Частное учреждение образования

«Колледж бизнеса и права»

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА СОРТИРОВКИ И БИНАРНОГО ПОИСКА В МАССИВЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Основы алгоритмизации и программирование»

КП Т.0910006.401

Руководитель проекта (Ю.В.Шаляпин)

Учащийся (А.О.Зеневич)

2022

Частное учреждение образования

«Колледж бизнеса и права»

**Отделение** ПОИТ

**Специальность** 2-40-01 01 «Программное обеспечение информационных технологий»

**Группа** Т-091

**ЗАДАНИЕ**

на курсовое проектирование

Учащемуся (-ейся) Зеневичу Александру Олеговичу

1. Тема проекта: Программы для сортировки и бинарного поиска в массиве

2. Срок сдачи законченного проекта: **6 июня 2022 года**

3. Основные функциональные требования к проекту: разработать на языке программирования C++ в среде разработки MS Visual Studio программу для сортировки и бинарного поиска в массиве.

Реализовать: авторизацию и аутентификацию пользователей (роли: администратор, пользователь), локализацию на английском, русском и белорусском языке; возможность выбора цвета фона окна приложения из 3-х вариантов; ввод пользователя данных о размере массивов, ввод их содержимого и обеспечить возможность рандомного заполнения массивов. Реализовать сортировку Хоара и бинарный поиск в таблице символов, основанной на двумерном массиве. Визуализировать работу алгоритма бинарного поиске в виде деревьев для массивов размерами N=17 и N=24. В программе визуализировать массивы и содержимое файлов, обеспечить возможность чтения и записи файлов.

Обеспечить организацию лаконичного и понятного пользователя интерфейса программы средствами создания оконных форм с элементами меню для пользователя. Создать справочную систему приложения и установочных диск. Программа должна сохранять данные в файл (-ы) формата \*.docx, \*.txt и (-или) \*.bin, читать из них данные.

**Состав проекта:**

**А) Расчётно-пояснительная записка (перечень подлежащих разработке вопросов):**

Введение. 1. Постановка задачи. 2. Вычислительная система. 3. Проектирование задачи. 4. Описание программного средства. 5. Методика испытаний. 6. Применение. Заключение. Литература. Приложения (листинги программных модулей, копии экранных форм, отчёты и др.)

**Б) Графическая часть проекта:**

Лист 1. Блок-схема работы алгоритма сортировки Хоара

Лист 2. Блок-схема работы программы (общая)

**5. Календарный график работы на весь период проектирования:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Постановка задачи, определение вычислительной системы | | | 19.04 – 24.04 |
| 2 | Объектно-ориентированный анализ задачи, проектирование задачи | | | 25.04 – 01.05 |
| 3 | Описание программного средства | | | 02.05 – 22.05 |
| 4 | Методика испытаний (функциональное тестирование) | | | 23.05 – 27.05 |
| 5 | Применение | | | 28.05 – 31.05 |
| 6 | Оформление пояснительной записки | | | 01.06 – 05.06 |
| Председатель цикловой комиссии ПОИТ №10 | | | В.Ю.Михалевич | |
| Руководитель проекта | | | Ю.В.Шаляпин | |
| Дата выдачи задания «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 года | | | | |
| Задания принял к исполнению, | | | | |
| «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 года | | Подпись учащегося: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Датаа*

*Лист*

*3*

*КП Т.091*

*Разраб.*

*Зеневич.А.О*

*Провер.*

*Шаляпин.Ю.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Программная реализация программного «Помощник Прораба»*

*Лит.*

*Листов*

*КБиП*

*74*

*у*

# Введение

В ходе выполнении курсового проекта было задействовано множество форумов, сайтов, интернет-ресурсов, справочных материалов, методических указаний.

В первом разделе «Объектно-ориентированный анализ и проектирование системы»

описывается сущность поставленной задачи, предметная область, список задач, которые должны быть автоматизированы за счёт программного средства. Так же перечисляются основные функции программы, выполняется построение информационной модели, которая, в свою очередь, отражает сущности задач вместе с их свойствами.

Во втором разделе «Вычислительная система» описывается аппаратные и операционные ресурсы компьютера, на котором разрабатывалось программное средство.

В третьем разделе «Проектирование задачи» перечисляются основные требования к программному средству, логическую и физическую структуры данных в контексте среды разработки, структуру и описание функций пользователя в рамках среды разрабатываемого программного средства, функции и элементы управления, проектирование справочной системы программного средства.

В четвертом разделе «Описание программного средства» будут перечислены способы реализации функциональной части приложения и её функциональное назначение.

В пятом разделе «Методика испытаний» описывается перечень требований к техническим средствам для проведения тестирования, требования к информационной и программной совместимости. Представляются результаты функционального тестирования.

В шестом разделе «Применение» происходит описание сведений о назначении программного средства и области его применения, ограничениях. Так же описываются требования к техническим и программным ресурсам. Именно в этом разделе приводится структура справочной системы, а также методика её использования.

В заключении будут представлены выводы о созданном программном приложении, определена степень соответствия поставленной задачи и выполненной работы.

На практике бинарные деревья применяются, когда в каждой точке какого-нибудь вычислительного процесса нужно принимать одно из 2-х возможных решений. Существует множество задач, решаемых таким способом. Одна из них – выполнение операции, условно говоря, X с каждым элементом дерева. X рассматривается в качестве параметра обобщённой задачи посещения всех вершин либо задачи обхода дерева. Если рассмотреть такую задачу в качестве единого последовательного процесса, то можно сказать, что отдельные вершины посещаются в определённом порядке, то есть могу считаться линейно расположенными.

Целью данной курсовой работы является создание программного средства для сортировки и бинарного поиска в массиве. Для достижения поставленной цели необходимо выполнить следующие задачи:

* рассмотрение аналогов программных средств;
* изучение алгоритмов правильного распределения нагрузки;
* объектно-ориентированный анализ и проектирование системы;
* выявление требований к аппаратным и операционным ресурсам;
* определение необходимых инструментов для разработки.
* проектирование задачи;
* описание программного средства;
* написание программы;
* проведение испытаний и устранение выявленных ошибок.

При анализе электронных и литературных источников было выявлено два интернет-приложения, отвечающее требованиям к разрабатываемой программе: https://visualgo.net/en/bst и https://www.cs.usfca.edu/~galles/visualization/BST.html

