы. В дальнейшем они были исправлены и все имеющиеся функции протестированы снова.

Таблица 4.1 – Тестирование программного средства

|  |  |  |
| --- | --- | --- |
| Тестируемая операция | Требуемый  результат | Полученный  результат |
| Запуск программы | Появление главного рабочего окна программы со всеми его компонентами | После нажатия на значок программы (открытия приложения) появилось окно программы со всеми компонентами |
| Ввод неверных (не числовых) символов в поля для диапазона числовых значений | Сообщение об ошибке, остутствие неверно введенных символов | Программа вылетает, появление сведений об ошибке |
| Нажатие на одну из кнопок генерирования чисел и их анализа | Появление сгенерированных чисел параллельно с заполненными полями для анализа случайно сгенерированных чисел | Появление в поле сгеенрированного множества чисел, а также появление в ячейках анализа данных чисел |
| Нажатие на одну из кнопок генерирования и анализа чисел с незаполненными полями для диапазона чисел | Появление надписи об ошибке в окнах результата | Программа вылетает, появление сведений об оишбке |
| Нажатие на кнопку «Сброс» | Очистка всех полей | Очистка всех полей |
| Нажатие на главное меню «Меню» | Появление возможных дополнительных функций для программы | Появление возможных дополнительных функций для программы |
| Нажатие на кнопку «Подробнее об анализе» | Появление нового окна с инструкцией к анализу множества чисел для дополнительного изучения информации | Повяление нового окна с инстукцией к анализу множества чисел. |
| Завершение программы | Закрытие формы программы | Программа завершила свою работу |

# 5 Руководство пользователя

# Для запуска программы необходимо дважды нажать на файл с названием «Kurs.exe». После этого на экране появится главное окно (см. рисунок 5.1).

# 

Рисунок 5.1 – Главное окно программы «Kurs»

Далее предлагается ввести два n-разрядных числа в поля. Это и будет диапазон для последующего генерирования чисел (см. рисунок 5.2).

# 

Рисунок 5.2 – Поля для ввода диапазона чисел

При попытке ввести некорректный символ, программа будет вылетать и сообщать об ошибке (см. рисунок 5.3).

