Содержание

[Введение 5](#_Toc137071782)

[Глава 1. Исследование 6](#_Toc137071783)

[1.1. Исследование и анализ существующих программных решений 6](#_Toc137071784)

[1.1.1. Crossword compiler 11 7](#_Toc137071785)

[1.1.2. My Crossword Maker 8](#_Toc137071786)

[1.1.3. Crossword puzzle games 8](#_Toc137071787)

[1.1.4. EclipseCrossword 9](#_Toc137071788)

[1.1.5. Сравнительный анализ 10](#_Toc137071789)

[1.2. Инструментальный аппарат 11](#_Toc137071790)

[1.3. Постановка задач 12](#_Toc137071791)

[Глава 2. Моделирование 13](#_Toc137071792)

[2.1. Функциональная схема генератора кроссвордов 13](#_Toc137071793)

[2.2. Описание модулей и алгоритмов генератора кроссвордов 14](#_Toc137071794)

[2.2.1. Модуль меню 15](#_Toc137071795)

[2.2.2. Модуль авто-создания кроссворда 16](#_Toc137071796)

[2.2.3. Модуль пользовательского создания кроссворда 17](#_Toc137071797)

[2.2.4. Модуль решения кроссворда 18](#_Toc137071798)

[2.2.5. Модуль подведения итогов 19](#_Toc137071799)

[Выводы 19](#_Toc137071800)

[Глава 3. Разработка 21](#_Toc137071801)

[3.1. Окно с меню 21](#_Toc137071802)

[3.2. Окно выбора создания кроссворда 22](#_Toc137071803)

[3.3. Окно пользовательского создания 25](#_Toc137071804)

[3.4. Модуль авто генерации 27](#_Toc137071805)

[3.5. Окно решения кроссворда 28](#_Toc137071806)

[3.6. Окно подведения итогов 30](#_Toc137071807)

[Выводы 31](#_Toc137071808)

[Заключение 32](#_Toc137071809)

[Список использованной литературы 34](#_Toc137071810)

[Приложение А. Листинг кода 35](#_Toc137071811)

[Приложение А.1. Программный код формы MainMenu 35](#_Toc137071812)

[Приложение А.2. Программный код формы instructs 38](#_Toc137071813)

[Приложение А.3 Программный код формы SubForm 38](#_Toc137071814)

[Приложение А.4. Программный код формы UserCreate 39](#_Toc137071815)

[Приложение А.5. Программный код формы Crosshelp 46](#_Toc137071816)

[Приложение А.6. Программный код формы Deftable 48](#_Toc137071817)

[Приложение А.7. Программный код формы AutoCreate 49](#_Toc137071818)

[Приложение А.8. Программный код формы Results 59](#_Toc137071819)

[Приложение A.9. Программный код файла Programs.cs 60](#_Toc137071820)

[Приложение Б. Интерфейс приложения 61](#_Toc137071821)

[Приложение Б.1. Окно меню 61](#_Toc137071822)

[Приложение Б.2. Инструкции 62](#_Toc137071823)

[Приложение Б.3. Меню выбора режима 63](#_Toc137071824)

[Приложение Б.4. Окно SubForm 64](#_Toc137071825)

[Приложение Б.5. Окно пользовательского создания 65](#_Toc137071826)

[Приложение Б.6. Окно DefTable 66](#_Toc137071827)

[Приложение Б.7. Окно решения кроссворда 67](#_Toc137071828)

[Приложение Б.8. Окно итогов 68](#_Toc137071829)

# Введение

Кроссворды - головоломка, представляющая из себя переплетение рядов клеток, заполняемых словами по заданным значениям. Они тренируют память, расширяют словарный запас, повышают концентрацию. Кроссворды полезны как для развлечения, так и для обучения.

Существуют различные виды кроссворды: линейные кроссворды, филворды, кейворды – иногда появляются кроссворды основой которых не являются слова, например японские кроссворды, в которых надо расшифровать картинку, однако в данной работе будет рассмотрен классический кроссворд.

Цель данной работы - разработать программу для генерации и игры в кроссворды. Результатом работы должно быть приложение, которое может генерировать кроссворды различного размера, а также предоставлять возможность играть в них. Пользователь должен иметь возможность выбирать тематику кроссворда, его размер, а также проверять свои ответы.

Для достижения цели были поставлены следующие задачи:

1. исследование и анализ существующих программных решений;
2. моделирование системы;
3. разработка системы.

Результаты данной работы могут быть полезны как для обычных пользователей, которые любят играть в кроссворды, так и для образовательных учреждений, которые могут использовать разработанную программу для обучения языковым навыкам и развития мышления у своих студентов.

# Глава 1. Исследование

В данной главе будет проведено исследование включающее в себя анализ и сравнение доступных решений. По окончании главы к программе будут составлены определённые требования для её лучшей реализации.

## 1.1. Исследование и анализ существующих программных решений

При поиске в Интернете человек может найти огромное количество различных генераторов кроссвордов. Это могут быть варианты с простым дизайном и функционалом, позволяющие создать простой со стороны дизайна, но правильно составленный, решаемый кроссворд. Существуют так же и полноценные приложения с более точной и гибкой настройкой.

При анализе существующих решений были выявлены следующие значащие критерии:

1. цена;
2. сложность использования;
3. возможности настройки;
4. платформа.

Цена один из важнейших критериев, которым руководствуется человек при выборе. Генератор может быть платным, или бесплатным, или иметь платные подписки или версии. В работе будет указана цена и наличие платных вариаций.

Для удобства пользователя необходимо чтобы кроссворд легко создавался, поэтому будет оцениваться насколько простым для пользователя будет взаимодействие с генератором.

В работе будут описаны возможности настройки: насколько гибко можно настроить генератор кроссворда под конкретные требования, например форма кроссворда или размер поля в котором генерируется кроссворд.

Так же оцениваться будет на каких устройствах доступны генераторы.

В конце разбора всех генераторов по критериям будет проведён сравнительный анализ и представленна сравнительная таблица для быстрого анализа каждой вариации.

### 1.1.1. Crossword compiler 11

Crossword Compiler 11 - это программное обеспечение для создания кроссвордов и других игр со словами[1].

Цена: Стоимость программы Crossword Compiler 11 зависит от выбранной лицензии. Для домашнего использования существует бесплатная демо-версия программы, но с ограниченным набором функций. Полная версия программы с безлимитным доступом ко всем функциям стоит 2130 рублей в месяц.

Сложность использования: Программа Crossword Compiler 11 имеет функцию автоматической генерации кроссвордов, которая может быть использована для создания базового кроссворда за несколько минут. Однако, для создания кроссвордов более высокого уровня сложности, может потребоваться дополнительное время.

Возможность настройки: Программа Crossword Compiler 11 имеет широкий диапазон настроек, которые позволяют настроить кроссворд под нужды пользователя. Например, пользователь может настроить количество блоков в кроссворде, размер ячеек, цветовые схемы, шрифты и т.д. Также возможно добавление пользовательских слов и подсказок.

Доступность на разных платформах: Crossword Compiler 11 доступен только для ПК с операционной системой Windows. Однако, созданные кроссворды могут быть экспортированы в различные форматы, такие как HTML, PDF, RTF, и могут быть использованы на разных платформах.

Подводя итог, Crossword Compiler 11 - это программное   
обеспечение для создания кроссвордов с широкими   
возможностями настройки, программное обеспечение   
предлагает множество функциональных возможностей,   
которые упрощают процесс создания кроссвордов.   
Crossword Compiler 11 доступен только для ПК с операционной системой Windows.

### 1.1.2. My Crossword Maker

My Crossword Maker - это онлайн-сервис для создания кроссвордов[2].

Цена: My Crossword Maker предлагает бесплатную версию со всеми основными функциями. Существует платная версия, которая дает больше возможностей, таких как сохранение неограниченного количества кроссвордов и доступ к более продвинутым функциям. Стоимость подписки составляет около 400 рублей в месяц.

Сложность использования: Генерация кроссвордов может происходить автоматически или же пользователю будет предоставлен функционал для самостоятельного создания кроссворда, использование которого может вызвать небольшие затруднения.

Возможность настройки: My Crossword Maker предоставляет множество настроек для создания кроссвордов. Можно выбрать размер кроссворда, количество слов, их направление, а также настроить ограничения на использование слов или фраз.

Доступность на разных платформах: My Crossword Maker доступен на любой платформе, где есть доступ к интернету. Создавать кроссворды можно на компьютере, планшете или смартфоне, используя любой веб-браузер.

My Crossword Maker - это онлайн-инструмент для создания кроссвордов, который предоставляет множество настроек и возможностей. Бесплатная версия позволяет создавать кроссворды без ограничений, а платная версия предоставляет более продвинутые функции.

### 1.1.3. Crossword puzzle games

Crossword puzzle games – сайт с готовыми примерами кроссвордов и возможностью создания собственных[3].

Цена: Сайт Crossword puzzle games предоставляет бесплатный доступ ко всем кроссвордам и генерации своихсобственных.

Сложность использования: из-за того, что пользователь может только ввести список слов и их определений сложность его использования является минимальной

Возможность настройки: настройки как таковые отсутствуют, пользователь может только ввести список слов и определений или воспользоваться уже готовыми кроссвордами.

Доступность на разных платформах: Сайт Crossword puzzle games доступен на разных платформах, включая ПК, мобильные устройства и планшеты.

Сайт Crossword puzzle games - бесплатный ресурс с широким выбором кроссвордов разной сложности. Также пользователи могут составить собственный кроссвордов. Являясь сайтом доступен на всех платформах с выходом в интернет

### 1.1.4. EclipseCrossword

EclipseCrossword предоставляет собой инструмент для создания собственных кроссвордов[4].

Цена: Инструмент для создания кроссвордов на сайте EclipseCrossword полностью бесплатный, что делает его доступным для всех пользователей.

Сложность использования: Создание кроссвордов в приложении EclipseCrossword не требует особых навыков. Создание промтейшего кроссворда не займёт больше 5 минут.

Возможность настройки: Пользователи создают собственных список слов и после выбирают из предложенных расстановок слов.

Доступность на разных платформах: Инструмент для создания кроссвордов EclipseCrossword доступен только на ПК с операционной системой Windows.

Итак, EclipseCrossword - бесплатный инструмент для создания кроссвордов, доступный только на ПК с операционной системой Windows. Создание кроссвордов предельно простое и не вызывает никаких трудносей. Пользователи могут ввести собственный список слов, и выбрать желаемую ими расстановку этих слов

### 1.1.5. Сравнительный анализ

После описания и разбора каждого из 4 генераторов стоит составить сравнительную таблицу (см. таблица 1.1). Для всех критериев будут определены шкалы оценок. Их всего три:

1. абсолютная;
2. бинарная;
3. тернарная.

Абсолютная шкала оценок будет относиться к цене. Абсолютная шкала оценок показывает какое-то целое число. Для цены это будет количество рублей.

Бинарная шкала оценок принимает только два значения: простая, либо продвинутая. Данная шкала будет относиться к критериям сложности использования и возможности настройки.

Тернарная шкала оценок включает три значения: сайт, ОП Windows, компьютерные ОП.

Таблица 1.1 - сравнительный анализ существующих программ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий**  **Генератор** | **Цена**  **(рубли)** | **Сложность использования** | **Возможность настройки** | **Доступность на разных платформах** |
| Crossword complier 11 | 2130 | Продвинутая | Продвинутая | ОП Windows |
| My crossword maker | 400 | Проcтая | Продвинутая | Сайт |
| Crossword puzzle games | 0 | Простая | Простая | Сайт |
| EclipseCrossword | 0 | Простая | Простая | ОП Windows |

Так как пользователь с малой вероятностью будет платить ежемесечную подписку ради создания кроссворда, у Crosswordcomplier 11 и My crossword maker мы будем оценивать их бесплатные вариации.