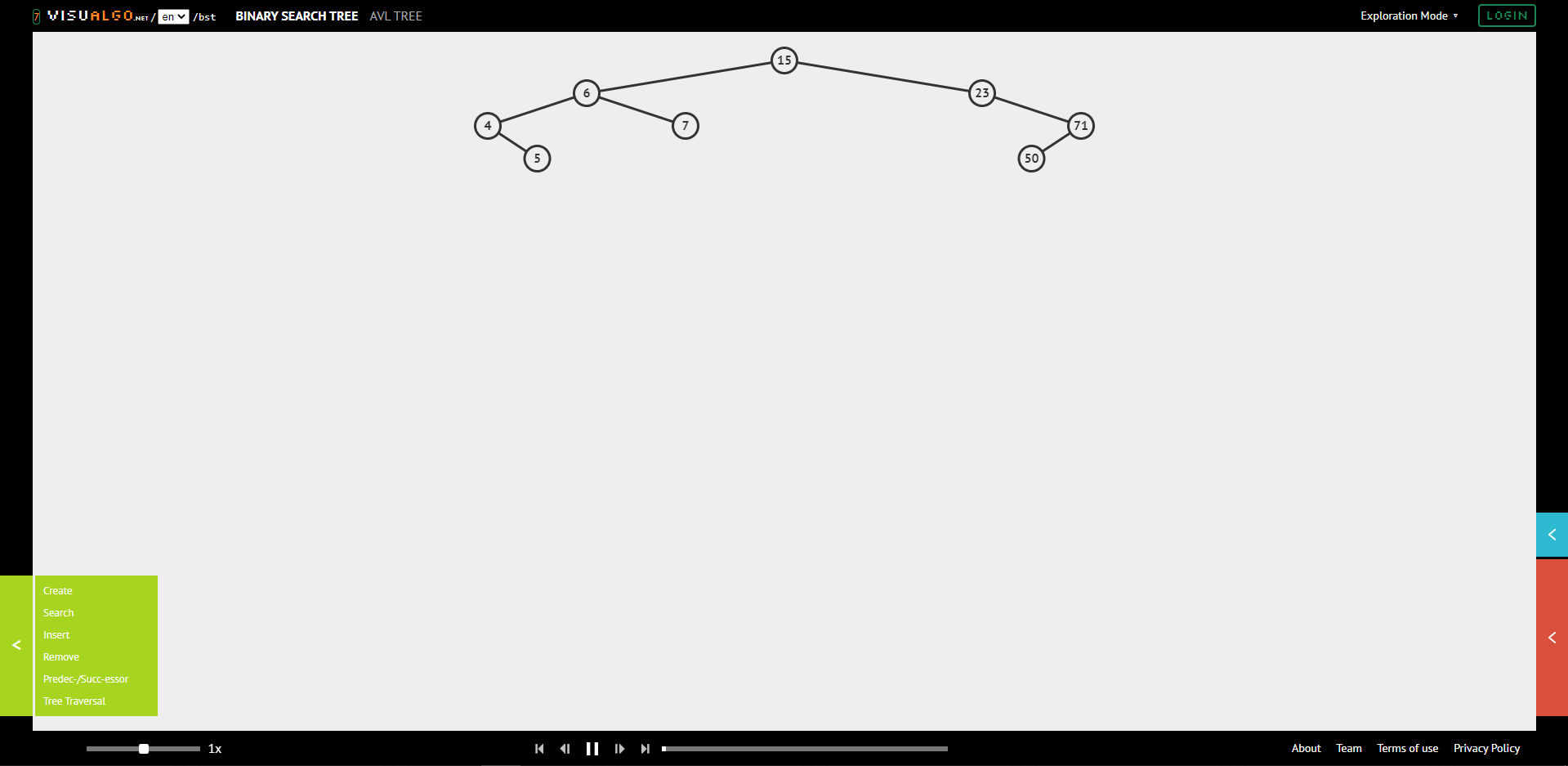


Рисунок 1 - Интернет-ресурс https://visualgo.net/en/bst

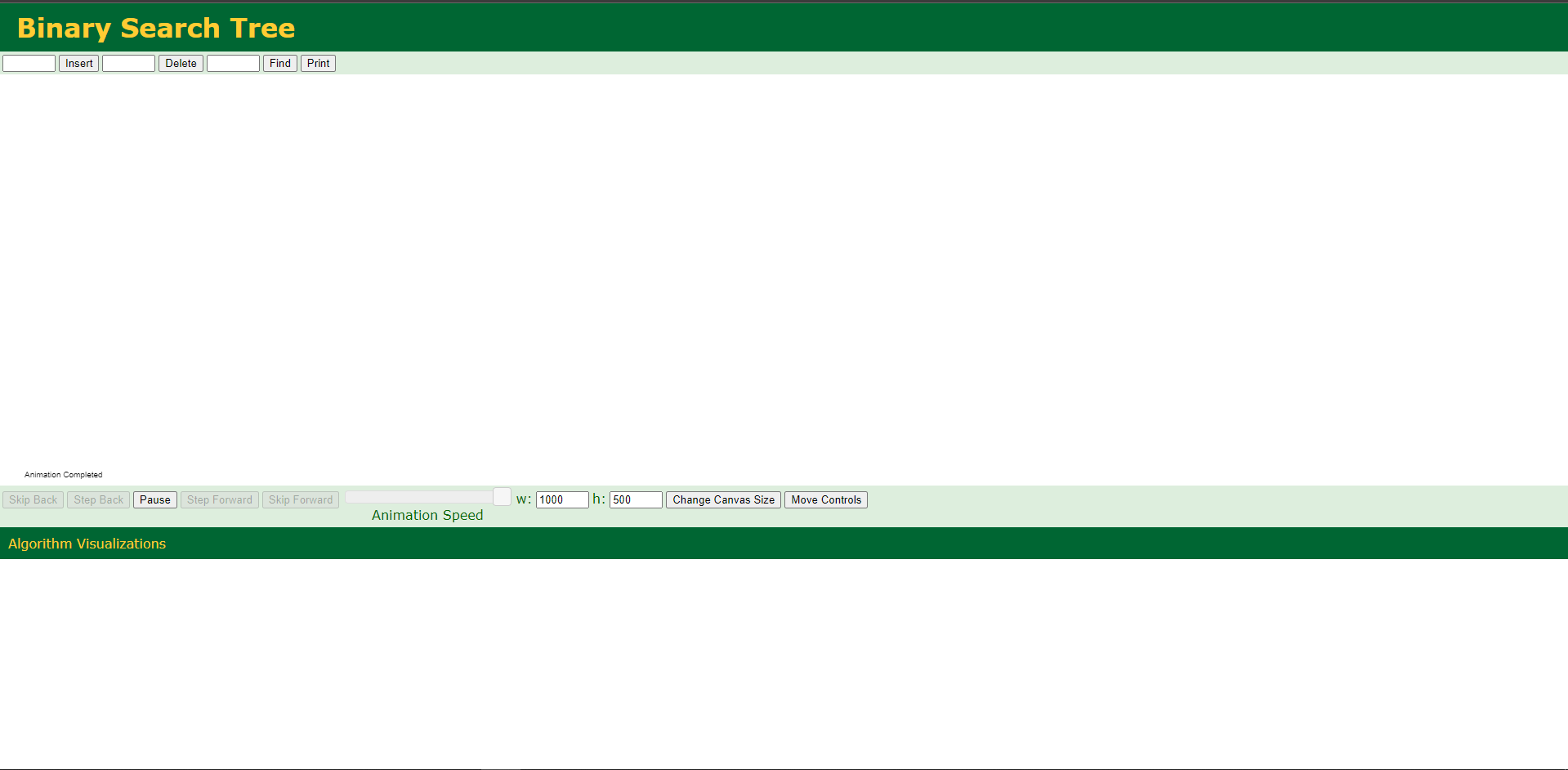


Рисунок 2 - Интернет-ресурс https://www.cs.usfca.edu/~galles/visualization/BST.html

# Объектно-ориентированный анализ и проектирование системы

## Сущность задачи

При любой сортировке происходит сравнение величин (полей сортировки). Массив чисел сортируется по принципу сравнения: если надо отсортировать по возрастанию (неубыванию), то сравниваем значения соседних элементов, и если левый элемент по хранимому в нем значению больше правого, то хранимые в них значения надо обменять местами; если массив надо отсортировать по убыванию (невозрастанию), то сравниваем значения соседних элементов, и если леворасположенный элемент хранит в себе значение меньшее, чем в праворасположенном элементе, то хранимые в них значения надо обменять местами. При более сложных алгоритмах сортировок сравниваться могут не соседние элементы массива, а имеющие между собой другие «промежуточные» элементы, но все равно эти удаленные друг от друга элементы сравниваются и учитывается, какой элемент расположен левее, а какой – правее. Возможно оперирование и сразу последовательностями элементов (группы расположенных рядом элементов, удовлетворяющих критерию сортировки, переставляются с другими группами элементов).

Бинарное дерево — это иерархическая структура данных, в которой каждый узел имеет значение (оно же является в данном случае и ключом) и ссылки на левого и правого потомка. Узел, находящийся на самом верхнем уровне (не являющийся чьим-либо потомком), называется корнем. Узлы, не имеющие потомков (оба потомка которых равны NULL) называются листьями.

Программное средство «Сортировки и бинарного поиска в массиве» позволит легче работать с массивом, без труда его сортировать, а также визуализировать его в виде бинарных деревьев и выполнять поиск.

Потенциальная аудитория – математики, студенты, программисты.

Функции необходимые для автоматизации:

* процесс ввода информации о массиве, необходимой для сортировки и работы бинарного дерева;
* экспорт массива в текстовый документ.

Исходя из всех вышеперечисленных факторов можно сделать вывод что написанное программное средство позволит начинающим программистам, математикам или же студентам показать и визуализировать работу с массивами и бинарными деревьями.

## Проектирование модели

Основное окно программы состоит из 3 вкладок: «Инициализация массива», «Сортировка Хоара» и «Бинарное дерево». В первой вкладке реализована возможность инициализации массива, используя кнопки «рандом размер» для рандомизации размера массива и «рандом массив» для его случайного заполнения и вывода в текстовое окно. Также массив можно создать индивидуально, для этого нужно указать размер массива по строкам и столбцам в текстовое окно, а после нажать кнопку «Создание массива» и начать его дальнейшее заполнение вручную через кнопку «Добавление числа в массив». Вторая вкладка отвечает за сортировку Хоара и его вывод в текстовое окно. При нажатии на кнопку «Сортировать» произойдёт сортировка массива и его вывод в текстовое окно. В третьей вкладке реализована возможность визуализации бинарного дерева при нажатии кнопки «Создать». Ниже находится окошко для бинарного поиска. При вводе ключа в текстовое окно, а после, нажатии кнопки «Поиск», будет реализован бинарный поиск. Кнопка «Очистить» очищает текстовые окна, уничтожает массивы, вызывает функцию очистки бинарного дерева и позволяет начать ввод заново.