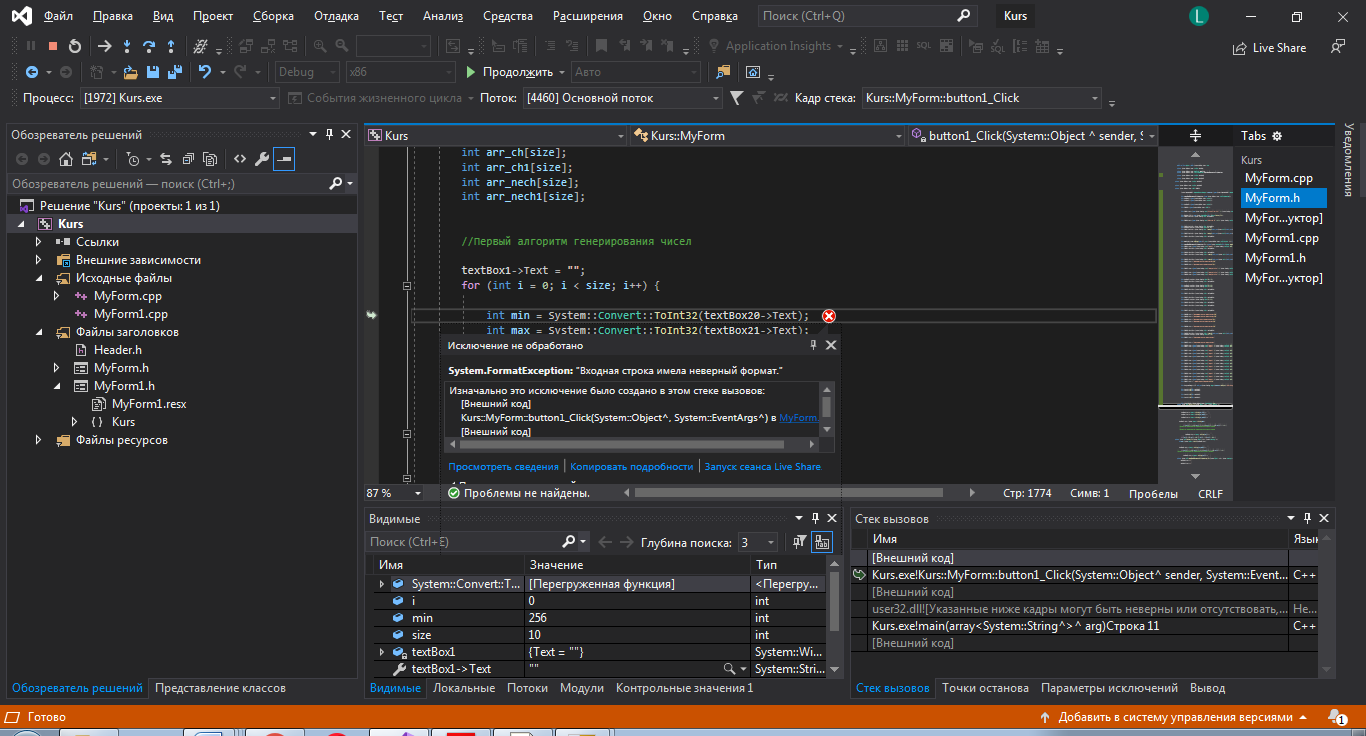


Рисунок 5.3 – Сведения об ошибке при некорректном вводе

Следующим шагом пользователь может выбрать любой из предложенных алгоритмов генерирования чисел и получить ряд данного множества чисел в ячейках (см. рисунок 5.4).