Сложность использования: Сrossword puzzle games самый простой генератор из представленных, но это так же отражается на его функционале

и малом количестве вводимых слов, EclipseCrossword схож по функционалу, но позволяет ввести больше слов. Функционал Crossword compiler 11 из-за огромного количества различных кнопок можно считать перегруженным и затруднительным для использования. My crossword maker предоставляет возможность авто-генерации, а так же простой и удобный функционал для самоличного создания кроссворда, что является самым оптимальным вариантом из представленных.

Возможность настройки: у Сrossword puzzle games и EclipseCrossword возможность настройки отсутствует. Crossword complier 11 предоставляет самый широкий спектр настроек из всех представленных генераторов. My crossword maker предоставляет все необходимые настройки для создания кроссворда, из-за простоты использования пользователем такая возможность настройки будет самой лучшей.

Доступность на разных платформах: хоть сайты и доступны на всех платформах их использование на мобильных устройствах или в условиях медленного интернета не удобно, поэтому будет написана программа.

## 1.2. Инструментальный аппарат

При написании курсовой работы будут использованы:

1. язык программирования C#;
2. среда разработки программного обеспечения Microsoft Visual Studio[5];
3. платформа пользовательского интерфейса для создания классических приложений Windows Windows Form[6].

## 1.3. Постановка задач

Все программы и сайты, которые были описаны выше, по-своему уникальны. Сделав краткие выводы и описания для каждого из генераторов, а также проведя сравнительный анализ, стоит выделить основные вещи, которые будут учитываться при разработке собственного генератора кроссвордов:

1. сделать собственное приложение, ведь хоть сайты и доступны на всех устройствах, их функционал уступает программным версиям;
2. необходимо создать понятный и удобный интерфейс, чтобы пользователь было комфортно использовать программу;
3. добавить настройки кроссворда, для того чтобы пользователь мог настроить некоторые параметры под себя;
4. стоит создать обратную связь с пользователем, при решении кроссворда показывать некоторые параметры, например количество правильно отгаданных слов.

**Выводы**

В первой главе были рассмотрены существующие решения на рынке, а именно 4 генератора кроссвордов: «Crossword Compiler 11», «My Crossword Maker», «Crossword puzzle games», «EclipseCrossword». Для каждого генератора было представлено небольшое описание.

Все четыре вариации генераторов кроссвордов были разобраны по таким критериям как: ценa, сложность генерации, возможности настройки и доступность на разных платформах. Для быстроты и удобства просмотра критерии были занесены в сравнительную таблицу.

На основе всей представленной информации был проведен сравнительный анализ, по итогам которого были сформированы функциональные требования, которые можно коротко охарактеризовать как:

1. создать собственное приложение с достаточным функционалом для любителей кроссвордов;
2. создать минималистичный и удобной дизайн;
3. создать вариативную настройку кроссворда;
4. построить обратную связь с пользователем.

# Глава 2. Моделирование

В данном разделе будет проведено моделирование работы программы генератор кроссвордов на основе функциональных требований, которые были выдвинуты в первой главе. Структура работы программы генератор кроссвордов состоит из взаимосвязанных между собой модулей, каждый из которых отвечает за свои поставленные задачи и реализует некоторые алгоритмы. В данной работе было выделено 5 модулей:

1. модуль меню;
2. модуль авто-создания кроссворда;
3. модуль пользовательского создания кроссворда;
4. модуль решения кроссворда;
5. модуль подведения итогов.

Для каждого из модулей будет представлено свое описание работы на основании блок-схем. Для наглядности была сформирована функциональная схема игры (рис. 2.1).

## 2.1. Функциональная схема генератора кроссвордов

На рисунке 2.1 представлена функциональная схема игры.

Модуль меню – встречает пользователя при запуске приложения. Выводит кнопки инструкции, создания кроссворда и выхода.

Модуль авто-создания кроссворда - автоматически создает кроссворд на основе введенных параметров. Он использует список слов и определений введённых пользователем. Затем модуль располагает слова в сетке таким образом, чтобы они пересекались по правилам кроссворда

Модуль пользовательского создания кроссворда - позволяет пользователю самостоятельно создавать кроссворд. Пользователь выбирает размеры кроссворда и расставляет слова в сетке.

Модуль решения кроссворда - позволяет пользователю решить созданный им или сгенерированный программой кроссворд, получить подсказки, если возникли трудности.

Модуль подведения итогов - выводит на экран результаты игры: количество правильных ответов, время, затраченное на игру, количество использованных подсказок.

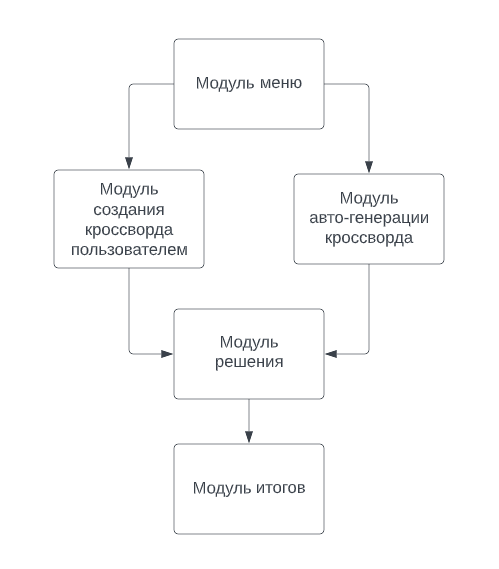


Рисунок 2.1 - функциональная схема программы.

## 2.2. Описание модулей и алгоритмов генератора кроссвордов

В данной подглаве будет реализовано проектирование всех модулей генератора кроссворда и описан их принцип работы. Также будут рассмотрены алгоритмы, которые составляют основу модулей.

### 2.2.1. Модуль меню

Принцип работы модуля меню заключается в том, чтобы обеспечить пользователя о доступных вариантах действий в приложении.

Инструкция: после выбора данной опции отрывается окно с описанием функций программы и инструкциями как их использовать

Выход: закрывает приложение

Создание кроссворда: пользователю на выбор даётся самостоятельное создание кроссворда или же авто-генерация кроссворда на основе введённого пользователем списка слов.

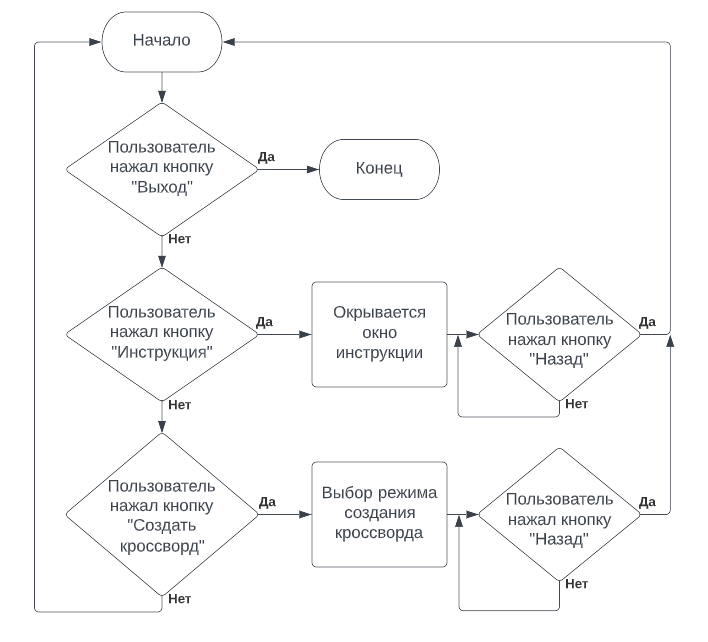


Рисунок 2.2 - блок-схема модуля меню.

### 2.2.2. Модуль авто-создания кроссворда

Принцип работы модуля авто-создания кроссворда заключается в том, что программа автоматически создает кроссворд на основе заданных пользователем слов и определений.

Ввод списка слов и его проверка: Пользователь вводит необходимое ему количество слов и определений, после программа проверяет можно ли из данного списка слов составить кроссворд, если нет то просит пользователя вновь ввести список слов.

Расстановка слов и создание пустой сетки: после проверки введённого пользователем списка слов программа расставляет их в сетке. По завершении программа выдаёт готовую для решения пустую сетку кроссворда.

Добавление определений: после размещения слов в кроссворде, программа добавляет определения для каждого слова. Определения выставляются в соответсвии со словами из списка введённого пользователем и выводятся в отдельном окне.



Рисунок 2.3 - блок схема модуля авто-создания кроссворда

### 2.2.3. Модуль пользовательского создания кроссворда

Принцип работы модуля пользовательского создания кроссворда заключается в том, что пользователь сам создает кроссворд, используя редактор кроссворда, доступный в программе.

Размещение слов: пользователь выбтрает линию в кроссворде размеры которого он указал. Пользователь вводит слово, а так же его определение. После программа проверяет размещено ли слово по правилам кроссворда.

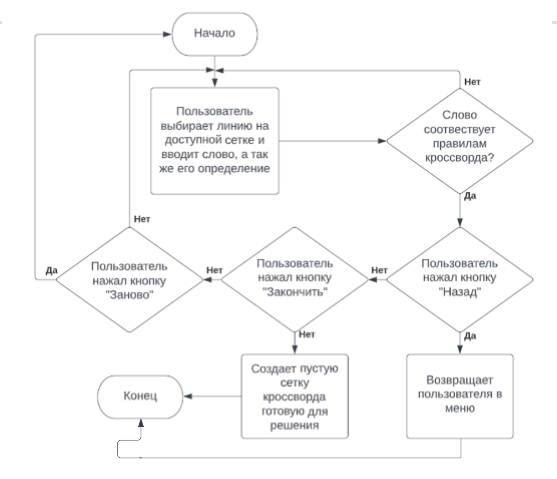


Рисунок 2.4 - блок-схема модуля пользовательского создания кроссворда

Меню кнопок: в данном редакторе кроссворда доступны такие функции как: заново – стираются все введённые пользователем слова и пользователь может вновь заполнить кроссворд, назад – возвращает в меню программы, закончить – запускает алгоритм создания пустой сетки.

Создание пустой сетки: программа создаёт пустую сетку кроссворда готовую для решения на основе заполненного пользователем кроссворда.

Добавление определений: высвечивает пользователю окно, где написаны определения слов написанные им же, а так же указана их нумерация в кроссворде и направление.

### 2.2.4. Модуль решения кроссворда

Принцип работы модуля решения пользователем кроссворда заключается в том, что программа позволяет пользоватю заполнять игровое поле и использовать некоторые функции.

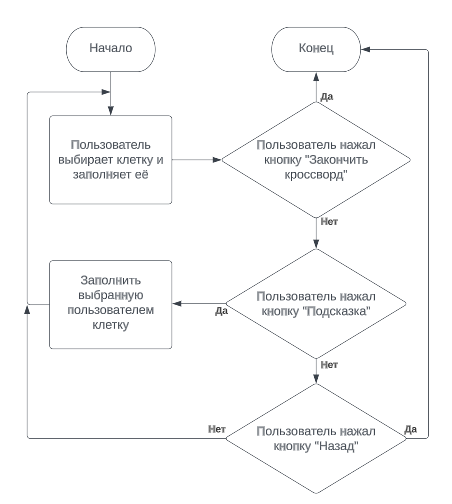


Рисунок 2.5 - блок схема модуля решения кроссворда

Ввод ответов: пользователь вводит ответы на определения в соответствующие клетки кроссворда.

Подсказки: при нажатии на кнопку «Подсказка», пользователь выбирает клетку значение которой он хочет узнать, после этого значение клетки изменить будет невозможно.

Окончание игры: программа завершается когда пользователь нажимает кнопку «Закончить кроссворд» или «Назад».