Пользователю доступна возможность чтения и записи файлов посредством меню приложения. Также, в меню доступны функции смены темы оформления, локализации, вывода справки программы, а также выход из самой программы.

# Вычислительная система

## Требования к аппаратным операционным ресурсам

Довольно важную роль при разработке программных средств имеет компьютер, на котором ведется разработка. Написание программного средства «Сортировки и бинарного поиска в массиве» будет вестись на персональном компьютере, со следующими характеристиками:

* процессор Intel Core i7-4700HQ, 2013 года выпуска, техпроцесс 22 нанометра, частота 3.4 GHz, 4 ядра, 8 потоков;
* оперативная память 8ГБ, DDR3, 1600 МГц;
* видеокарта Nvidia GeForce GTX 850M 4ГБ;
* операционная система MS Windows 10 Pro, 64-разрядная.

## Инструменты разработки

Для быстрой и эффективной разработки, отладки и развертки приложения были использованы нижеследующие программные средства:

* операционная система MS Windows 10 Pro;
* среда программирования MS Visual Studio 2022 – использовалась для написания кода программы, построения форм приложения, отладки и развертки;
* язык программирования «C++» – был выбран из-за быстродействия, гибкости и

возможности работать с крупными массивами данных;

* paint – программа использовалась для создания иконки приложения в формате «png»;
* веб-приложение www.xiconeditor.com – использовалось для конвертации «png» файла иконки в формат «ico»;
* smart install maker 5.04 – использовалась для создания инсталлятора и деинсталлятора

для готовой программы.

# Проектирование задачи

## Требования к приложению

Основные функциональные требования к проекту

* реализовать авторизацию и аутентификацию пользователей, локализацию на трёх языках (английский, русский и белорусский), а также оформление (возможность выбора цвета фона окна приложения из 3-х вариантов);
* реализовать ввод пользователем данных о размере массивов, ввод их содержимого и обеспечить возможность рандомного заполнения массивов;
* реализовать сортировку Хоара и бинарный поиск;
* реализовать визуализацию работу алгоритма бинарного поиска в виде деревьев
* реализовать справочную систему программы.

Требования к интерфейсу максимально просты, интерфейс должен быть «дружелюбен» и понятен пользователю. Так же интерфейс должен обеспечивать организацию интуитивно-понятного пользования им.

Защиты информации от несанкционированного использования не предусмотрено ТЗКП. Средством защиты может выступать пользователь, который должен предусмотреть список людей, которые будут иметь доступ к данному программному средству.

## Концептуальный прототип

Разработанное программное средство представляет собой файл «.exe».

При двойном нажатии правой кнопкой мыши на файл, запускается окно авторизации и аутентификации пользователей состоящая из главного меню, текстовых окон для заполнения данных и соответствующими кнопками управления, изображённых на Рисунок 3.

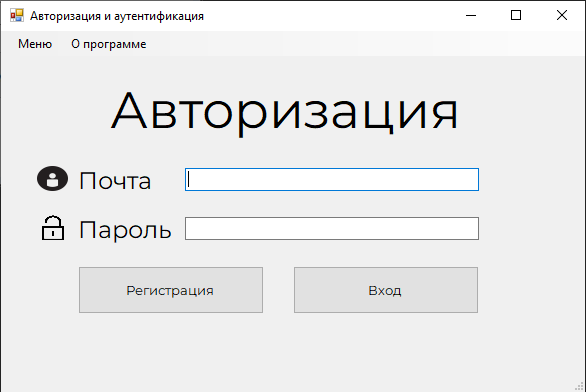


Рисунок 3 - Окно авторизации и аутентификации

В окне регистрации представлены регистрация пользователя, состоящая из главного меню, текстовых окон для заполнения данных и соответствующими кнопками управления, изображённых на Рисунок 4.

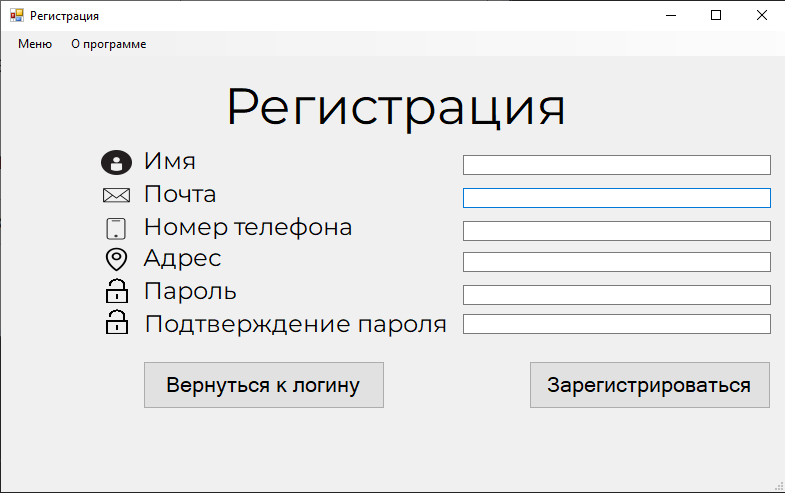


Рисунок 4 - Окно регистрации

Основное окно программы состоит из 3 вкладок: «Инициализация массива», «Сортировка Хоара» и «Бинарное дерево». В первой вкладке реализована возможность инициализации массива, используя кнопки «рандом размер» для рандомизации размера массива и «рандом массив» для его случайного заполнения и вывода в текстовое окно. Также массив можно создать индивидуально, для этого нужно указать размер массива по строкам и столбцам в текстовое окно, а после нажать кнопку «Создание массива» и начать его дальнейшее заполнение вручную через кнопку «Добавление числа в массив». Представлено на Рисунок 5.

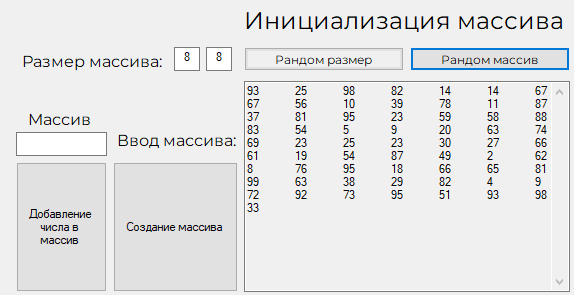


Рисунок 5 - Вкладка инициализации массива

Вторая вкладка отвечает за сортировку Хоара и его вывод в текстовое окно. При нажатии на кнопку «Сортировать» произойдёт сортировка массива и его вывод в текстовое окно. Представлено на Рисунок 6.

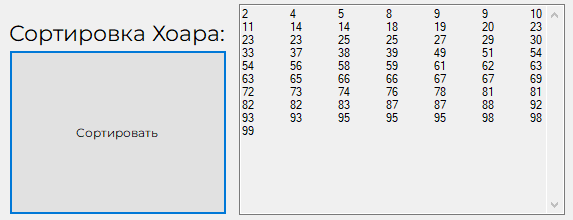


Рисунок 6 - Вкладка Сортировки Хоара

В третьей вкладке реализована возможность визуализации бинарного дерева при нажатии кнопки «Создать». Ниже находится окошко для бинарного поиска. При вводе ключа в текстовое окно, а после, нажатии кнопки «Поиск», будет реализован бинарный поиск. Кнопка «Очистить» очищает текстовые окна, уничтожает массивы, вызывает функцию очистки бинарного дерева и позволяет начать ввод заново. Представлено на Рисунок 7.



Рисунок 7 - Вкладка бинарного дерева и поиска

Пользователю также доступна возможность чтения и записи файлов посредством меню приложения. Представлено на Рисунок 8 и Рисунок 9.