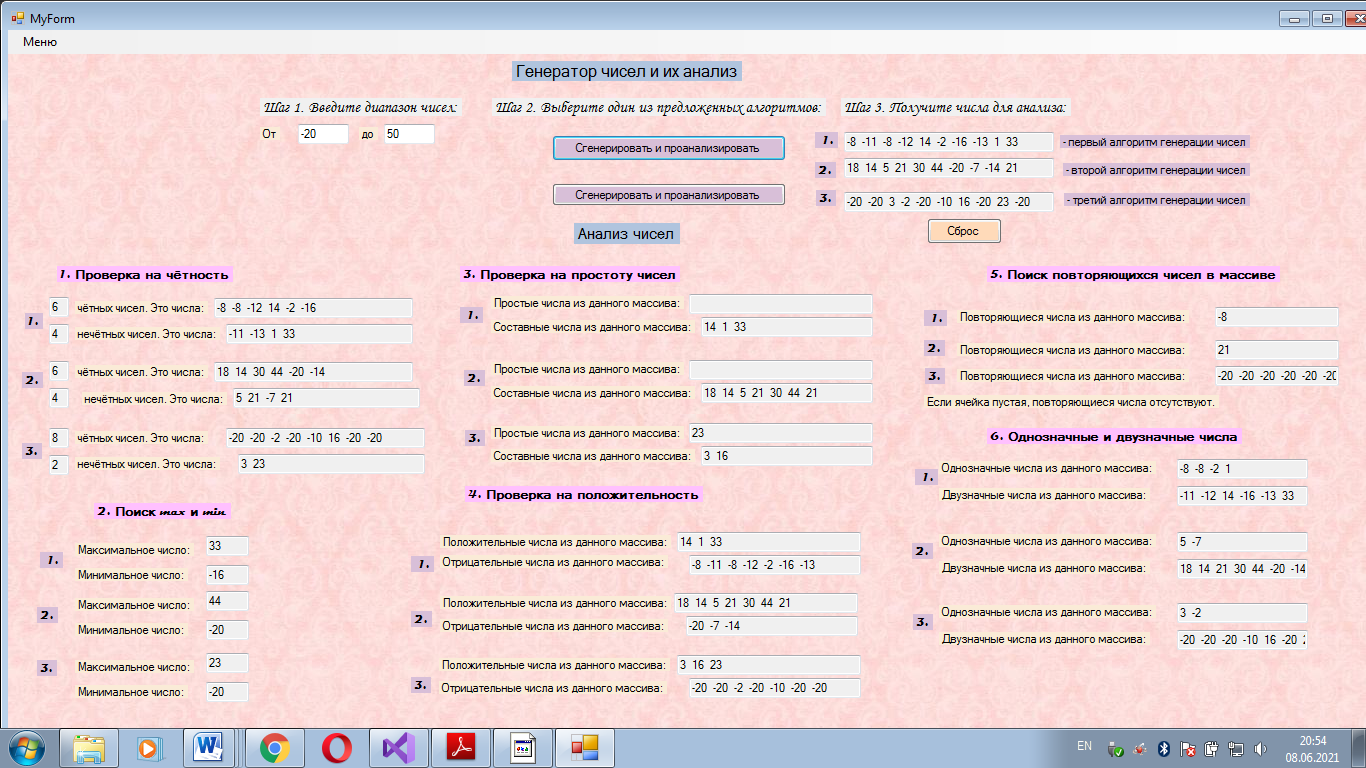


Рисунок 5.4 – Получения ряда чисел различными алгоритмами генерации

Параллельно с генерированием чисел, пользователь получает подробный анализ ряда чисел. Анализ включает в себя проверки чисел на чётность, поиск максимальных и минимальных чисел из данного множества, нахождение простых и составных чисел, определение положительных и отрицательных чисел, поиск повторяющихся чисел в массиве, а также возможность нахождения однозначных и двузначных чисел (см. рисунок 5.5).