### 2.2.5. Модуль подведения итогов

Принцип работы модуля подведения итогов кроссворда заключается в том, что программа вычисляет результаты игры и отображает их пользователю.

Подсчет правильно решенных слов: программа подсчитывает количество правильно решенных слов и отображает его пользователю.

Расчет использованных подсказок: программа отслеживает количество подсказок использованных пользователем.

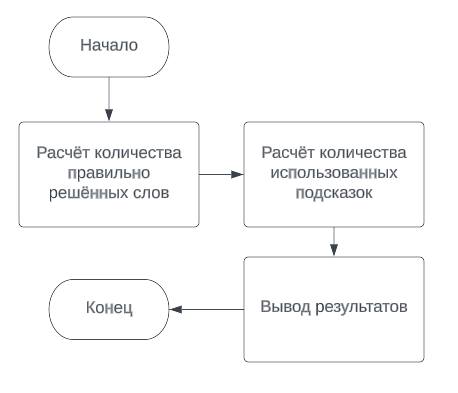


Рисунок 2.6 - блок-схемамодуля подведения итогов.

Вывод результатов: программа отображает пользователю результаты , включая количество правильно решенных слов, использование подсказок.

## Выводы

В данной главе было проведено моделирование работы игры «Кроссворд» с генератором. Была разработана функциональная схема а так же были разработаны 5 модулей:

1. модуль меню;
2. модуль авто-создания кроссворда;
3. модуль пользовательского создания кроссворда;
4. модуль решения кроссворда;
5. модуль подведения итогов.

Для наглядности были созданы блок схемы для каждого из модулей. Было составлено описание каждого модуля. Описание модулей включает в себя объяснение того, как работают данные модули и какие алгоритмы и задачи они выполняют.

В соответсвии с поставленными задачами в предыдущей главе, было реализованно простое и понятное меню.

Для более обширных настроек кроссворда был спроектирован модуль пользовательского создания кроссворда.

Для поддержания обратной связи с пользователем был создан модуль подведения итогов.

# Глава 3. Разработка

В рамках данной работы была разработана игра «Кроссворд» с генерацией. В третьей главе будут представлены скриншоты самой игры и отрывки кода, которые отвечают за выполнение определенных алгоритмов и задач.

## 3.1. Окно с меню

При запуске программы на экране пользователя появляется окно с меню(см рис 3.1). Пользователю доступны кнопки “Создать кроссворд”, “Инструкции” и “Выход”.

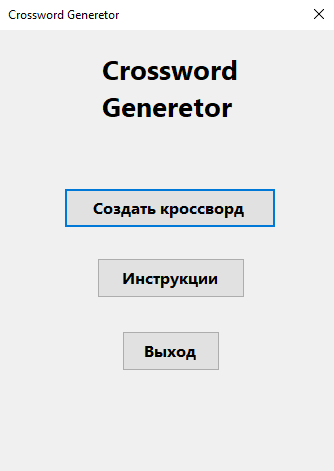


Рисунок 3.1 – окно меню

При нажатии на кнопку “Создать кроссворд” запускается меню создания кроссворда. В создании кнопок помогают методы selfgamecreatebutton() и compgamecreatebutton()(см. приложение А),создающие новые поля и кнопки.

При нажатии на кнопку “Инструкции” открывается окно, объясняющее управление в данном приложении(см. приложение Б).

При нажатии на кнопку “Выход” приложение закрывается.

Выше перечисленные кнопки работают благодаря обработчикам событий, представленных в листинге 3.1.

private void ExitBut\_Click(object sender, EventArgs e)

{

this.Close();

}

private void GameStartBut\_Click(object sender, EventArgs e)

{

GameStartBut.Enabled = false;

GameStartBut.Visible = false;

ExitBut.Visible = false;

ExitBut.Enabled = false;

InfBut.Enabled = false;

InfBut.Visible = false;

selfgamecreatebutton();

autogamecreatebutton();

Button back = new Button();

back.Name = "back";

back.Location = new Point(122, 380);

back.Size = new Size(98,30);

back.Text = "Назад";

back.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold);

back.Click += new System.EventHandler(back\_Click);

Controls.Add(back);

}

private void InfBut\_Click(object sender, EventArgs e)

{

instructs instr = new instructs(this);

this.Hide();

instr.Show();

}

Листинг 3.1 – обработчики событий окна меню

## 3.2. Окно выбора создания кроссворда

После нажатия кнопки “Создать кроссворд ” открывается меню выбора режима создания(см рис 3.2).

При нажатии кнопки “создать лично” пользователю необходимо ввести размеры кроссворда, после нажатия кнопки “Начать”, приложение считывает размеры кроссворда с помощью метода StartUserCreate\_Click(), далее открывается окно пользовательского создания кроссворда.

При нажатии кнопки “Сгенерировать” пользователю необходимо ввести количество слов которое он собирается ввести, после того как метод StartCompCreate\_Click() считает введённые данные откроется новая форма – SubForm.

При нажатии кнопки «Назад» пользователь возвращается к главному меню.

Выше перечисленные кнопки работают благодаря обработчикам событий, представленных в листинге 3.2.

private void StartUserCreate\_Click(object sender, EventArgs e){

if ((Controls["HeightBox"].Text != "" ) &&(Regex.IsMatch(Controls["HeightBox"].Text, @"^\d+$"))){

UserCreate userCreate = new UserCreate(Convert.ToInt32(Controls["HeightBox"].Text),this);

UserCreate.ShowMainAfterBack handler = new UserCreate.ShowMainAfterBack(Mainhelpfunc);

userCreate.showback += handler;

userCreate.Show();

this.Hide();}

}

private void StartCompCreate\_Click(object sender, EventArgs e){

if ((Controls["SizeBox"].Text != "") && (int.TryParse(Controls["SizeBox"].Text,out \_)))

{

if(Convert.ToInt32(Controls["SizeBox"].Text) <= 30)

{

SubForm test = new SubForm(this, Convert.ToInt32(Controls["SizeBox"].Text));

test.Show();

this.Hide();

}

}

}

private void back\_Click(object sender, EventArgs e){

GameStartBut.Enabled = true;

GameStartBut.Visible = true;

ExitBut.Visible = true;

ExitBut.Enabled = true;

InfBut.Enabled = true;

InfBut.Visible = true;

Controls["back"].Dispose();

Controls["SelfGameCreate"].Dispose();

Controls["StartUserCreate"].Dispose();

Controls["HeightInfo"].Dispose();

Controls["HeightBox"].Dispose();

Controls["CompGameCreate"].Dispose();

Controls["StartCompCreate"].Dispose();

Controls["SizeInfo"].Dispose();

Controls["SizeBox"].Dispose();}

Листинг 3.2 – обработчики событий окна выбора создания кроссворда

SubForm нужна для того чтобы пользователь мог комфортно ввести слова и их определения(см.рис 3.3).



Рисунок 3.2 – окно выбора режима создания

При нажатии кнопки «Назад» пользователь возвращается   
к главному меню.

При нажатии кнопки «Готово» создаётся структура ToAuto(см. приложение А), которая после передаётся в модульавто создания.

Обработчик события кнопки «Готово» будет представлен в листинге 3.3.

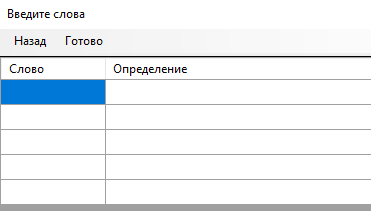


Рисунок 3.3 – SubForm

private void готовоToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<ToAuto> rnd = createList();

AutoCross NewAuto = new AutoCross(menu, rnd);

this.Close();

NewAuto.Show();

}

Листинг 3.3 – обработчики события кнопки «Готово»

## 3.3. Окно пользовательского создания

После того как пользователь нажал кнопку “Создать лично” ввёл размеры кроссворда и нажал кнопку “Начать”, открывается окно пользовательского создания(см рис 3.3).

В окне представлено 4 кнопки – “Закончить”, “Заново”, “Посмотреть список слов” и “Назад”.

При нажатии кнопки “Закончить” открывается окно решения кроссворда.

При нажатии кнопки “Заново”, поле обновляется и пользователь может по новой ввести слова.

При нажатии кнопки “Посмотреть список слов” открывается окно с информацией по уже введённым словам(см 3.4)

При нажатии кнопки “Назад” пользователь возвращается в главное меню.

Так же в окне представлено поле введённого пользователем размера. При двойном нажатии на клетку она выделяет клетки по вертикали и горизонтали. При двойном нажатии на клетку в выделенных линиях, открывается окно где пользователю необходимо ввести слово и его определение(см. приложение Б). Если в диапазоне который выбрал пользователь уже находились буквы, они будут отмечены в окне и их будет невозможно изменить, происходит это благодаря методу helpcreated() в форме Crosshelp(см приложение А). После этого слово появляется на поле, а информации о нём добавляется в список определений.

Обработчики событий данных кнопок, а также обработчик события двойного клика по полю будет представлен в листинге 3.4.

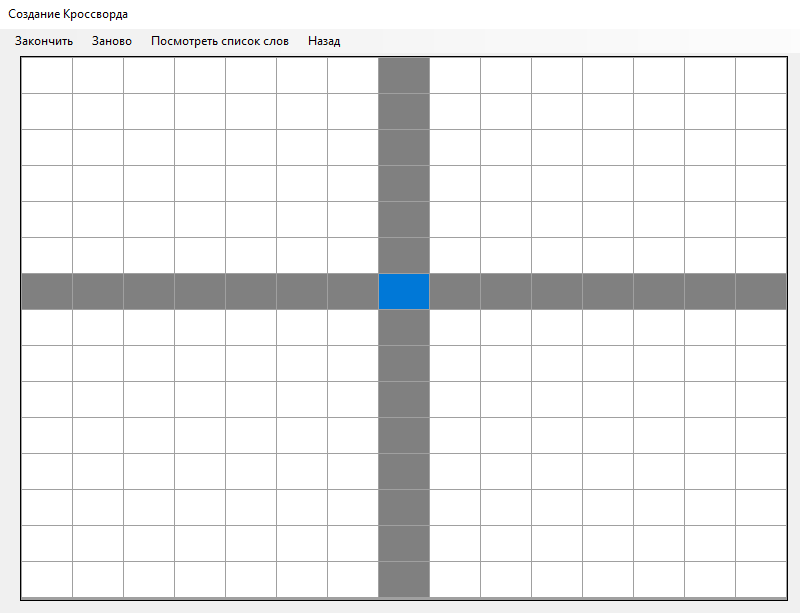


Рисунок 3.3 – окно пользовательского создания.

Окно с информацией по уже введённым словам реализуется при помощи формы DefTable(см. приложение А). Так же в форме присутствует кнопка обновить реализованная при помощи обработчика события, а так же делегата(см приложение А).

private void посмотретьСписокСловToolStripMenuItem\_Click(object sender, EventArgs e){

DefTable deftable = new DefTable(Crosslist);

deftable.SetDesktopLocation(this.Location.X+this.Width+1, this.Location.Y);

DefTable.defboardupdate handler = new DefTable.defboardupdate(Defhelpfunc);

deftable.update += handler;

deftable.Show();

}

private void зановоToolStripMenuItem\_Click(object sender, EventArgs e){

Crosslist = new List<crossW>();

lastrow = 0;

lastcol = 0;

board.Rows.Clear();

board.Columns.Clear();

CreateBoard();

}

private void закончитьToolStripMenuItem\_Click(object sender, EventArgs e){

board.Rows.Clear();

board.Columns.Clear();

CreateGameBoard();

}

private void board\_CellDoubleClick(object sender, DataGridViewCellEventArgs e){

if ((e.RowIndex == lastrow || e.ColumnIndex == lastcol)

&& ((board.Rows[e.RowIndex].Cells[e.ColumnIndex].Style.BackColor == Color.Cornsilk)

&& (board.Rows[lastrow].Cells[lastcol].Style.BackColor == Color.Cornsilk)) == false){

ChooseWordZone(e.RowIndex, lastrow, e.ColumnIndex, lastcol);

WordWriteDown(Crosslist[Crosslist.Count-1]);

ClearCellCross(lastrow, lastcol);

lastrow = e.RowIndex;

lastcol = e.ColumnIndex;

}

else{

ClearCellCross(lastrow, lastcol);

CreateCellCross(e.RowIndex, e.ColumnIndex);

lastrow = e.RowIndex;

lastcol = e.ColumnIndex;

}

}

Листинг 3.4 – обработчики событий кнопок и двойного нажатия.