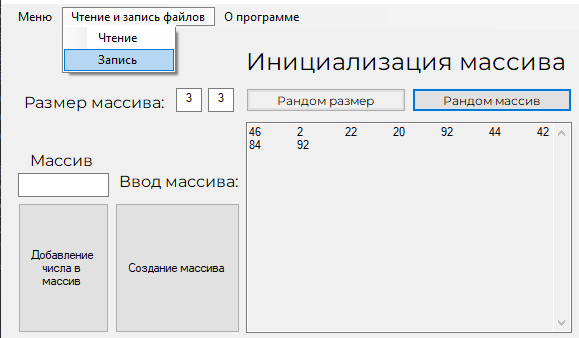


Рисунок 8 - Запись массива в файл

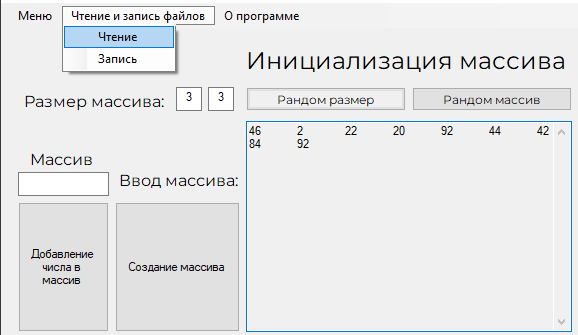


Рисунок 9 - Чтение массива из файла

Также, в меню доступны функции смены темы оформления, локализации, а также выход из самой программы. Представлено на Рисунок 10 и Рисунок 11.

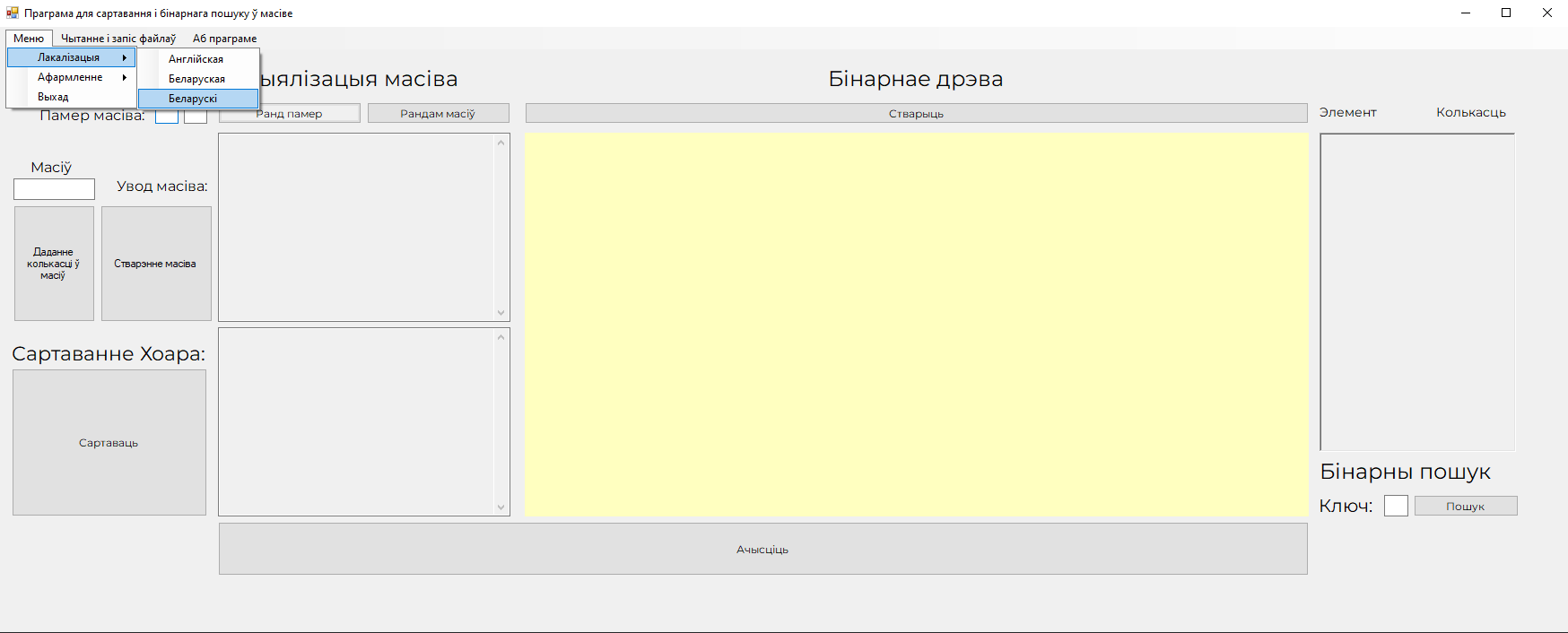


Рисунок 10 - Локализация

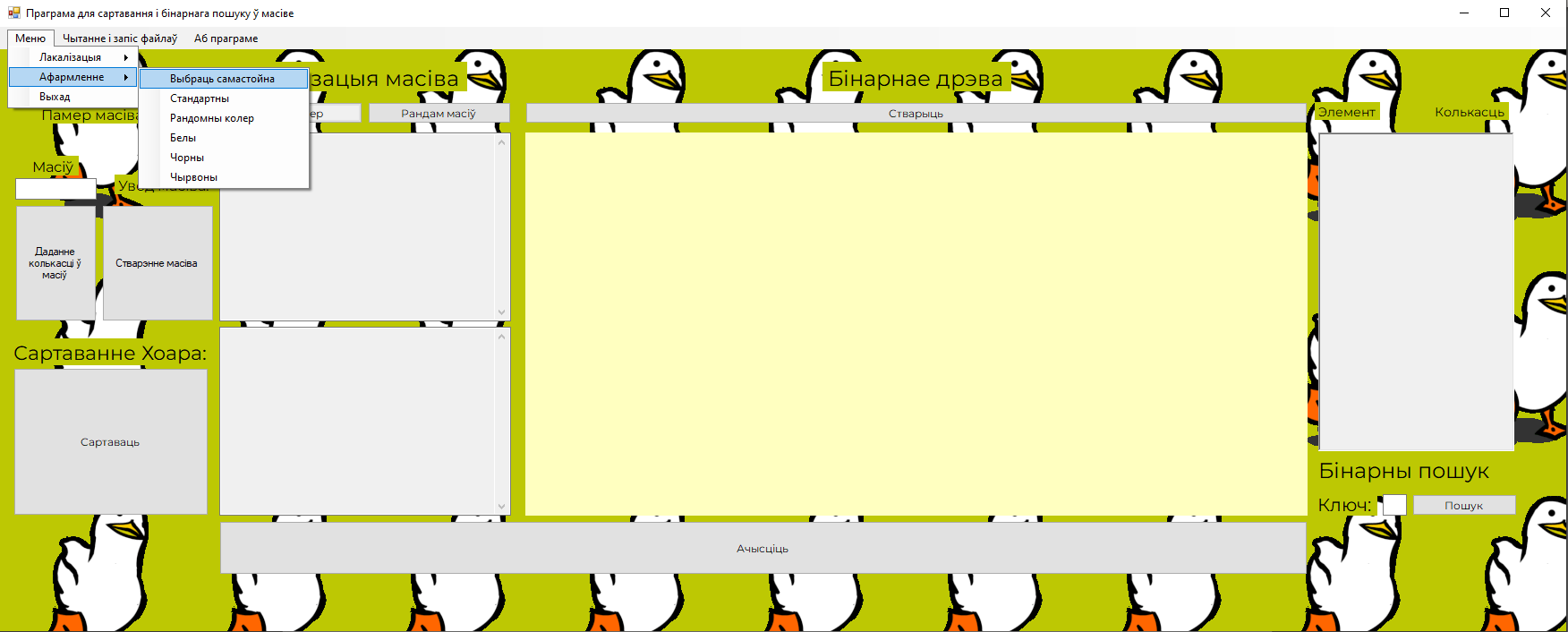


Рисунок 11 - Оформление

В последней кнопки меню реализован вывод краткой справки о программе. Представлено на Рисунок 12.