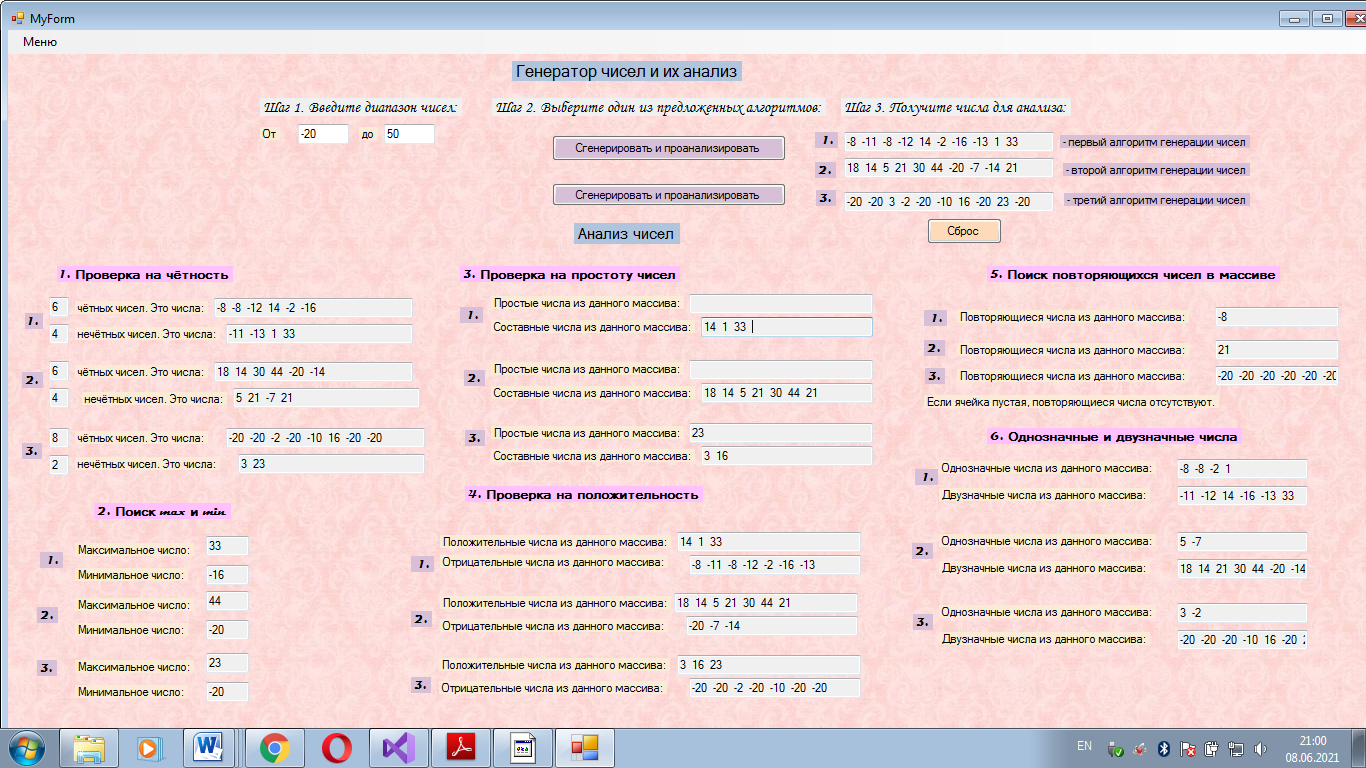


Рисунок 5.5 – Интерфейс анализа сгенерированного множества чисел

При желании пользователь может открыть инструкцию «Подробнее об анализе», которая располагается в главном «Меню» (см. рисунок 5.6).

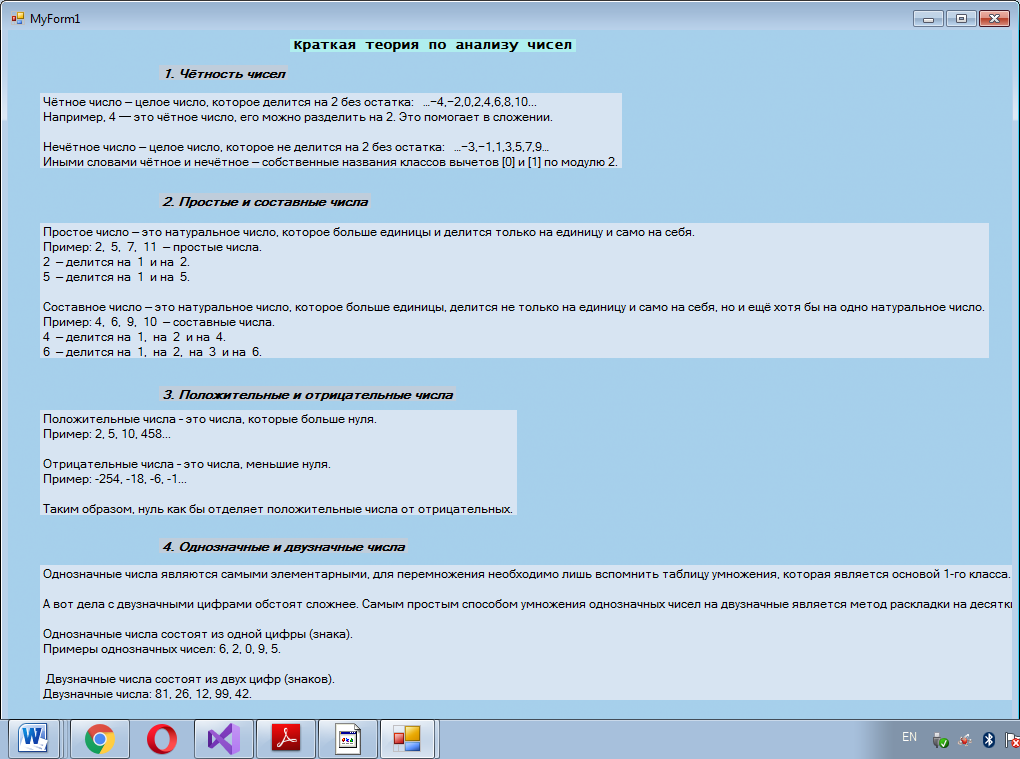


Рисунок 5.6 – Инструкция к анализу чисел

Для очистки всех полей необходимо нажать на кнопку «Сброс».