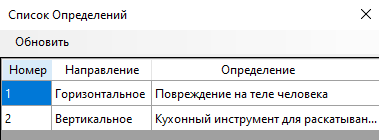


Рисунок 3.4 – окно со списком определений

## 3.4. Модуль авто генерации

Если пользователь выберет сгенерировать кроссворд, и заполнит форму SubForm, запустится модуль авто генерации. Метод Fill() заполняет список слов list<string> words, метод CrossGenerate()(см приложение А) расставляет их в игровую сетку. Самым важным методом внутри CrossGenerate() является Func().В листинге 3.5 представлены методы Fill() и Func()

private void Fill(){

int size 20;

foreach(var elem in original){

words.Add(elem.word);

}

gameChars = new cell[size, size];

numbers = new int[size, size];

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

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

gameChars[y, x] = new cell();

gameChars[y, x].data = '0';

gameChars[y, x].y = y;

gameChars[y, x].x = x;

}

}

private void func(HelpWithWords data){

if (words.Count == 0) return;

CurWordCells = FillCurCellList(data);

List<HelpWithWords> PossibleWords = new List<HelpWithWords>();

foreach (var elem in CurWordCells){

if (elem.state != cell.State.Both){

for (int j = 0; j < words.Count; j++){

if (words[j].Contains(elem.data) && !haveIn(PossibleWords, words[j]) && collision(elem, words[j]))

{

PossibleWords.Add(new HelpWithWords(words[j],elem));

break;

}

}

}

}

if(PossibleWords.Count >= 2){

HelpWithWords n1 = PossibleWords[rnd.Next(PossibleWords.Count - 1)];

if (haveIn(help, n1.word)) return;

PossibleWords.Remove(n1);

words.Remove(n1.word);

wordFill(n1.startCell, n1.word);

func(help[help.Count-1]);

HelpWithWords n2 = PossibleWords[rnd.Next(PossibleWords.Count - 1)];

if (haveIn(help, n2.word)) return;

PossibleWords.Remove(n2);

words.Remove(n2.word);

wordFill(n2.startCell, n2.word);

func(help[help.Count - 1]);

}

else if (PossibleWords.Count == 1){

HelpWithWords n1 = PossibleWords[rnd.Next(PossibleWords.Count - 1)];

if (haveIn(help, n1.word)) return;

PossibleWords.Remove(n1);

words.Remove(n1.word);

wordFill(n1.startCell, n1.word);

func(help[help.Count - 1]);

}

}

Листинг 3.5 – методы Fill() и Funс()

## 3.5. Окно решения кроссворда

При нажатии кнопки “Закончить” в окне пользовательского создания или после ввода всех слов в режиме авто генерации, открывается окно решения кроссворда (см рис 3.6) а так же окно списка определений(см рис 3.4).

В окне представлены 4 кнопки – “Закончить кроссворд”, “Назад” и “Подсказка”,При нажатии кнопки “Закончить кроссворд” открывает окно подведения итогов.

При нажатии кнопки “Назад” возвращает пользователя в главное меню.

При нажатии кнопки “Подсказка”, пользователь может двойным кликом выбрать клетку, где хочет увидеть букву.

На поле, в белые клетки, можно вставлять буквы, бледно-жёлтые клетки показывают, где начинается слова. Так же благодаря методу board\_EditingControlShowing() в клетку нельзя вписать более одного символа.

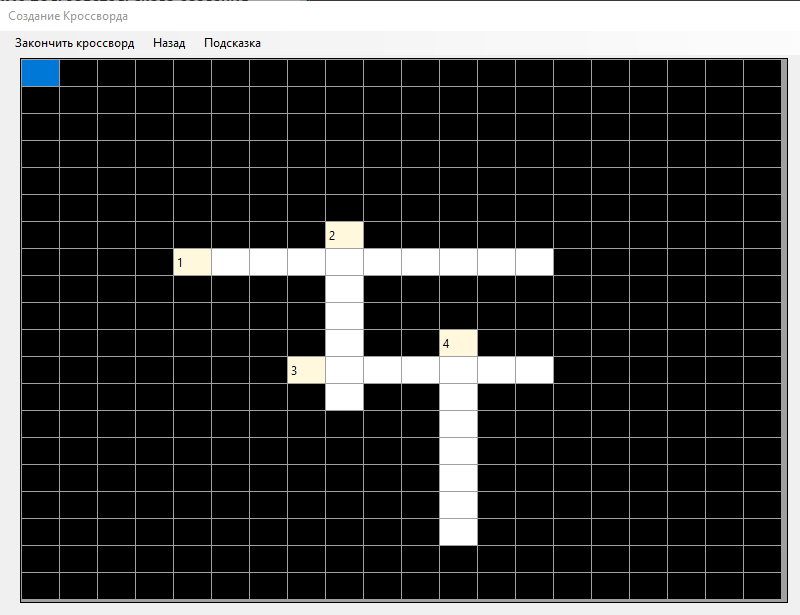


Рисунок 3.6 – окно решения кроссворда

При нажатии кнопки «Закончить кроссворд» запускается ResultsOutPut(), который подсчитывает количество правильно введённых слов. В листинге 3.6 представлена реализация методов ResultsOutPut() и board\_EditingControlShowing().

private int ResultsOutPut()

{

int ans = 0;

foreach(var elem in help)

{

CurWordCells = FillCurCellList(elem);

bool got = true;

foreach(cell data in CurWordCells)

{

if(data.data.ToString() != board[data.x,data.y].EditedFormattedValue.ToString())

{

got = false;

break;

}

}

if (got) ans++;

}

return ans;

}

private void board\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e){

if (e.Control is TextBox)

{

(e.Control as TextBox).MaxLength = 1;

}

}

Листинг 3.6 - методы ResultsOutPut() и board\_EditingControlShowing()

## 3.6. Окно подведения итогов

После нажатия кнопки “Закончить кроссворд”,   
открывается окно подведения итогов (см рис 3.7). Оно показывает   
количество использованных пользователем подсказок, а также количество отгаданных слов.

После нажатия кнопки «К меню» пользователь вернётся в главное меню программы, и сможет по новой создать кроссворд.

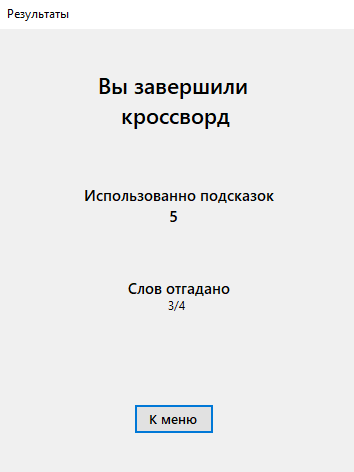


Рисунок 3.7 – окно подведения итогов

Заполняется окно подведения итогов благодаря конструктору данной формы, а так же методу filler(). Описание данных элементов представлено в листинге 3.7.

public Results(int helpCount,int wordCount,int listlen,MainMenu prevwind)

{

menu = prevwind;

InitializeComponent();

filler(helpCount,wordCount,listlen);

}

private void filler(int helpCount, int wordcount,int listlen)

{

HelpCount.Text = helpCount.ToString();

wordCount.Text = $"{wordcount}/{listlen}";

}

Листинг 3.7 – конструктор формы Results и метод filler()

## Выводы

В данной главе была проведена разработка игры «Кроссворд» с генерацией. Игра была разработана на основании функциональных требований, поставленных в первой главе, а именно:

1. было создано приложение с функционалом, достаточным для любителей кроссвордов;
2. программа выполнена в минималистичном стиле;
3. был создан отдельный режим, где пользователь может создать кроссворд на своё усмотрение.

Были представлены скриншоты основных и самых важных окон программы. Рассмотрены основные возможности программы. В главе были представлены листинги кода, описывающие работу кнопок следующих окон:

1. окно меню;
2. окно выбора режима создания;
3. окно пользовательского создания;
4. окно решения кроссворда.

Были продемонстрированы листинги обработки событий двойного нажатия по полю. Также был приведен листинг некоторых важных методов программы таких как: авто-заполнение сетки и наполнения списка слов.

# Заключение

В ходе данной курсовой работы была спроектирована и реализована игра «Кроссворд» с генератором.

Были рассмотрены существующие решения на рынке, а именно 4 генератора кроссвордов: «Crossword Compiler 11», «My Crossword Maker», «Crossword puzzle games», «EclipseCrossword».

Всем четырем вариациям генераторов было дано краткое описание, так же они были разобраны по таким критериям как: ценa, сложность генерации, возможности настройки и доступность на разных платформах. Для быстроты и удобства просмотра была создана сравнительная таблица.

На основе всей представленной информации был проведен сравнительный анализ, по итогам которого были сформированы задачи и требования.

На основе поставленных задач в ходе курсовой работы было проведено моделирование работы игры «Кроссворд» с генератором. Была разработана функциональная схема а так 5 модулей:

1. модуль меню;
2. модуль авто-создания кроссворда;
3. модуль пользовательского создания кроссворда;
4. модуль решения кроссворда;
5. модуль подведения итогов.

Для наглядности были созданы блок схемы для каждого из модулей и составлено описание.

В соответсвии с поставленными задачами в предыдущей главе, было реализованно простое и понятное меню.

Для более обширных настроек кроссворда был спроектирован модуль пользовательского создания кроссворда.

Для поддержания обратной связи с пользователем был создан модуль подведения итогов.

Была проведена разроботка игры «Кроссворд» с генератором

1. было создано приложение с функционалом, достаточным для любителей кроссвордов;
2. программа выполнена в минималистичном стиле;
3. был создан отдельный режим, где пользователь может создать кроссворд на своё усмотрение;
4. для обратной связи было создано окно подведения итогов.

Были представлены скриншоты основных окон программы. Рассмотрены основные возможности программы. Были представлены листинги кода, описывающие работу кнопок следующих окон:

1. окно меню;
2. окно выбора режима создания;
3. окно пользовательского создания;
4. окно решения кроссворда.

Были продемонстрированы листинги обработки событий двойного нажатия по полю. Также был приведен листинг некоторых важных методов программы таких как: авто-заполнение сетки и наполнения списка слов.

# Список использованной литературы

1. Crossword Compiler: Crossword Puzzle Maker Software /   
   crossword-compiler.com [Электронный ресурс]. Режим доступа <https://www.crossword-compiler.com/>. Дата обращения: 07.03.2023.
2. Create Your Own Crossword | Easy Crossword Puzzle Maker / mycrosswordmaker.com [Электронный ресурс]. Режим доступа <https://mycrosswordmaker.com/>. Дата обращения: 07.03.2023.
3. Crossword Puzzle Games - Play Or Create Your Own Puzzles / crosswordpuzzlegames.com [Электронный ресурс]. Режим доступа <https://www.crosswordpuzzlegames.com/>. Дата обращения: 07.03.2023.
4. EclipseCrossword / eclipsecrossword.com [Электронный ресурс]. Режим доступа <https://www.eclipsecrossword.com/>. Дата обращения: 07.03.2023.
5. Visual Studio: IDE и редактор кода для разработчиков и групп, работающих с программным обеспечением / visualstudio.microsoft.com [Электронный ресурс]. Режим доступа: <https://visualstudio.microsoft.com/ru/>. Дата обращения: 07.03.2023.
6. Документация по Windows Forms для .NET | Microsoft Learn / learn.microsoft.com [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8>. Дата обращения: 07.03.2023.