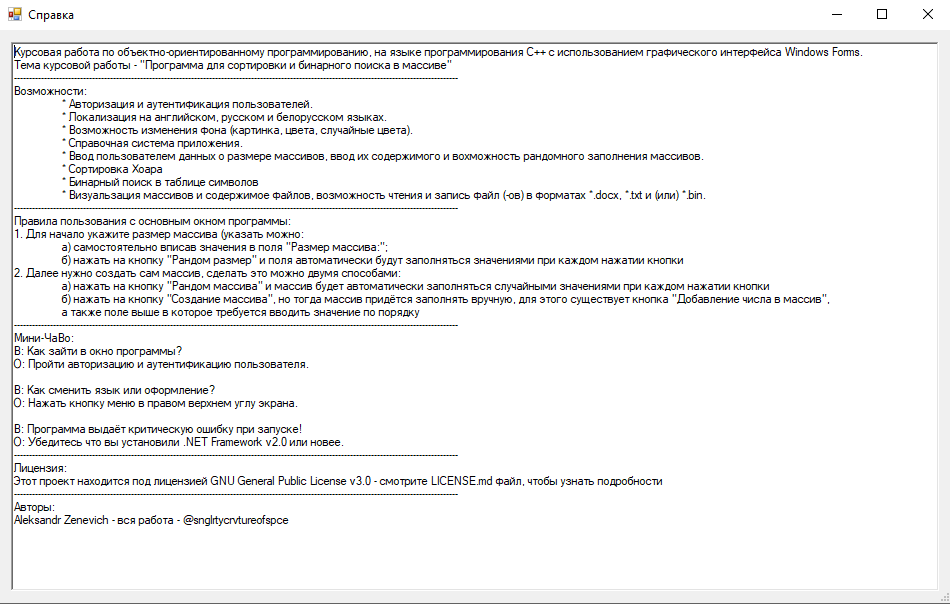


Рисунок 12 – Справка

## Организация данных

| Размещение | Элемент экрана | Класс | Действие |
| --- | --- | --- | --- |
| Форма программы | Рандом размер | Button | Рандомизация размера массива и его отображение в текстовые окна |
| Рандом массив | Button | Рандомизация элементов массива и его отображение в текстовое окно |
| Создание массива | Button | Создание пустого массива и его отображение в текстовое окно |
| Добавление числа в массив | Button | Добавление числа в массив начиная с нулевого элемента и его постоянное обновление в текстовом окне |
| Сортировать | Button | Реализует сортировку массива алгоритмом быстрой сортировки Хоара |
| Создать | Button | Визуализация бинарного дерева в панель |
| Очистить | Button | Очищает текстовые окна, уничтожает массивы, вызывает функцию очистки бинарного дерева и позволяет начать ввод заново. |
| Поиск | Button | Реализует бинарный поиск, ключ берётся с текстового окна. |
| Меню – Локализация | ToolStripMenuItem | Выбор языка приложения |
| Меню – Оформление | ToolStripMenuItem | Выбор темы приложения |
| Меню – Выход | ToolStripMenuItem | Закрытие приложения |
| Чтение | ToolStripMenuItem | Чтение данных из файла и вывод в текстовое окно |
| Запись | ToolStripMenuItem | Запись данных из текстового окна в файл |
| О программе | ToolStripMenuItem | Открытие краткой справки |

Таблица 1 - Элементы управления и визуализации

## Функции: логическая и физическая организация и элементы управления

| Функция | Компонент меню | Название элемента  интерфейса | Реализация |
| --- | --- | --- | --- |
| Смена языка | английскийToolStripMenuItem1\_Click | Щелчок мыши по кнопке «Меню – Локализация – Английский» | Средства С++ |
| Смена языка | русскийToolStripMenuItem2\_Click | Щелчок мыши по кнопке «Меню – Локализация – Русский» | Средства С++ |
| Смена языка | русскийToolStripMenuItem3\_Click | Щелчок мыши по кнопке «Меню – Локализация – Белорусский» | Средства С++ |
| Смена оформление заднего фона картинкой форматов (\*.BMP; \*.JPG; \*.GIF; \*.PNG) | выбратьСамостоятельноToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Оформление – Выбрать самостоятельно» | Средства С++ |
| Смена цвета заднего фона на стандартный | стандартныйToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Оформление – Стандартный» | Средства С++ |
| Рандомизация цвета заднего фона | рандомныйЦветToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Оформление – Рандомный цвет» | Средства С++ |
| Смена цвета заднего фона на белый | белыйToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Оформление – Белый» | Средства С++ |
| Смена цвета заднего фона на чёрный | чёрныйToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Оформление – Чёрный» | Средства С++ |
| Смена цвета заднего фона на красный | красныйToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Оформление – Красный» | Средства С++ |
| Закрыть программу | выходToolStripMenuItem\_Click | Щелчок мыши по кнопке «Меню – Выход» | Средства С++ |
| Вывод окна справки программы | оПрограммеToolStripMenuItem\_Click | Щелчок мыши по кнопке «О программе» | Средства С++ |
| Рандомизация элементов массива и его отображение в текстовое окно | button1\_Click | Кнопка «Рандом массив» | Средства С++ |
| Рандомизация размера массива и его отображение в текстовые окна | button2\_Click | Кнопка «Рандом размер» | Средства С++ |
| Очищает текстовые окна, уничтожает массивы, вызывает функцию очистки бинарного дерева и позволяет начать ввод заново. | button3\_Click | Кнопка «Очистить» | Средства С++ |
| Реализует сортировку массива алгоритмом быстрой сортировки Хоара | button4\_Click | Кнопка «Сортировать» | Средства С++ |
| Создание пустого массива и его отображение в текстовое окно | button5\_Click | Кнопка «Создание массива» | Средства С++ |
| Добавление числа в массив начиная с нулевого элемента и его постоянное обновление в текстовом окне | button6\_Click | Кнопка «Добавление числа в массив» | Средства С++ |
| Реализует бинарный поиск, ключ берётся с текстового окна. | button7\_Click | Кнопка «Поиск» | Средства С++ |
| Визуализация бинарного дерева в панель | button8\_Click | Кнопка «Создать» | Средства С++ |
| Чтение данных из файла и вывод в текстовое окно | чтениеToolStripMenuItem\_Click | Щелчок мыши по кнопке «Чтение и запись файлов – Чтение» | Средства С++ |
| Запись данных из текстового окна в файл | записьToolStripMenuItem\_Click | Щелчок мыши по кнопке «Чтение и запись файлов – Запись» | Средства С++ |

# Описание программного средства

## Общие сведения

Детальные сведения о программе и проблеме, которую она решает, приводятся в подразделе 1.1.

Также, в ходе выполнения курсового проекта были задействованы следующие подключаемые модули приложения:

1. «vector»;
2. «string».

«Windows Forms» – интерфейс программирования приложений, отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Имеется возможность разработки на C#, C++, VB.Net, J# и др.

«vector» – это структура данных, которая уже является моделью динамического массива, стандартный шаблон обобщённого программирования и реализующий его.

«string» – класс с методами и переменными для организации работы со строками в языке программирования С++. Он включён в стандартную библиотеку С++. Название образовано от имени строчного типа данных (англ. string; с англ. - «строка»).

## Функциональное назначение

Программа разработана для работы с массивом, без труда его сортировать, а также визуализировать его в виде бинарных деревьев и выполнять поиск. Все ошибки, которые могут возникнуть во время работы программы обработаны с использованием блоков «try-catch».

## Входные и выходные данные

В качестве входных данных используются конвертируемые целочисленные значения из текстовых окон для размера массива, а также если пользователь выберет инициализацию массива самостоятельно. Для бинарного дерева в качестве входных данных используются структуры «Node» класса «BinaryTree».

В качестве выходных данных используются текстовые окна, весь вывод массива, а после и отсортированный также в нём. Также, в свою очередь выходными данными являются данные, записанные в текстовый файл.

# Методика испытаний

## Технические требования

Для полноценного функционирования разработанного программного приложения, качественного выполнения всех поставленных задач, требуется ПК следующей конфигурации:

* процессор с частотой не менее 1 ГГц;
* ОЗУ от 1 ГБ (для 32-разрядной ОС) или 2 ГБ (для 64-разрядной ОС);
* от 16 (для 32-разрядной ОС) или от 20 ГБ (для 64-разрядной ОС) свободного места на жестком диске;
* наличие DirectX 9 и более поздняя версия с драйвером WDDM 1.0.
* .NET Framework 4.0 или новее.
* операционная система: MS Windows 7 или новее;

Для проведения испытаний функций сохранения и загрузки использовались файлы с заранее сохраненными в них данными. Тестирование проводилось на аппаратных базах ноутбуков Asus N750JK и HP Pavilion 15, а также на стационарных ПК с различными техническими конфигурациями. Во время тестирования программы проблем, связанных с аппаратной совместимостью, выявлено не было.

## Функциональное тестирование

В этом подразделе приведены примеры реакций программы на ввод некорректных данных. При обнаружении ошибки отображается модальное окно с информацией о ней, а ввод некорректных данных отменяется. Это предохраняет пользователя от записи этих данных в программу и позволяет продолжить ее выполнение без прерываний и критических ошибок.