Для закрытия программы необходимо нажать на крест в правом верхнем углу или открыть «Меню» и нажать на «Закрыть».

# Заключение

В ходе выполнения курсового проекта была достигнута цель проектирования – разработан курсовой проект «Генерирование случайных чисел и их анализ» с графической реализацией. Все компоненты в ходе реализации были обёрнуты в класс языка C++ для более удобной работы над структурой.

Затем, с помощью Windows Forms, была реализована графическая оболочка для визуализации структуры данных и более удобного ввода значений в поля, предназначенные для этого. После нажатий кнопок будут происходить все необходимые действия, предусмотренные программой.

В пояснительной записке выполнены анализ задачи, проектирование и реализация программных средств, которые являются необходимыми этапами разработки программного обеспечения.

Курсовой проект «Генерирование случайных чисел и их анализ» помогает закрепить полученные знания на протяжения всего курса обучения, улучшить навыки решения поставленных задач, дает возможность научиться поиску необходимой информации для облегчения решения поставленной задачи.

Важным достижением при разработке курсового проекта «Генерирование случайных чисел и их анализ» стала успешная работа всех запланированных функций, особенно удобность интерфейсов дополнительных окон. Вместе с курсовым проектом «Генерирование случайных чисел и их анализ» идет встроенная в программу подробная инструкция, описывающая назначение и принципы использования соответствующих функций.

Таким образом, в результате разработки курсового проекта «Генерирование случайных чисел и их анализ» были достигнуты все поставленные цели и реализованы предусмотренные задачи. Удалось создать программу с удобным графическим интерфейсом, сочетающимся со всеми ее возможностями. Для удобства работы использовались автоматизированные и стабильно работающие функции. Также в ходе разработки было изучено много новой информации, компонентов и свойств Visual Studio 2019 и языка программирования C++, использовавшихся при работе над курсовым проектом «Генерирование случайных чисел и их анализ».

# Список использованных источников

Литература

1. Албахари, Джозеф C++ 3.0. Справочник / Джозеф Албахари , Бен Албахари. М.: БХВПетербург, 2012. 944 c.

2. Медведьев В.И. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java. / В.И. Медведьев – 2-е изд. – Казань: РИЦ «Школа», 2010. -444 c.

3. Лафоре Роберт Объектно-ориентированное программирование в C++ / Роберт Латофе. – Питер: Санкт-Петербург. 2019. – 928 с.

4. Доусон Майкл Изучаем C++ через программирование игр / Майкл Доунсон. – Питер: Санкт-Петербург. 2020. – 352 с.

5. Вагнер, Билл С++Эффективное программирование / Билл Вагнер. М.: ЛОРИ, 2013. 320 c.

Интернет-ресурсы

6. Интернет ресурс DrawIo [электронный ресурс] Режим доступа: <https://app.diagrams.net/#G1pmLYmOM6D9bOF9xbtPqSsZu72U3mEjAk>. – 29.05.2021

7. Интернет ресурс Argon online [электронный ресурс] Режим доступа: <http://argon.pro/windows/other/ssavers>. – 17.04.2021

8. Интернет ресурс SMMplanner [электронный ресурс] Режим доступа: <https://smmplanner.com/blog/top-randomaizierov-dlia-konkursov-v-sotssietiakh>. – 19.05.2021

# **ПРИЛОЖЕНИЕ А**

(обязательное)

**Код программы**

**MyForm.cpp:**

#include "MyForm.h"