# Приложение А. Листинг кода

## Приложение А.1. Программный код формы MainMenu

public partial class MainMenu : Form

{

public MainMenu()

{

InitializeComponent();

}

private void ExitBut\_Click(object sender, EventArgs e)

{

this.Close();

}

private void GameStartBut\_Click(object sender, EventArgs e)

{

GameStartBut.Enabled = false;

GameStartBut.Visible = false;

ExitBut.Visible = false;

ExitBut.Enabled = false;

InfBut.Enabled = false;

InfBut.Visible = false;

selfgamecreatebutton();

autogamecreatebutton();

Button back = new Button();

back.Name = "back";

back.Location = new Point(122, 380);

back.Size = new Size(98,30);

back.Text = "Назад";

back.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold);

back.Click += new System.EventHandler(back\_Click);

Controls.Add(back);

}

private void back\_Click(object sender, EventArgs e)

{

GameStartBut.Enabled = true;

GameStartBut.Visible = true;

ExitBut.Visible = true;

ExitBut.Enabled = true;

InfBut.Enabled = true;

InfBut.Visible = true;

Controls["back"].Dispose();

Controls["SelfGameCreate"].Dispose();

Controls["StartUserCreate"].Dispose();

Controls["HeightInfo"].Dispose();

Controls["HeightBox"].Dispose();

Controls["CompGameCreate"].Dispose();

Controls["StartCompCreate"].Dispose();

Controls["SizeInfo"].Dispose();

Controls["SizeBox"].Dispose();

}

private void selfgamecreatebutton(){

Button SelfGameCreate = new Button();

SelfGameCreate.Name = "SelfGameCreate";

SelfGameCreate.Location = new Point(27, 160);

SelfGameCreate.Size = new Size(121, 40);

SelfGameCreate.Text = "Создать лично";

SelfGameCreate.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold);

SelfGameCreate.Click += new System.EventHandler(SelfGameCreate\_Click);

Controls.Add(SelfGameCreate);

Label HeightInfo = new Label();

HeightInfo.Name = "HeightInfo";

HeightInfo.Location = new Point(27, 230);

HeightInfo.AutoSize = true;

HeightInfo.Text = "Введите размеры\r\nКроссворда";

HeightInfo.Visible = false;

Controls.Add(HeightInfo);

TextBox HeightBox = new TextBox();

HeightBox.Name = "HeightBox";

HeightBox.MaxLength = 2;

HeightBox.Location = new Point(27, 265);

HeightBox.Visible = false;

HeightBox.Enabled = false;

Controls.Add(HeightBox);

Button StartUserCreate = new Button();

StartUserCreate.Name = "StartUserCreate";

StartUserCreate.Location = new Point(27, 330);

StartUserCreate.Size = new Size(121, 35);

StartUserCreate.Text = "Начать";

StartUserCreate.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold);

StartUserCreate.Visible = false;

StartUserCreate.Enabled = false;

StartUserCreate.Click += new System.EventHandler(StartUserCreate\_Click);

Controls.Add(StartUserCreate);

}

private void autogamecreatebutton(){

Button CompGameCreate = new Button();

CompGameCreate.Name = "CompGameCreate";

CompGameCreate.Location = new Point(175 + 7, 160);

CompGameCreate.Size = new Size(121, 40);

CompGameCreate.Text = "Сгенерировать";

CompGameCreate.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold);

Controls.Add(CompGameCreate);

CompGameCreate.Click += new System.EventHandler(CompGameCreate\_Click);

Label SizeInfo = new Label();

SizeInfo.Name = "SizeInfo";

SizeInfo.Location = new Point(175 + 7, 230);

SizeInfo.AutoSize = true;

SizeInfo.Text = "Введите количество\r\nслов";

SizeInfo.Visible = false;

Controls.Add(SizeInfo);

TextBox SizeBox = new TextBox();

SizeBox.Name = "SizeBox";

SizeBox.MaxLength = 2;

SizeBox.Location = new Point(175 + 7, 265);

SizeBox.Visible = false;

SizeBox.Enabled = false;

Controls.Add(SizeBox);

Button StartCompCreate = new Button();

StartCompCreate.Name = "StartCompCreate";

StartCompCreate.Location = new Point(175 + 7, 330);

StartCompCreate.Size = new Size(121, 35);

StartCompCreate.Text = "Начать";

StartCompCreate.Font = new System.Drawing.Font("Segoe UI", 9F, System.Drawing.FontStyle.Bold);

StartCompCreate.Visible = false;

StartCompCreate.Enabled = false;

StartCompCreate.Click += new System.EventHandler(StartCompCreate\_Click);

Controls.Add(StartCompCreate);

}

private void SelfGameCreate\_Click(object sender, EventArgs e)

{

Controls["StartUserCreate"].Visible = true;

Controls["StartUserCreate"].Enabled = true;

Controls["HeightInfo"].Visible = true;

Controls["HeightBox"].Visible = true;

Controls["HeightBox"].Enabled = true;

Controls["SelfGameCreate"].Click -= new System.EventHandler(SelfGameCreate\_Click);

}

private void CompGameCreate\_Click(object sender, EventArgs e)

{

Controls["StartCompCreate"].Visible = true;

Controls["StartCompCreate"].Enabled = true;

Controls["SizeInfo"].Visible = true;

Controls["SizeBox"].Visible = true;

Controls["SizeBox"].Enabled = true;

Controls["CompGameCreate"].Click -= new System.EventHandler(CompGameCreate\_Click);

}

private void StartUserCreate\_Click(object sender, EventArgs e){

if ((Controls["HeightBox"].Text != "" ) &&(Regex.IsMatch(Controls["HeightBox"].Text, @"^\d+$")))

{

UserCreate userCreate = new UserCreate(Convert.ToInt32(Controls["HeightBox"].Text),this);

userCreate.Show();

this.Hide();

}

}

private void StartCompCreate\_Click(object sender, EventArgs e)

{

if ((Controls["SizeBox"].Text != "") && (int.TryParse(Controls["SizeBox"].Text,out \_)))

{

if(Convert.ToInt32(Controls["SizeBox"].Text) <= 20)

{

SubForm test = new SubForm(this, Convert.ToInt32(Controls["SizeBox"].Text));

test.Show();

this.Hide();

}

}

}

private void InfBut\_Click(object sender, EventArgs e){

instructs instr = new instructs(this);

this.Hide();

instr.Show();

}

}

## Приложение А.2. Программный код формы instructs

public partial class instructs : Form

{

MainMenu menu;

public instructs(MainMenu menu)

{

InitializeComponent();

this.menu = menu;

}

private void backBut\_Click(object sender, EventArgs e)

{

menu.Show();

this.Close();

}

}

## Приложение А.3 Программный код формы SubForm

public partial class SubForm : Form

{

MainMenu menu;

int count;

public SubForm(MainMenu menu,int count)

{

InitializeComponent();

this.menu = menu;

this.count = count;

Createboard(count);

}

private void назадToolStripMenuItem\_click(object sender, EventArgs e)

{

this.Close();

menu.Show();

}

private void Createboard(int count)

{

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

{

locBoard.Rows.Add();

}

}

private List<ToAuto> createList()

{

List<ToAuto> list = new List<ToAuto>();

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

{

list.Add(new ToAuto(locBoard.Rows[i].Cells[0].EditedFormattedValue.ToString(), locBoard.Rows[i].Cells[1].EditedFormattedValue.ToString()));

}

return list;

}

private void готовоToolStripMenuItem\_Click(object sender, EventArgs e)

{

List<ToAuto> rnd = createList();

AutoCross NewAuto = new AutoCross(menu, rnd);

this.Close();

NewAuto.Show();

}

}

public struct ToAuto

{

public string word;

public string defin;

public ToAuto(string word,string defin)

{

this.word = word;

this.defin = defin;

}

}

}

## Приложение А.4. Программный код формы UserCreate

public partial class UserCreate : Form

{

public int lastrow = 0;

public int lastcol = 0;

int boardsize;

public List<crossW> Crosslist = new List<crossW>();

char[,] gamechars;

int[,] numbers;

int helpcount = 0;

MainMenu menu;

public UserCreate()

{

InitializeComponent();

}

public UserCreate(int boardsize, MainMenu menu)

{

this.boardsize = boardsize;

InitializeComponent();

CreateBoard();

this.menu = menu;

}

public void CreateBoard()

{

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

{

board.Columns.Add("","");

}

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

{

board.Rows.Add();

}

foreach (DataGridViewColumn c in board.Columns)

c.Width = board.Width / board.Columns.Count;

foreach(DataGridViewRow c in board.Rows)

c.Height = board.Height / board.Rows.Count;

закончитьКроссвордToolStripMenuItem.Enabled = false;

закончитьКроссвордToolStripMenuItem.Visible = false;

подсказкаToolStripMenuItem.Enabled = false;

подсказкаToolStripMenuItem.Visible = false;

}

public void CreateGameBoard()

{

CreateBoard();

GameCharsNnumbersfill();

board.ReadOnly = false;

board.CellDoubleClick -= new System.Windows.Forms.DataGridViewCellEventHandler(board\_CellDoubleClick);

закончитьКроссвордToolStripMenuItem.Enabled = true;

закончитьКроссвордToolStripMenuItem.Visible = true;

закончитьToolStripMenuItem.Visible = false;

закончитьToolStripMenuItem.Enabled = false;

зановоToolStripMenuItem.Visible = false;

зановоToolStripMenuItem.Enabled = false;

посмотретьСписокСловToolStripMenuItem.Enabled = false;

посмотретьСписокСловToolStripMenuItem.Visible = false;

подсказкаToolStripMenuItem.Enabled = true;

подсказкаToolStripMenuItem.Visible = true;

board.EditingControlShowing += new System.Windows.Forms.DataGridViewEditingControlShowingEventHandler(board\_EditingControlShowing);

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

{

for (int j = 0; j < boardsize; j++)

{

if (gamechars[i, j] == '0')

{

board[j,i].ReadOnly = true;

board[j,i].Style.BackColor = Color.Black;

}

if (numbers[i, j] != 0)

{

board.Rows[j].Cells[i].Value = numbers[i, j];

board.Rows[j].Cells[i].Style.BackColor = Color.Cornsilk;

board.Rows[j].Cells[i].ToolTipText = numbers[i, j].ToString();

}

}

}

DefTable deftable = new DefTable(Crosslist);

deftable.SetDesktopLocation(this.Location.X + this.Width + 1, this.Location.Y);

deftable.menuStrip1.Visible = false;

deftable.menuStrip1.Enabled = false;

deftable.Show();

}

private void GameCharsNnumbersfill()

{

gamechars = new char[boardsize, boardsize];

numbers = new int[boardsize, boardsize];

for (int k = 0; k < boardsize; k++)

for (int n = 0; n < boardsize; n++) { gamechars[k, n] = '0'; numbers[k, n] = 0; }

foreach(crossW elem in Crosslist)

{

numbers[elem.Xstart,elem.Ystart] = Crosslist.IndexOf(elem)+1;

for(int i=0;i < elem.wordL+1; i++)

{

if (elem.vert) gamechars[elem.Ystart + i, elem.Xstart] = elem.word[i];

else gamechars[elem.Ystart, elem.Xstart + i] = elem.word[i];

}

}

}

private void test()