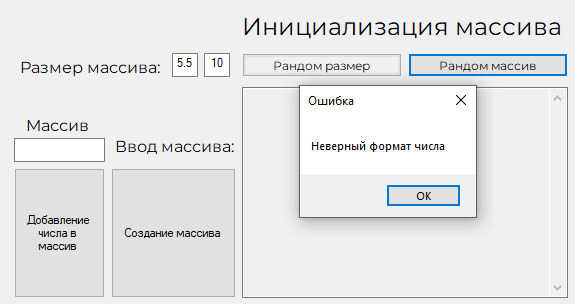


Рисунок 13 - Ввод данных неправильного формата в текстовое окно

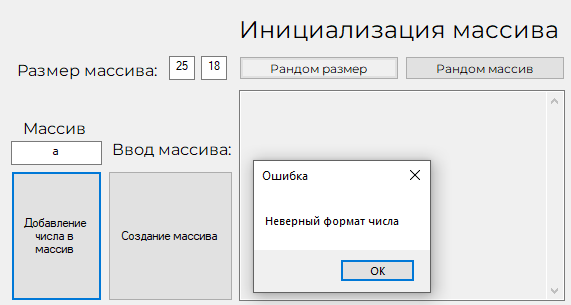


Рисунок 14 - Ввод буквы в окно целочисленного формата

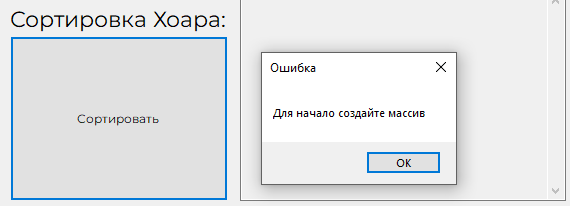


Рисунок 15 - Попытка сортировать массив не имея самого массива

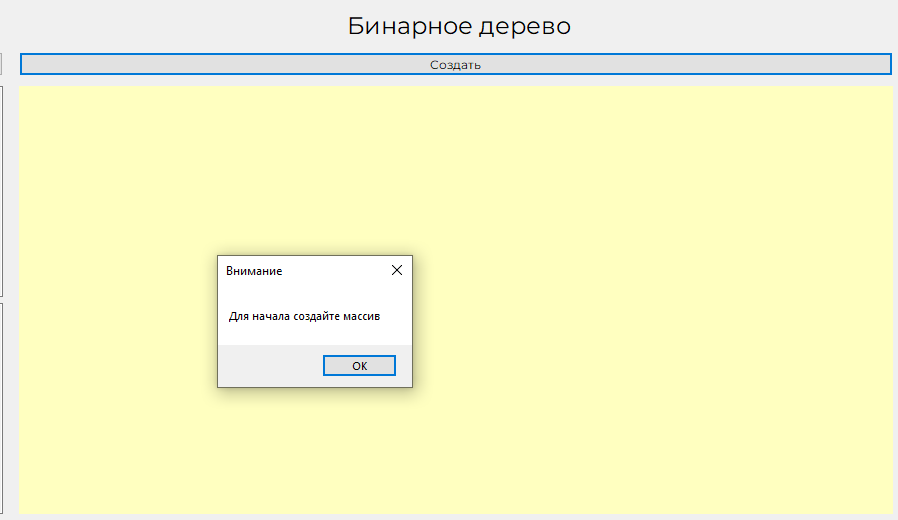


Рисунок 16 - Попытка создания бинарного дерева без инициализированного массива

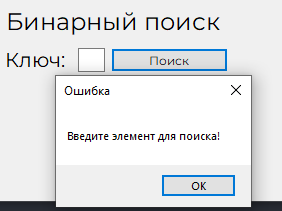


Рисунок 17 - Попытка бинарного поиска без ключа

# Применение

## Назначение программы

Назначение программы приведено в подразделе 4.2

## Условия применения

Данное программное приложение не обладает средствами, ограничивающими доступ к нему, а также средствами, запрещающих распространять программу. Для работы с программным средством пользователю нужна операционная система MS Windows.

Для программы установлены следующие ограничения:

* размер массива должен быть не больше 30 элементов;

Требования к техническим и программным ресурсам программы приведены в подразделе 2.1.

# Заключение

Во время написания курсовой работы была разработана, отлажена и развернута программа для сортировки и бинарного поиска в массиве, реализован современный и интуитивно понятный интерфейс, реализована авторизация и аутентификация пользователей, локализация и простое оформление, работа с файлами, окнами и прочими элементами Windows Forms. Созданный класс бинарного дерева можно легко адаптировать для интеграции в другие проекты. При разработке был получен ценный опыт разработки оконных приложений, работы в Visual C++ и фреймворком .NET. Итоговый продукт соответствует современным требованиям к прикладным приложениям, готов к использованию в сферах образования и программирования, допускает возможность дальнейшей доработки и изменения. Ниже приведены некоторые ошибки и аспекты программы, которые могут быть изменены и доработаны:

* если размер массива большой, то отрисовка бинарного дерева выходит за рамки панели, и частичка не видна;
* можно сделать куда более приятное изменение цвета в оформлении, при выборе цвета лейблы искажаются, и при определённом цвете их можно не заметить;
* наличие более профессиональной справочной системы могло бы помочь пользователю разобраться в программе.

Разработанная программа может стать основой для полноценного инструмента работы с бинарными деревьями и массивами, который может включать себя такие возможности как:

* визуализация бинарного дерева;
* бинарный поиск;
* инициализация и вывод в текстовое окно массива;
* индивидуальное создание массива и его заполнение;
* сортировка массива методом Хоара.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2017. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
3. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб. : БХВ-Петербург, 2013. – 608 с.
4. Эрнст Нойферт. Строительное проектирование. 1936г – 176 с.
5. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. – 816 с
6. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. – 720 с.
7. Тепляков, С. Паттерны проектирования на платформе .NET / С.Тепляков. – СПб. : Питер, 2015. – 320 с.
8. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с
9. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
10. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
11. Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : http://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/. – Дата доступа : 25.04.2020.
12. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://www.microsoft.com/ru-ru/windows/features. – Дата доступа : 22.05.2020.
13. Visual Studio 2019 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://visualstudio.microsoft.com/ru/vs. – Дата доступа : 22.04.2020.

Приложение А

(обязательное)

Код программы

//Код файла main.cpp

#include <Windows.h>

#include "LoginForm.h"

#include "MainForm.h"

#include "RegisterForm.h"

/\*

main.cpp содержит обработчики событий основного окна

программы и другие вспомогательные функции

\*/

using namespace coursework; // использование пространства имён coursework

[STAThreadAttribute]

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

try

{

Application::EnableVisualStyles(); // визуальные стили для приложения

Application::SetCompatibleTextRenderingDefault(false); // обеспечения визуальной совместимости между элементами управления Windows Forms

User^ user = nullptr; // зануляем указатель

while (true) // пока истина

{

coursework::LoginForm loginForm; // создаём переменную для обращения к форме

loginForm.ShowDialog(); // показать форму логина

if (loginForm.switchToRegister)

{

coursework::RegisterForm registerForm; // создаём переменную для обращения к форме

registerForm.ShowDialog(); // показать форму регистрации

if (registerForm.switchToLogin)

{

continue; // продолжаем

}

else

{

user = registerForm.user;

break; // прерываем

}

}

else

{

user = loginForm.user;

break; // прерываем

}

}

if (user != nullptr) // пока не равен нулевому указателю

{

coursework::MainForm mainForm(user); // создаём переменную с параметром формы для дальнейшего обращения к ней

Application::Run(% mainForm); // запускаем форму

}

else

{

Application::Exit(); // закрытие окна и выброс приложения из памяти

}

return 0; // возвращаем

}

catch (System::Exception^ exception)

{

MessageBox::Show("exception->Message"); // вывод сообщения ошибки на экран

}

}

//Код файла LoginForm.h

#include "User.h"

#include "HELP.h"

/\*

LoginForm.h содержит панель авторизации пользователей

\*/

namespace coursework {

using namespace System::Data::SqlClient; // пространство имён для подключения SQL базы

using namespace System::Globalization; // использования пространства имён для локализации

using namespace System::Threading; // использования пространства имён для локализации

}

// Локализация

private: System::Void английскийToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // английский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("en-US");

CultureInfo::CurrentCulture = gcnew CultureInfo("en-US");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void русскийToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e) { // русский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("");

CultureInfo::CurrentCulture = gcnew CultureInfo("");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void русскийToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // белорусский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("be");

CultureInfo::CurrentCulture = gcnew CultureInfo("be");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

// Оформление

private: System::Void выбратьСамостоятельноToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // выбрать картинку

try

{

OpenFileDialog^ openDlg = gcnew OpenFileDialog(); // открыть окно диалога выбора файлов

openDlg->Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*"; // фильтр формата выбираемых картинок

if (System::Windows::Forms::DialogResult::OK == openDlg->ShowDialog())

{

this->BackgroundImage = Image::FromFile(openDlg->FileName); // задний фон = файл (картинка) выбранная пользователем