#include "string"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(cli::array<String^>^ arg) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Kurs::MyForm form; //WinFormsTest - имя вашего проекта

Application::Run(% form);

}

**MyForm.h:**

#pragma endregion

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

const int size = 10;

int arr[size];

int arr1[size];

int arr\_ch[size];

int arr\_ch1[size];

int arr\_nech[size];

int arr\_nech1[size];

//Первый алгоритм генерирования чисел

textBox1->Text = "";

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

int min = System::Convert::ToInt32(textBox20->Text);

int max = System::Convert::ToInt32(textBox21->Text);

static unsigned seed = 1; // зерно не должно быть 0

seed = (seed \* 73129 + 95121) % 100000;

int schet = seed % 123;

int y = min;

while (schet > 0) {

schet--;

y++;

if (y == max) {

y = min;

}

}

textBox1->Text += Convert::ToString(y) + " ";

arr1[i] = y;

}

//Второй алгоритм генерирования чисел

textBox8->Text = "";

Random^ rnd = gcnew Random();

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

int min = System::Convert::ToInt32(textBox20->Text);

int max = System::Convert::ToInt32(textBox21->Text);

int x = rnd->Next(min, max);

textBox8->Text += Convert::ToString(x) + " ";

arr[i] = x;

}

//Проверка на чётность для первого алгоритма

int sch = 0;

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

if (arr[i] % 2 == 0) {

arr\_ch[sch] = arr[i];

sch++;

}

}

textBox2->Text += Convert::ToString(sch);

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

textBox6->Text += Convert::ToString(arr\_ch[i]) + " ";

}

sch = 0;

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

if (arr[i] % 2 != 0) {

arr\_nech[sch] = arr[i];

sch++;

}

}

textBox3->Text += Convert::ToString(sch);

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

textBox7->Text += Convert::ToString(arr\_nech[i]) + " ";

}

//Проверка на чётность для второго алгоритма

int tsch = 0;

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

if (arr1[i] % 2 == 0) {

arr\_ch1[tsch] = arr1[i];

tsch++;

}

}

textBox24->Text += Convert::ToString(tsch);

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

textBox22->Text += Convert::ToString(arr\_ch1[i]) + " ";

}

tsch = 0;

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

if (arr1[i] % 2 != 0) {

arr\_nech1[tsch] = arr1[i];

tsch++;

}

}

textBox23->Text += Convert::ToString(tsch);

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

textBox9->Text += Convert::ToString(arr\_nech1[i]) + " ";

}

//Поиск максимум и минимум для первого алгоритма

int min = arr[0];

int max = arr[0];

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

if (max < arr[i]) {

max = arr[i];

}

}

textBox4->Text = System::Convert::ToString(max);

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

if (min > arr[i]) {

min = arr[i];

}

}

textBox5->Text = System::Convert::ToString(min);

//Поиск максимум и минимум для второго алгоритма

int mini = arr1[0];

int maxi = arr1[0];

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

if (maxi < arr1[i]) {

maxi = arr1[i];

}

}

textBox31->Text = System::Convert::ToString(maxi);

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

if (mini > arr1[i]) {

mini = arr1[i];

}

}

textBox30->Text = System::Convert::ToString(mini);

//Проверка на простоту для первого алгоритма

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

bool check = true;

if (arr[j] > 0) {

for (int i = 2;i < arr[j];i++) {

if (arr[j] % i == 0) {

check = false;

}

}

if (check) {

textBox10->Text += Convert::ToString(arr[j]) + " ";

}

else textBox11->Text += Convert::ToString(arr[j]) + " ";

}

}

//Проверка на простоту для второго алгоритма

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

bool check = true;

if (arr1[j] > 0) {

for (int i = 2;i < arr1[j];i++) {

if (arr1[j] % i == 0) {

check = false;

}

}

if (check) {

textBox34->Text += Convert::ToString(arr1[j]) + " ";