{

Label test = new Label();

test.Location = new Point(10, 10);

test.Size = new Size(500, 500);

String test2 = "";

for(int i = 0; i < 21; i++)

{

test2 += "\n";

for(int j = 0; j < 21; j++) test2 += gamechars[i,j].ToString()+" ";

}

test.Text = test2;

this.Controls.Add(test);

test.BringToFront();

}

private void board\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if ((e.RowIndex == lastrow || e.ColumnIndex == lastcol)

&& ((board.Rows[e.RowIndex].Cells[e.ColumnIndex].Style.BackColor == Color.Cornsilk)

&& (board.Rows[lastrow].Cells[lastcol].Style.BackColor == Color.Cornsilk)) == false

/\* && collision(e.RowIndex, lastrow, e.ColumnIndex, lastcol)\*/)

{

ChooseWordZone(e.RowIndex, lastrow, e.ColumnIndex, lastcol);

WordWriteDown(Crosslist[Crosslist.Count-1]);

ClearCellCross(lastrow, lastcol);

lastrow = e.RowIndex;

lastcol = e.ColumnIndex;

}

else

{

ClearCellCross(lastrow, lastcol);

CreateCellCross(e.RowIndex, e.ColumnIndex);

lastrow = e.RowIndex;

lastcol = e.ColumnIndex;

}

}

private void CreateCellCross(int r, int c)

{

//board.Rows[r].Cells[c].Style.BackColor = Color.Black;

int counter = r-1;

while(counter >= 0)

{

if (board.Rows[counter].Cells[c].Style.BackColor != Color.Cornsilk)

board.Rows[counter].Cells[c].Style.BackColor = Color.Gray;

counter--;

}

counter = r+1;

while (counter < board.Rows.Count)

{

if (board.Rows[counter].Cells[c].Style.BackColor != Color.Cornsilk)

board.Rows[counter].Cells[c].Style.BackColor = Color.Gray;

counter++;

}

counter = c - 1;

while (counter >= 0)

{

if (board.Rows[r].Cells[counter].Style.BackColor != Color.Cornsilk)

board.Rows[r].Cells[counter].Style.BackColor = Color.Gray;

counter--;

}

counter = c + 1;

while (counter < board.Columns.Count)

{

if (board.Rows[r].Cells[counter].Style.BackColor != Color.Cornsilk)

board.Rows[r].Cells[counter].Style.BackColor = Color.Gray;

counter++;

}

}

private void ClearCellCross(int r,int c)

{

if (board.Rows[r].Cells[c].Style.BackColor != Color.Cornsilk)

board.Rows[r].Cells[c].Style.BackColor = Color.White;

int counter = r - 1;

while (counter >= 0)

{

if (board.Rows[counter].Cells[c].Style.BackColor != Color.Cornsilk)

board.Rows[counter].Cells[c].Style.BackColor = Color.White;

counter--;

}

counter = r + 1;

while (counter < board.Rows.Count)

{

if (board.Rows[counter].Cells[c].Style.BackColor != Color.Cornsilk)

board.Rows[counter].Cells[c].Style.BackColor = Color.White;

counter++;

}

counter = c - 1;

while (counter >= 0)

{

if (board.Rows[r].Cells[counter].Style.BackColor != Color.Cornsilk)

board.Rows[r].Cells[counter].Style.BackColor = Color.White;

counter--;

}

counter = c + 1;

while (counter < board.Columns.Count)

{

if (board.Rows[r].Cells[counter].Style.BackColor != Color.Cornsilk)

board.Rows[r].Cells[counter].Style.BackColor = Color.White;

counter++;

}

}

private void ChooseWordZone(int rn,int rl,int cn,int cl)

{

if(cn == cl)

{

if (rn > rl)

{

for (int i = rl; i < rn + 1; i++)

{

board.Rows[i].Cells[cn].Style.BackColor = Color.Aqua;

}

}

else for (int i = rn; i < rl + 1; i++)

{

board.Rows[i].Cells[cn].Style.BackColor = Color.Aqua;

}

}

else

{

if (cn > cl)

{

for (int i = cl; i < cn + 1; i++)

{

board.Rows[rn].Cells[i].Style.BackColor = Color.Aqua;

}

}

else for (int i = cn; i < cl+1; i++)

{

board.Rows[rn].Cells[i].Style.BackColor = Color.Aqua;

}

}

int wordlenght;

bool vert = false;

if (cn == cl) vert = true;

else vert = false;

if (vert) wordlenght = Math.Max(rn, rl) - Math.Min(rn, rl);

else wordlenght = Math.Max(cn, cl) - Math.Min(cn, cl);

Crosslist.Add(new crossW(Math.Min(cl,cn),Math.Min(rn,rl),wordlenght,vert));

Crosshelp secwind = new Crosshelp(Crosslist,this.board);

secwind.FormClosing += (sender1, e1) =>

{

Crosslist[Crosslist.Count -1].word = secwind.legoword;

Crosslist[Crosslist.Count-1].wordD = secwind.oprtext.Text;

};

secwind.ShowDialog();

}

public void WordWriteDown(crossW joja)

{

int counter;

if (joja.vert)

{

counter = joja.Ystart;

foreach (var chr in joja.word)

{

board.Rows[counter].Cells[joja.Xstart].Value = chr;

board.Rows[counter].Cells[joja.Xstart].Style.BackColor = Color.Cornsilk;

counter++;

}

}

else

{

counter = joja.Xstart;

foreach (var chr in joja.word)

{

board.Rows[joja.Ystart].Cells[counter].Value = chr;

board.Rows[joja.Ystart].Cells[counter].Style.BackColor = Color.Cornsilk;

counter++;

}

}

board.Rows[joja.Ystart].Cells[joja.Xstart].Value = (Crosslist.IndexOf(joja)+1).ToString() + "|" + board.Rows[joja.Ystart].Cells[joja.Xstart].Value;

}

private void посмотретьСписокСловToolStripMenuItem\_Click(object sender, EventArgs e)

{

DefTable deftable = new DefTable(Crosslist);

deftable.SetDesktopLocation(this.Location.X+this.Width+1, this.Location.Y);

DefTable.defboardupdate handler = new DefTable.defboardupdate(Defhelpfunc);

deftable.update += handler;

deftable.Show();

}

private void Defhelpfunc(object sender, EventArgs e)

{

посмотретьСписокСловToolStripMenuItem.PerformClick();

}

private void зановоToolStripMenuItem\_Click(object sender, EventArgs e)

{

Crosslist = new List<crossW>();

lastrow = 0;

lastcol = 0;

board.Rows.Clear();

board.Columns.Clear();

CreateBoard();

}

private void закончитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

board.Rows.Clear();

board.Columns.Clear();

CreateGameBoard();

}

private void назадToolStripMenuItem\_Click(object sender, EventArgs e)

{

menu.Show();

this.Close();

}

private void board\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

if (e.Control is TextBox)

{

(e.Control as TextBox).MaxLength = 1;

}

}

private int GameEnd()

{

int CorCount = 0;

foreach (crossW elem in Crosslist)

{

bool wordCorrect = true;

if (elem.vert)

{

for(int i = 0;i < elem.wordL; i++)

{

if (board[elem.Xstart, elem.Ystart + i].EditedFormattedValue == null)

{

wordCorrect = false;

break;

}

else if (board[elem.Xstart , elem.Ystart+i].EditedFormattedValue.ToString() != gamechars[elem.Ystart+i, elem.Xstart ].ToString())

{

wordCorrect=false;

break;

}

}

}

else

{

for (int i = 0; i < elem.wordL; i++)

{

if (board[elem.Xstart+i, elem.Ystart].EditedFormattedValue == null)

{

wordCorrect = false;

break;

}

else if (board[elem.Xstart +i , elem.Ystart ].EditedFormattedValue.ToString() != gamechars[elem.Ystart, elem.Xstart+i].ToString())

{

wordCorrect = false;

break;

}

}

}

if (wordCorrect) CorCount++;

}

return CorCount; private void подсказкаToolStripMenuItem\_Click(object sender, EventArgs e)

{

board.CellDoubleClick += new System.Windows.Forms.DataGridViewCellEventHandler(Gameboard\_CellDoubleClick);

}

private void Gameboard\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (gamechars[e.RowIndex, e.ColumnIndex] != '0')

{

board[e.ColumnIndex,e.RowIndex].Value = gamechars[e.RowIndex, e.ColumnIndex].ToString();

board[e.ColumnIndex, e.RowIndex].Style.BackColor = Color.Red;

board[e.ColumnIndex, e.RowIndex].ReadOnly = true;

board.CellDoubleClick -= new System.Windows.Forms.DataGridViewCellEventHandler(Gameboard\_CellDoubleClick);

helpcount++;

}

}

private void закончитьКроссвордToolStripMenuItem\_Click(object sender, EventArgs e)

{

Results newResultsWin = new Results( helpcount, GameEnd(), Crosslist.Count, menu);

newResultsWin.Show();

this.Close();

}

}

public class crossW

{

public int Xstart;

public int Ystart;

public int wordL;

public bool vert;

public string word;

public string wordD;

public crossW(int xstart, int ystart, int wordL, bool vert)

{

Xstart = xstart;

Ystart = ystart;

this.wordL = wordL;

this.vert = vert;

}

}

## Приложение А.5. Программный код формы Crosshelp

public partial class Crosshelp : Form

{

public string legoword;

private bool vertical;

private int worL;

public Crosshelp()

{

InitializeComponent();

}

public Crosshelp(List<crossW> Crosslist,DataGridView main)

{

InitializeComponent();

helpcreated(Crosslist[Crosslist.Count-1],main);

}

private void helpcreated(crossW joja, DataGridView main)

{

vertical = joja.vert;

worL = joja.wordL;

if (joja.vert)

{

helpboard.Columns.Add("", "");

for (int i = 0; i < joja.wordL+1; i++) helpboard.Rows.Add();

foreach (DataGridViewColumn c in helpboard.Columns)

c.Width = 30;

for (int i = 0; i < joja.wordL+1; i++)

{

if (main.Rows[joja.Ystart + i].Cells[joja.Xstart].Value != null)

{

if (main.Rows[joja.Ystart + i].Cells[joja.Xstart].Value.ToString().Length == 3)

{

helpboard.Rows[i].Cells[0].Value = main.Rows[joja.Ystart + i].Cells[joja.Xstart].Value.ToString()[2];

helpboard.Rows[i].Cells[0].ReadOnly = true;

}

else

{

helpboard.Rows[i].Cells[0].Value = main.Rows[joja.Ystart + i].Cells[joja.Xstart].Value;

helpboard.Rows[i].Cells[0].ReadOnly = true;

}

}

}

}

else

{

for (int i = 0; i < joja.wordL+1; i++) helpboard.Columns.Add("", "");

helpboard.Rows.Add();

foreach (DataGridViewColumn c in helpboard.Columns)

c.Width = 30;

for (int i = 0; i < joja.wordL + 1; i++)

{

if (main.Rows[joja.Ystart].Cells[joja.Xstart+i].Value != null)

{

if (main.Rows[joja.Ystart].Cells[joja.Xstart + i].Value.ToString().Length == 3)

{

helpboard.Rows[0].Cells[i].Value = main.Rows[joja.Ystart].Cells[joja.Xstart + i].Value.ToString()[2];

helpboard.Rows[0].Cells[i].ReadOnly = true;

}

else

{

helpboard.Rows[0].Cells[i].Value = main.Rows[joja.Ystart].Cells[joja.Xstart + i].Value;

helpboard.Rows[0].Cells[i].ReadOnly = true;

}

}

}

}

helpboard.AllowUserToAddRows = false;

}

private void button1\_Click(object sender, EventArgs e)

{

bool closing = true;

legoword = "";

if (vertical)

{

for (int i = 0; i < worL + 1; i++)