}

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void стандартныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // стандартный цвет

this->BackColor = SystemColors::ButtonFace; // стандартный цвет заднего цвета фона

this->BackgroundImage = nullptr; // инициализируем нулевой указатель на картинку заднего фона

}

private: System::Void рандомныйЦветToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // рандомный цвет

try

{

Random rn; // объявляем переменную рандом

// рандомим константы для цвета RGB

const int r = rn.Next(0, 255); // (минимальное значение, максимальное значение)

const int g = rn.Next(0, 255); // (минимальное значение, максимальное значение)

const int b = rn.Next(0, 255); // (минимальное значение, максимальное значение)

this->BackColor = Color::FromArgb(r, g, b); // задний цвет фона = рандомный цвет из переменных r, g, b

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void белыйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // белый цвет

this->BackColor = Color::White; // белый цвет заднего цвета фона

}

private: System::Void чёрныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // чёрный цвет

this->BackColor = Color::Black; // чёрный цвет заднего цвета фона

}

private: System::Void красныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // красный цвет

this->BackColor = Color::Red; // красный цвет заднего цвета фона

}

// Описание кнопки выхода и справки программы

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // выход из программы

this->Close(); // закрытие формы

}

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

coursework::HELP helpform; // создаём переменную для работы с формой

helpform.ShowDialog(); // обращаемся к форме и открываем её

}

// Описание алгоритма кнопки "Регистрация"

public: bool switchToRegister = false; // глобальная переменная для переключения к регистрации, изначально инициализируем как ложь (0)

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

this->switchToRegister = true; // инициализируем переменную истиной (1), алгоритм активизируется

this->Close(); // закрытие формы

}

// Описание алгоритма логина

public: User^ user = nullptr; // глобальная переменная класса пользователь ("User.h"), изначально инициализируем как нулевой указатель

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

String^ email = this->tbEmail->Text; // инициализируем переменную строкового класса данными из текстового поля

String^ password = this->tbPassword->Text; // инициализируем переменную строкового класса данными из текстового поля

if (email->Length == 0 || password->Length == 0) // проверка на заполнение всех полей

{

MessageBox::Show("Please enter email and password", "Email or password empty", MessageBoxButtons::OK);

return;

}

try

{

String^ connString = "Data Source=localhost\\sqlexpress;Initial Catalog=courseWORK;Integrated Security=True"; // инициализируем переменную строкового класса адресом подключения к базе данных

SqlConnection sqlConn(connString); // задаём адрес для подключения к базе данных

sqlConn.Open(); // открываем соединение с базой данных

String^ sqlQuery = "SELECT \* FROM users WHERE email=@email AND password=@password;";

SqlCommand command(sqlQuery, % sqlConn);

command.Parameters->AddWithValue("@email", email);

command.Parameters->AddWithValue("@password", password);

SqlDataReader^ reader = command.ExecuteReader();

if (reader->Read())

{

user = gcnew User;

user->id = reader->GetInt32(0);

user->name = reader->GetString(1);

user->email = reader->GetString(2);

user->phone = reader->GetString(3);

user->address = reader->GetString(4);

user->password = reader->GetString(5);

this->Close(); // закрытие формы

}

else

{

MessageBox::Show("Email or Password is incorrect", "Email or Password Error", MessageBoxButtons::OK); // вывод сообщения на экран

}

}

catch (Exception^ exception)

{

MessageBox::Show("Failed to connect to database", "Database Connection Error", MessageBoxButtons::OK); // вывод сообщения на экран

}

}

};

}

//Код файла RegisterForm.h

#include "User.h"

#include "HELP.h"

/\*

RegisterForm.h содержит панель регистрации пользователей

\*/

namespace coursework {

using namespace System::Data::SqlClient; // пространство имён для подключения SQL базы

using namespace System::Globalization; // использования пространства имён для локализации

using namespace System::Threading; // использования пространства имён для локализации

}

// Локализация

private: System::Void английскийToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e) { // английский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("en-US");

CultureInfo::CurrentCulture = gcnew CultureInfo("en-US");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message);

}

}

private: System::Void русскийToolStripMenuItem2\_Click(System::Object^ sender, System::EventArgs^ e) { // русский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("");

CultureInfo::CurrentCulture = gcnew CultureInfo("");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message);

}

}

private: System::Void русскийToolStripMenuItem3\_Click(System::Object^ sender, System::EventArgs^ e) {// белорусский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("be");

CultureInfo::CurrentCulture = gcnew CultureInfo("be");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message);

}

}

// Оформление

private: System::Void выбратьСамостоятельноToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // выбрать картинку

try

{

OpenFileDialog^ openDlg = gcnew OpenFileDialog(); // открыть окно диалога выбора файлов

openDlg->Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*"; // фильтр формата выбираемых картинок

if (System::Windows::Forms::DialogResult::OK == openDlg->ShowDialog())

{

this->BackgroundImage = Image::FromFile(openDlg->FileName); // задний фон = файл (картинка) выбранная пользователем

}

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void стандартныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // стандартный цвет

this->BackColor = SystemColors::ButtonFace; // стандартный цвет заднего цвета фона

this->BackgroundImage = nullptr; // инициализируем нулевой указатель на картинку заднего фона

}

private: System::Void рандомныйЦветToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // рандомный цвет

try

{

Random rn; // объявляем переменную рандом

// рандомим константы для цвета RGB

const int r = rn.Next(0, 255); // (минимальное значение, максимальное значение)

const int g = rn.Next(0, 255); // (минимальное значение, максимальное значение)

const int b = rn.Next(0, 255); // (минимальное значение, максимальное значение)

this->BackColor = Color::FromArgb(r, g, b); // задний цвет фона = рандомный цвет из переменных r, g, b

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void белыйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // белый цвет

this->BackColor = Color::White; // белый цвет заднего цвета фона

}

private: System::Void чёрныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // чёрный цвет

this->BackColor = Color::Black; // чёрный цвет заднего цвета фона

}

private: System::Void красныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // красный цвет

this->BackColor = Color::Red; // красный цвет заднего цвета фона

}

// Описание кнопки выхода и справки программы

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // выход из программы

this->Close(); // закрытие формы

}

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

coursework::HELP helpform; // создаём переменную для работы с формой

helpform.ShowDialog(); // обращаемся к форме и открываем её

}

// Описание алгоритма кнопки "Вернуться к логину"

public: bool switchToLogin = false; // глобальная переменная для переключения к логину, изначально инициализируем как ложь (0)

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

this->switchToLogin = true; // инициализируем переменную истиной (1), алгоритм активизируется

this->Close(); // закрытие формы

}

// Описание алгоритма регистрации

public: User^ user = nullptr; // глобальная переменная класса пользователь ("User.h"), изначально инициализируем как нулевой указатель

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

String^ name = tbName->Text; // инициализируем переменную строкового класса данными из текстового поля

String^ email = tbEmail->Text; // инициализируем переменную строкового класса данными из текстового поля

String^ phone = tbPhone->Text; // инициализируем переменную строкового класса данными из текстового поля

String^ address = tbAddress->Text; // инициализируем переменную строкового класса данными из текстового поля

String^ password = tbPassword->Text; // инициализируем переменную строкового класса данными из текстового поля

String^ confirmPassword = tbConfirmPassword->Text; // инициализируем переменную строкового класса данными из текстового поля

if (name->Length == 0 || email->Length == 0 || phone->Length == 0 || address->Length == 0 || password->Length == 0) // проверка на заполнение всех полей

{

MessageBox::Show("Please enter all the fields", "On or more empty fields", MessageBoxButtons::OK); // вывод сообщения на экран

return; // возвращаем

}

if (String::Compare(password, confirmPassword) != 0) // сравниваем пароль и подтверждение пароля

{

MessageBox::Show("Password and Confirm Password do not match", "Password Error", MessageBoxButtons::OK); // вывод сообщения на экран

return; // возвращаем

}

try

{

String^ connString = "Data Source=localhost\\sqlexpress;Initial Catalog=courseWORK;Integrated Security=True"; // инициализируем переменную строкового класса адресом подключения к базе данных

SqlConnection sqlConn(connString); // задаём адрес для подключения к базе данных

sqlConn.Open(); // открываем соединение с базой данных

String^ sqlQuery = "INSERT INTO users " + "(name, email, phone, address, password) VALUES " + "(@name, @email, @phone, @address, @password);";

SqlCommand command(sqlQuery, % sqlConn);

command.Parameters->AddWithValue("@name", name);

command.Parameters->AddWithValue("@email", email);

command.Parameters->AddWithValue("@phone", phone);

command.Parameters->AddWithValue("@address", address);

command.Parameters->AddWithValue("@password", password);

command.ExecuteNonQuery();

user = gcnew User;

user->name = name;

user->email = email;

user->phone = phone;

user->address = address;

user->password = password;

this->Close(); // закрытие формы

}

catch (Exception^ exception)

{

MessageBox::Show("Failed to register new user", "Register Failure", MessageBoxButtons::OK); // вывод сообщения на экран

}

}

};

}

//Код файла HELP.h

/\*

HELP.h содержит панель справки программы

\*/

namespace coursework {

using namespace System::IO; // Содержит типы, позволяющие читать и записывать файлы и потоки данных,

// а также типы, обеспечивающие базовую поддержку файлов и каталогов.