}

else textBox35->Text += Convert::ToString(arr1[j]) + " ";

}

}

//Проверка на положительное или отрицательное число первого алгоритма

int sch1 = 0;

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

if (arr[i] >= 0) {

arr\_ch[sch1] = arr[i];

sch1++;

}

}

textBox12->Text += Convert::ToString(sch1);

for (int i = 0; i < sch1; i++) {

textBox48->Text += Convert::ToString(arr\_ch[i]) + " ";

}

sch1 = 0;

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

if (arr[i] < 0) {

arr\_nech[sch1] = arr[i];

sch1++;

}

}

textBox14->Text += Convert::ToString(sch1);

for (int i = 0; i < sch1; i++) {

textBox47->Text += Convert::ToString(arr\_nech[i]) + " ";

}

//Проверка на положительное или отрицательное число второго алгоритма

int tsch1 = 0;

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

if (arr1[i] >= 0) {

arr\_ch1[tsch1] = arr1[i];

tsch1++;

}

}

textBox12->Text += Convert::ToString(tsch1);

for (int i = 0; i < tsch1; i++) {

textBox40->Text += Convert::ToString(arr\_ch1[i]) + " ";

}

tsch1 = 0;

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

if (arr1[i] < 0) {

arr\_nech1[tsch1] = arr1[i];

tsch1++;

}

}

textBox14->Text += Convert::ToString(tsch1);

for (int i = 0; i < tsch1; i++) {

textBox19->Text += Convert::ToString(arr\_nech1[i]) + " ";

}

//Проверка на повторение чисел первого алгоритма

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

{

for (int j = i + 1;j < size; j++)

{

if (arr[i] == arr[j])

textBox16->Text += Convert::ToString(arr[i]) + " ";

}

}

//Проверка на повторение чисел второго алгоритма

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

{

for (int j = i + 1;j < size; j++)

{

if (arr1[i] == arr1[j])

textBox38->Text += Convert::ToString(arr1[i]) + " ";

}

}

//Проверка чисел на количество знаков первого алгоритма

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

if (arr[i] > -10 && arr[i] < 10) {

textBox17->Text += Convert::ToString(arr[i]) + " ";

}

}

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

if ((arr[i]<-9 && arr[i] >-100) || (arr[i] > 9 && arr[i] < 100)) {

textBox18->Text += Convert::ToString(arr[i]) + " ";

}

}

//Проверка чисел на количество знаков второго алгоритма

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

if (arr1[i] > -10 && arr1[i] < 10) {

textBox42->Text += Convert::ToString(arr1[i]) + " ";

}

}

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

if ((arr1[i]<-9 && arr1[i] >-100) || (arr1[i] > 9 && arr1[i] < 100)) {

textBox41->Text += Convert::ToString(arr1[i]) + " ";

}

}

}

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

textBox1->Text = "";

const int size = 10;

int arr2[size];

int arr\_ch2[size];

int arr\_nech2[size];

// Третий алгоритм генерирования чисел

srand(time(0));

mt19937 engine;

engine.seed(time(nullptr));

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

int min = System::Convert::ToInt32(textBox20->Text);

int max = System::Convert::ToInt32(textBox21->Text);

int schet = engine();

int z = min;

while (schet > 0) {

schet--;

z++;

if (z == max) {

z = min;

}

}

textBox25->Text += Convert::ToString(z) + " ";

arr2[i] = z;

}

//Проверка на чётность для третьего алгоритма

int trsch = 0;

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

if (arr2[i] % 2 == 0) {

arr\_ch2[trsch] = arr2[i];

trsch++;

}

}

textBox29->Text += Convert::ToString(trsch);

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

textBox27->Text += Convert::ToString(arr\_ch2[i]) + " ";

}

trsch = 0;

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

if (arr2[i] % 2 != 0) {

arr\_nech2[trsch] = arr2[i];

trsch++;

}

}

textBox28->Text += Convert::ToString(trsch);