{

if (helpboard.Rows[i].Cells[0].Value == null) closing = false;

else legoword += helpboard.Rows[i].Cells[0].Value.ToString();

}

}

else for (int i = 0; i < worL + 1; i++)

{

if (helpboard.Rows[0].Cells[i].Value == null) closing = false;

else legoword += helpboard.Rows[0].Cells[i].Value.ToString();

}

if(legoword.Length == worL+1 && closing)

this.Close();

}

private void helpboard\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

if (e.Control is TextBox)

{

(e.Control as TextBox).MaxLength = 1;

}

}

}

## Приложение А.6. Программный код формы Deftable

public partial class DefTable : Form

{

public delegate void defboardupdate(object sender, EventArgs e);

public event defboardupdate update;

public DefTable()

{

InitializeComponent();

}

public DefTable(List<crossW> wordList)

{

InitializeComponent();

ColumnsStyle();

Defshow(wordList);

}

public DefTable(List<SentToDefTb> wordlist)

{

InitializeComponent();

ColumnsStyle();

Defshow(wordlist);

}

private void ColumnsStyle()

{

DefinTable.Columns[0].Width = 50;

DefinTable.Columns[1].Width = 100;

DefinTable.Columns[2].Width = 181+50;

}

private void обновитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (update != null)

{

update(this, EventArgs.Empty);

}

this.Close();

}

private void Defshow(List<crossW> Wordsndefs)

{

DefinTable.Rows.Clear();

for(int i = 0; i < Wordsndefs.Count; i++)

{

DefinTable.Rows.Add();

DefinTable.Rows[i].Cells[0].Value = i+1;

if (Wordsndefs[i].vert) DefinTable.Rows[i].Cells[1].Value = "Вертикальное";

else DefinTable.Rows[i].Cells[1].Value = "Горизонтальное";

DefinTable.Rows[i].Cells[2].Value = Wordsndefs[i].wordD;

}

}

private void Defshow(List<SentToDefTb> wordlist)

{

DefinTable.Rows.Clear();

for (int i = 0; i < wordlist.Count; i++)

{

DefinTable.Rows.Add();

DefinTable.Rows[i].Cells[0].Value = i + 1;

if (wordlist[i].vert) DefinTable.Rows[i].Cells[1].Value = "Вертикальное";

else DefinTable.Rows[i].Cells[1].Value = "Горизонтальное";

DefinTable.Rows[i].Cells[2].Value = wordlist[i].def;

}

}

}

## Приложение А.7. Программный код формы AutoCreate

public partial class AutoCross : Form

{

List<string> words = new List<string>();

List<HelpWithWords> help = new List<HelpWithWords>();

List<doubled> db = new List<doubled>();

List<cell> CurWordCells = new List<cell>();

cell[,] gameChars = new cell[20, 20];

int[,] numbers = new int[20, 20];

List<ToAuto> original;

string CurWord;

Random rnd = new Random();

List<ChoosingBest> xxx = new List<ChoosingBest>();

int helpcount;

int size = 20;

MainMenu menu;

public AutoCross(MainMenu menu, List<ToAuto> original)

{

InitializeComponent();

this.menu = menu;

this.original = original;

// test

Fill();

CrossGenerate();

test();

CreateBoard();

DefTable deftable = new DefTable(sentToDefTbs());

deftable.SetDesktopLocation(this.Location.X + this.Width + 1, this.Location.Y);

deftable.menuStrip1.Visible = false;

deftable.menuStrip1.Enabled = false;

deftable.Show();

}

private void CreateBoard()

{

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

{

board.Columns.Add("", "");

}

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

{

board.Rows.Add();

}

foreach (DataGridViewColumn c in board.Columns)

c.Width = board.Width / board.Columns.Count;

foreach (DataGridViewRow c in board.Rows)

c.Height = board.Height / board.Rows.Count;

NumbFill();

board.EditingControlShowing += new System.Windows.Forms.DataGridViewEditingControlShowingEventHandler(board\_EditingControlShowing);

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

{

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

{

if (gameChars[i, j].data == '0')

{

board[j, i].ReadOnly = true;

board[j, i].Style.BackColor = Color.Black;

}

if (numbers[i, j] != 0)

{

board.Rows[j].Cells[i].Value = numbers[i, j];

board.Rows[j].Cells[i].Style.BackColor = Color.Cornsilk;

board.Rows[j].Cells[i].ToolTipText = numbers[i, j].ToString();

}

}

}

foreach (var elem in db)

{

board.Rows[elem.y].Cells[elem.x].Value = elem.first.ToString() + "|" + elem.second.ToString();

}

}

private void board\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

if (e.Control is TextBox)

{

(e.Control as TextBox).MaxLength = 1;

}

}

private void Fill()

{

int size = 20;

foreach (var elem in original)

{

words.Add(elem.word);

}

gameChars = new cell[size, size];

numbers = new int[size, size];

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

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

{

gameChars[y, x] = new cell();

gameChars[y, x].data = '0';

gameChars[y, x].y = y;

gameChars[y, x].x = x;

}

}

private void CrossGenerate()

{

//Случайно ставим первое слово

CurWord = words[rnd.Next(words.Count)];

for (int i = 0; i < CurWord.Length; i++)

{

gameChars[(gameChars.GetLength(0) / 2), (gameChars.GetLength(0) / 2) - (CurWord.Length / 2) + i].data = CurWord[i];

gameChars[(gameChars.GetLength(0) / 2), (gameChars.GetLength(0) / 2) - (CurWord.Length / 2) + i].state = cell.State.Hor;

CurWordCells.Add(gameChars[(gameChars.GetLength(0) / 2), (gameChars.GetLength(0) / 2) - (CurWord.Length / 2) + i]);

}

help.Add(new HelpWithWords(CurWord, gameChars[(gameChars.GetLength(0) / 2), (gameChars.GetLength(0) / 2) - (CurWord.Length / 2)]));

words.Remove(CurWord);

//Подбираем слова по буквам

//На одно слово ставим максимум два слова

func(help[0]);

//Доставляем слова

/\*

if (words.Count != 0)

{

int i = 0;

int losscount = 0;

while (true)

{

bool done = false;

FillCurCellList(help[i]);

foreach(cell elem in CurWordCells)

{

foreach(string word in words)

{

if(word.Contains(elem.data) && elem.state != cell.State.Both &&

collision(elem,word))

{

wordFill(elem, word);

words.Remove(word);

done = true;

break;

}

}

}

if(!done) losscount++;

if (losscount == help.Count) break;

if (i == help.Count - 1)

{

i = 0;

losscount = 0;

}

else i++;

}

}

\*/

}

private void wordFill(cell joja, string Curword)

{

if (joja.state == cell.State.Vert)

{

for (int i = 0; i < Curword.Length; i++)

{

gameChars[joja.y, joja.x - Curword.IndexOf(joja.data) + i].data = Curword[i];

gameChars[joja.y, joja.x - Curword.IndexOf(joja.data) + i].y = joja.y;

gameChars[joja.y, joja.x - Curword.IndexOf(joja.data) + i].x = joja.x - Curword.IndexOf(joja.data) + i;

gameChars[joja.y, joja.x - Curword.IndexOf(joja.data) + i].state = cell.State.Hor;

}

help.Add(new HelpWithWords(Curword, gameChars[joja.y, joja.x - Curword.IndexOf(joja.data)]));

}

else

{

for (int i = 0; i < Curword.Length; i++)

{

gameChars[joja.y - Curword.IndexOf(joja.data) + i, joja.x].data = Curword[i];

gameChars[joja.y - Curword.IndexOf(joja.data) + i, joja.x].y = joja.y - Curword.IndexOf(joja.data) + i;

gameChars[joja.y - Curword.IndexOf(joja.data) + i, joja.x].x = joja.x;

gameChars[joja.y - Curword.IndexOf(joja.data) + i, joja.x].state = cell.State.Vert;

}

help.Add(new HelpWithWords(Curword, gameChars[joja.y - Curword.IndexOf(joja.data), joja.x]));

}

joja.state = cell.State.Both;

}

private List<cell> FillCurCellList(HelpWithWords data)

{

List<cell> buf = new List<cell>();

if(data.startCell.state == cell.State.Both)

{

if(gameChars[data.startCell.y + 1, data.startCell.x].data == data.word[1] && gameChars[data.startCell.y + 2, data.startCell.x].data == data.word[2])

{

for (int i = 0; i < data.word.Length; i++) buf.Add(gameChars[data.startCell.y + i, data.startCell.x]);

}

else for (int i = 0; i < data.word.Length; i++) buf.Add(gameChars[data.startCell.y, data.startCell.x + i]);

}

else if(data.startCell.state == cell.State.Vert)

{

for (int i = 0; i < data.word.Length; i++) buf.Add(gameChars[data.startCell.y+i,data.startCell.x]);

}

else

{

for (int i = 0; i < data.word.Length; i++) buf.Add(gameChars[data.startCell.y, data.startCell.x+i]);

}

return buf;

}

private void test()

{

Label test = new Label();

test.Location = new Point(10, 10);

test.Size = new Size(500, 500);

for (int i = 0; i < 20; i++)

{

test.Text += "\r\n";

for (int j = 0; j < 20; j++)

test.Text += gameChars[i,j].data.ToString()+" ";

}

this.Controls.Add(test);

}

private bool collision(cell joja, string Curword)

{

bool noword = true;

if (joja.state == cell.State.Vert)

{

if ((joja.x - Curword.IndexOf(joja.data)) < 0 ||

(joja.x - Curword.IndexOf(joja.data)) + Curword.Length > gameChars.GetLength(0)) return false;

for (int i = 0; i < Curword.Length; i++)

{

if (joja.x != joja.x - Curword.IndexOf(joja.data) + i)

{

if (gameChars[joja.y, joja.x - Curword.IndexOf(joja.data) + i].data != '0')

{

noword = false;

break;

}

}

}

}

else

{

if ((joja.y - Curword.IndexOf(joja.data)) < 0 ||

(joja.y - Curword.IndexOf(joja.data)) + Curword.Length > gameChars.GetLength(0)) return false;

for (int i = 0; i < Curword.Length; i++)

{

if (joja.y != joja.y - Curword.IndexOf(joja.data) + i)

{

if (gameChars[joja.y - Curword.IndexOf(joja.data) + i, joja.x].data != '0')

{

noword = false;

break;

}

}

}

}

return noword;

}

private int BestPosCheck()

{

int up = -1;

int down = -1;

int left = -1;

int right = -1;

bool gotcha;

for (int y = 0; y < 10; y++)

{

gotcha = false;

for (int x = 0; x < 10; x++)

{

if (gameChars[y, x].data != 'o')

{

up = y;

gotcha = true;

break;

}

}

if (gotcha) break;

}

for (int y = 9; y > 0; y--)

{

gotcha = false;

for (int x = 0; x < 10; x++)

{

if (gameChars[y, x].data != 'o')

{

down = y;

gotcha = true;

break;

}

}

if (gotcha) break;

}

for (int x = 0; x < 10; x++)

{

gotcha = false;

for (int y = 0; y < 10; y++)

{

if (gameChars[y, x].data != 'o')

{

left = x;

gotcha = true;

break;

}

}

if (gotcha) break;

}

for (int x = 9; x > 0; x--)

{

gotcha = false;

for (int y = 0; y < 10; y++)

{

if (gameChars[y, x].data != 'o')

{

right = x;

gotcha = true;

break;

}

}

if (gotcha) break;

}

int ScoreSize = (right - left + 1) \* (down - up + 1);

return ScoreSize;

}

private void func(HelpWithWords data)

{

if (words.Count == 0) return;

CurWordCells = FillCurCellList(data);

//Создаём posWords

List<HelpWithWords> PossibleWords = new List<HelpWithWords>();

foreach (var elem in CurWordCells)