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

textBox26->Text += Convert::ToString(arr\_nech2[i]) + " ";

}

//Поиск максимум и минимум для третьего алгоритма

int minim = arr2[0];

int maxim = arr2[0];

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

if (maxim < arr2[i]) {

maxim = arr2[i];

}

}

textBox33->Text = System::Convert::ToString(maxim);

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

if (minim > arr2[i]) {

minim = arr2[i];

}

}

textBox32->Text = System::Convert::ToString(minim);

//Проверка на простоту для третьего алгоритма

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

bool check = true;

if (arr2[j] > 0) {

for (int i = 2;i < arr2[j];i++) {

if (arr2[j] % i == 0) {

check = false;

}

}

if (check) {

textBox36->Text += Convert::ToString(arr2[j]) + " ";

}

else textBox37->Text += Convert::ToString(arr2[j]) + " ";

}

}

//Проверка на положительное или отрицательное число третьего алгоритма

int trsch2 = 0;

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

if (arr2[i] >= 0) {

arr\_ch2[trsch2] = arr2[i];

trsch2++;

}

}

textBox12->Text += Convert::ToString(trsch2);

for (int i = 0; i < trsch2; i++) {

textBox46->Text += Convert::ToString(arr\_ch2[i]) + " ";

}

trsch2 = 0;

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

if (arr2[i] < 0) {

arr\_nech2[trsch2] = arr2[i];

trsch2++;

}

}

textBox14->Text += Convert::ToString(trsch2);

for (int i = 0; i < trsch2; i++) {

textBox43->Text += Convert::ToString(arr\_nech2[i]) + " ";

}

//Проверка на повторение чисел третьего алгоритма

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

{

for (int j = i + 1;j < size; j++)

{

if (arr2[i] == arr2[j])

textBox39->Text += Convert::ToString(arr2[i]) + " ";

}

}

//Проверка чисел на количество знаков третьего алгоритма

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

if (arr2[i] > -10 && arr2[i] < 10) {

textBox45->Text += Convert::ToString(arr2[i]) + " ";

}

}

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

if ((arr2[i]<-9 && arr2[i] >-100) || (arr2[i] > 9 && arr2[i] < 100)) {

textBox44->Text += Convert::ToString(arr2[i]) + " ";

}

}

}

private: System::Void закрытьToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Close();

}

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

MyForm1^ M1 = gcnew MyForm1(); //создаём новый экземпляр

M1->Show();//открываем вторую форму

}

private: System::Void button3\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

textBox1->Text = L"";

textBox2->Text = L"";

textBox3->Text = L"";

textBox4->Text = L"";

textBox5->Text = L"";

textBox6->Text = L"";

textBox7->Text = L"";

textBox8->Text = L"";

textBox9->Text = L"";

textBox10->Text = L"";

textBox11->Text = L"";

textBox12->Text = L"";

textBox13->Text = L"";

textBox14->Text = L"";

textBox15->Text = L"";

textBox16->Text = L"";

textBox17->Text = L"";

textBox18->Text = L"";

textBox19->Text = L"";

textBox22->Text = L"";

textBox23->Text = L"";

textBox24->Text = L"";

textBox25->Text = L"";

textBox26->Text = L"";

textBox27->Text = L"";

textBox28->Text = L"";

textBox29->Text = L"";

textBox30->Text = L"";

textBox31->Text = L"";

textBox32->Text = L"";

textBox33->Text = L"";

textBox34->Text = L"";

textBox35->Text = L"";

textBox36->Text = L"";

textBox37->Text = L"";

textBox38->Text = L"";

textBox39->Text = L"";

textBox40->Text = L"";

textBox41->Text = L"";

textBox42->Text = L"";

textBox43->Text = L"";

textBox44->Text = L"";

textBox45->Text = L"";

textBox46->Text = L"";

textBox47->Text = L"";

textBox48->Text = L"";

}

};

}