{

if (elem.state != cell.State.Both)

{

for (int j = 0; j < words.Count; j++)

{

if (words[j].Contains(elem.data) && !haveIn(PossibleWords, words[j])

&& collision(elem, words[j]))

{

PossibleWords.Add(new HelpWithWords(words[j],elem));

break;

}

}

}

}

if(PossibleWords.Count >= 2)

{

HelpWithWords n1 = PossibleWords[rnd.Next(PossibleWords.Count - 1)];

if (haveIn(help, n1.word)) return;

PossibleWords.Remove(n1);

words.Remove(n1.word);

wordFill(n1.startCell, n1.word);

func(help[help.Count-1]);

HelpWithWords n2 = PossibleWords[rnd.Next(PossibleWords.Count - 1)];

if (haveIn(help, n2.word)) return;

PossibleWords.Remove(n2);

words.Remove(n2.word);

wordFill(n2.startCell, n2.word);

func(help[help.Count - 1]);

}

else if (PossibleWords.Count == 1)

{

HelpWithWords n1 = PossibleWords[rnd.Next(PossibleWords.Count - 1)];

if (haveIn(help, n1.word)) return;

PossibleWords.Remove(n1);

words.Remove(n1.word);

wordFill(n1.startCell, n1.word);

func(help[help.Count - 1]);

}

}

private bool haveIn(List<HelpWithWords> data, string line)

{

bool ans = false;

foreach (var elem in data)

{

if (elem.word == line)

{

ans = true;

break;

}

}

return ans;

}

private List<SentToDefTb> sentToDefTbs()

{

List<SentToDefTb> ans = new List<SentToDefTb>();

for (int i = 0; i < help.Count; i++)

{

bool vert = false;

if (help[i].startCell.state != cell.State.Both)

{

if (help[i].startCell.state == cell.State.Vert) vert = true;

else vert = false;

}

else

{

if (gameChars[help[i].startCell.y + 1, help[i].startCell.x].data == help[i].word[1]) vert = true;

else vert = false;

}

string def = " ";

foreach(var elem in original)

{

if (help[i].word == elem.word)

{

def = elem.defin;

break;

}

}

ans.Add(new SentToDefTb(vert, help[i].word, i + 1,def));

}

return ans;

}

private void NumbFill()

{

for(int i = 0; i <help.Count; i++)

{

if (numbers[help[i].startCell.x, help[i].startCell.y] != 0)

{ db.Add(new doubled(help[i].startCell.y, help[i].startCell.x, numbers[help[i].startCell.x, help[i].startCell.y], i + 1)); }

numbers[help[i].startCell.x, help[i].startCell.y] = i + 1;

}

}

private void подсказкаToolStripMenuItem\_Click(object sender, EventArgs e)

{

board.CellDoubleClick += new System.Windows.Forms.DataGridViewCellEventHandler(Gameboard\_CellDoubleClick);

}

private void Gameboard\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (gameChars[e.RowIndex, e.ColumnIndex].data != 'o')

{

board[e.ColumnIndex, e.RowIndex].Value = gameChars[e.RowIndex, e.ColumnIndex].data.ToString();

board[e.ColumnIndex, e.RowIndex].Style.BackColor = Color.Red;

board[e.ColumnIndex, e.RowIndex].ReadOnly = true;

board.CellDoubleClick -= new System.Windows.Forms.DataGridViewCellEventHandler(Gameboard\_CellDoubleClick);

helpcount++;

}

}

private void закончитьКроссвордToolStripMenuItem\_Click(object sender, EventArgs e)

{

Results newResultsWin = new Results(helpcount, ResultsOutPut(), help.Count, menu);

newResultsWin.Show();

this.Close();

}

private void назадToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Close();

menu.Show();

}

private int ResultsOutPut()

{

int ans = 0;

foreach(var elem in help)

{

CurWordCells = FillCurCellList(elem);

bool got = true;

foreach(cell data in CurWordCells)

{

if(data.data.ToString() != board[data.x,data.y].EditedFormattedValue.ToString())

{

got = false;

break;

}

}

if (got) ans++;

}

return ans;

}

}

public class cell

{

public enum State

{

None,

Vert,

Hor,

Both,

}

public char data;

public int x;

public int y;

public State state = State.None;

}

public struct HelpWithWords

{

public string word;

public cell startCell;

public HelpWithWords(string word, cell startCell)

{

this.word = word;

this.startCell = startCell;

}

}

public struct ChoosingBest

{

public cell[,] cells;

public int wordcount;

public int size;

public ChoosingBest(cell[,] cells,int wordcount,int size)

{

this.cells = cells;

this.wordcount = wordcount;

this.size = size;

}

}

public class SentToDefTb

{

public bool vert;

public string word;

public int number;

public string def;

public SentToDefTb(bool vert,string word,int number,string def)

{

this.vert = vert;

this.word = word;

this.number = number;

this.def = def;

}

}

public struct doubled

{

public int y;

public int x;

public int first;

public int second;

public doubled(int y,int x,int first,int second)

{

this.y = y;

this.x = x;

this.first = first;

this.second = second;

}

}

## Приложение А.8. Программный код формы Results

public partial class Results : Form

{

MainMenu menu;

public Results()

{

InitializeComponent();

}

public Results(int helpCount,int wordCount,int listlen,MainMenu prevwind)

{

menu = prevwind;

InitializeComponent();

filler(helpCount,wordCount,listlen);

}

private void filler(int helpCount, int wordcount,int listlen)

{

HelpCount.Text = helpCount.ToString();

wordCount.Text = $"{wordcount}/{listlen}";

}

private void AnatherBackBut\_Click(object sender, EventArgs e)

{

menu.Show();

this.Close();

}

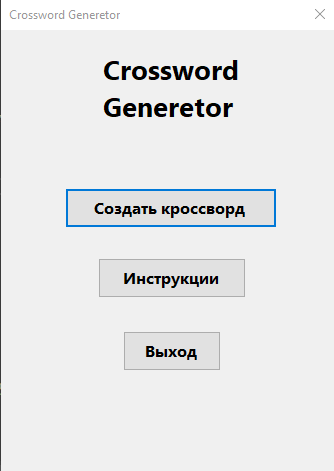
## Приложение A.9. Программный код файла Programs.cs

ApplicationConfiguration.Initialize();

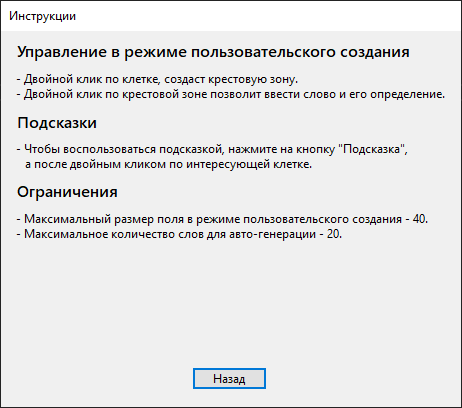
Application.Run(new MainMenu());

# Приложение Б. Интерфейс приложения

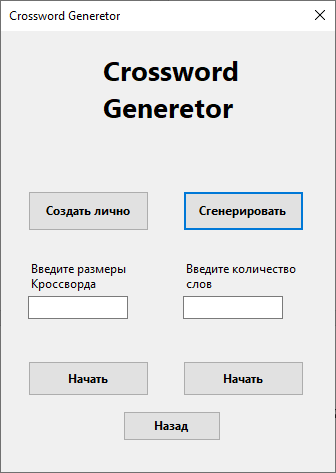
## Приложение Б.1. Окно меню



## Приложение Б.2. Инструкции



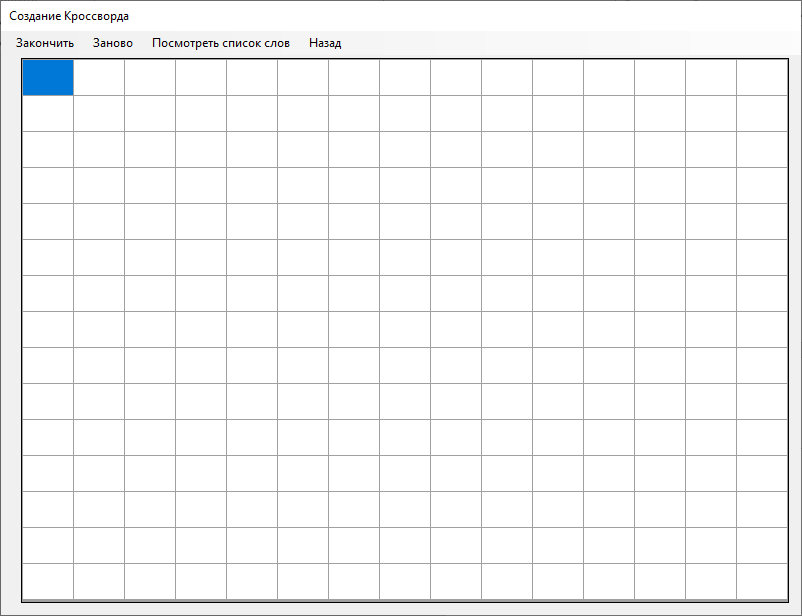
## Приложение Б.3. Меню выбора режима



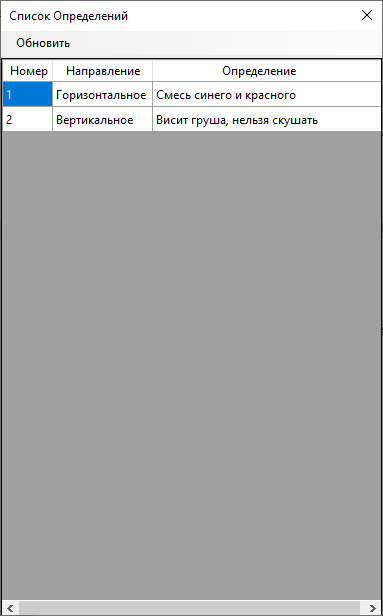
## Приложение Б.4. Окно SubForm



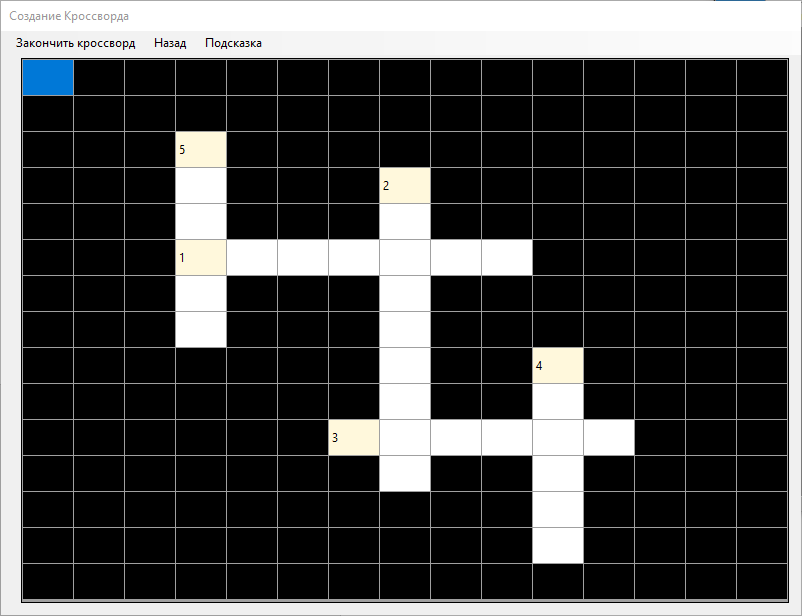
## Приложение Б.5. Окно пользовательского создания



## Приложение Б.6. Окно DefTable



## Приложение Б.7. Окно решения кроссворда



## Приложение Б.8. Окно итогов