}

private: System::Void HELP\_Load(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text = File::ReadAllText("HelpRU.txt"); // чтение справки из файла

}

};

}

//Код файла User.h

/\*

User.h - имеет класс пользователя с переменными всех данных, нужен для реализации SQL базы

\*/

using namespace System;

public ref class User

{

public:

int id;

String^ name;

String^ email;

String^ phone;

String^ address;

String^ password;

};

//Код файла User.h

#include "User.h"

#include "HELP.h"

#include "BinaryTree.h"

#include "functions.h"

/\*

MainForm.h - основное окно программы

\*/

namespace coursework { // использование пространства имён coursework

using namespace System::Globalization; // использования пространства имён для локализации

using namespace System::Threading; // использования пространства имён для локализации

MainForm(User^ user) // конструктор с параметром, принимает тип User

{

InitializeComponent();

bTree = new BinaryTree<int>();

}

BinaryTree<int>\* bTree;

// Локализация

private: System::Void английскийToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e) { // английский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("en-US");

CultureInfo::CurrentCulture = gcnew CultureInfo("en-US");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void русскийToolStripMenuItem2\_Click(System::Object^ sender, System::EventArgs^ e) { // русский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("");

CultureInfo::CurrentCulture = gcnew CultureInfo("");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void русскийToolStripMenuItem3\_Click(System::Object^ sender, System::EventArgs^ e) { // белорусский

try

{

this->Controls->Clear();

CultureInfo::CurrentUICulture = gcnew CultureInfo("be");

CultureInfo::CurrentCulture = gcnew CultureInfo("be");

InitializeComponent(); // инициализирует объекты пользовательского интерфейса

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

// Оформление

private: System::Void выбратьСамостоятельноToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // выбрать картинку

try

{

OpenFileDialog^ openDlg = gcnew OpenFileDialog(); // открыть окно диалога выбора файлов

openDlg->Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*"; // фильтр формата выбираемых картинок

if (System::Windows::Forms::DialogResult::OK == openDlg->ShowDialog())

{

this->BackgroundImage = Image::FromFile(openDlg->FileName); // задний фон = файл (картинка) выбранная пользователем

}

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void стандартныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // стандартный цвет

this->BackColor = SystemColors::ButtonFace; // стандартный цвет заднего цвета фона

this->BackgroundImage = nullptr; // инициализируем нулевой указатель на картинку заднего фона

}

private: System::Void рандомныйЦветToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // рандомный цвет

try

{

Random rn; // объявляем переменную рандом

// рандомим константы для цвета RGB

const int r = rn.Next(0, 255); // (минимальное значение, максимальное значение)

const int g = rn.Next(0, 255); // (минимальное значение, максимальное значение)

const int b = rn.Next(0, 255); // (минимальное значение, максимальное значение)

this->BackColor = Color::FromArgb(r, g, b); // задний цвет фона = рандомный цвет из переменных r, g, b

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void белыйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // белый цвет

this->BackColor = Color::White; // белый цвет заднего цвета фона

}

private: System::Void чёрныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // чёрный цвет

this->BackColor = Color::Black; // чёрный цвет заднего цвета фона

}

private: System::Void красныйToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // красный цвет

this->BackColor = Color::Red; // красный цвет заднего цвета фона

}

// Описание кнопки выхода и справки программы

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // выход из программы

this->Close(); // закрытие формы

}

private: System::Void оПрограммеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // вывод окна справки программы

coursework::HELP helpform; // создаём переменную для работы с формой

helpform.ShowDialog(); // обращаемся к форме и открываем её

}

// Описание работы алгоритма программы для сортировки и бинарного поиска в массиве

private: int rows = 0, cols = 0, r = 0; // размер динамически неуправляемого Сборщиком мусора массива rows (строки), cols (столбцы)

private: int\*\* arr = nullptr; // нулевой указатель на целочисленный неуправляемый двумерный массив

private: int\* arr2 = nullptr; // нулевой указатель на целочисленный неуправляемый одномерный массив

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) { // рандом массива и его вывод

try

{

this->textBox2->Clear(); // очистка текстового окна

this->textBox3->Clear(); // очистка текстового окна

this->rows = Convert::ToInt32(this->textBox1->Text); // конвертация введённым пользователем размера строки в целочисисленное значение

this->cols = Convert::ToInt32(this->textBox4->Text); // конвертация введённым пользователем размера столбца в целочисисленное значение

if (rows >= 31 || cols >= 31) // ограничение на размер строк и столбцов

{

MessageBox::Show("Размер массива должен быть не больше 30 элементов!"); // вывод сообщения на экран

return; // возвращаем

}

int\*\* arr = new int\* [cols]; // создаём указатель второго уровня на двумерный динамический массив, а его элементы - указатели на строки

if (arr == nullptr) // если выделить участок в ДООП под массив не удалось

// ДООП (Динамической Области Оперативной Памяти)

{

MessageBox::Show("Не удаётся создать массив на " + "[" + rows.ToString() + "]" + "[" + cols.ToString() + "]" + " элементов."); // вывод сообщения на экран

return; // возвращаем

}

for (int i = 0; i < cols; i++) // выделяем память под столбцы массива

{

arr[i] = new int[rows]; // выделяем память под стобцы массива, которые состоят из "обычных" элементов типа int

if (arr[i] == nullptr) // если под очередной столбец не удалось выделить место в ДООП

{

MessageBox::Show("Не удаётся создать массив для строки №" + i); // вывод сообщения на экран

for (int j = 0; j < i; j++) // освободим участки ДООП, которые успели ранее выделить под предыдущие столбцы массива

{

delete[] arr[j]; // удаляем одномерный массив-столбец

}

delete[] arr; // удаляем указатель и двумерный массив

arr = nullptr; // зануляем указатель, что бы он наверняка не содержал адрес области ДООП

return; // возвращаем

}

}

Random rn; // объявляем переменную функцией рандом

for (int i = 0; i < cols; i++) // циклом проходимся по столбцам массива

{

for (int j = 0; j < rows; j++) // вложенным циклом проходимся по строкам массива

{

arr[i][j] = rn.Next(0, 101); // инициализируем рандомным значением в промежутке от 0 до 100 включительно

}

}

for (int i = 0; i < cols; i++) // циклом проходимся по столбцам массива

{

for (int j = 0; j < rows; j++) // вложенным циклом проходимся по строкам массива

{

this->textBox2->Text += arr[i][j].ToString() + "\t"; // конвертация элементов массива в строку и перенос в текстовое поле с добавлением управляющего символа табуляции

}

this->textBox2->Text += "\n"; // добавление управляющего символа переноса на новую строку в текстовое окно

}

arr2 = new int[rows \* cols]; // создание одномерного динамический массив

for (int i = 0; i < cols; i++) // циклом проходимся по столбцам массива

{

for (int j = 0; j < rows; j++) // вложенным циклом проходимся по строкам массива

{

arr2[rows \* i + j] = arr[i][j];

}

}

}

catch (System::FormatException^ exception)

{

MessageBox::Show("Неверный формат числа", "Ошибка", MessageBoxButtons::OK); // вывод сообщения ошибки формата данных на экран

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) { // рандом размер массива

try

{

Random^ rn = gcnew Random(); // объявляем переменную функцией рандом

int a = (rn->Next(1, 31)); // инициализируем переменную рандомным значением в промежутке от 1 до 30 включительно

int b = (rn->Next(1, 31)); // инициализируем переменную рандомным значением в промежутке от 1 до 30 включительно

textBox1->Text = a.ToString(); // конвертация целочисленной переменной в строку и перенос в текстовое окно

textBox4->Text = b.ToString(); // конвертация целочисленной переменной в строку и перенос в текстовое окно

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) { // очистка текстовых окон и массивов

try

{

this->textBox1->Clear(); // очистка текстового окна

this->textBox2->Clear(); // очистка текстового окна

this->textBox3->Clear(); // очистка текстового окна

this->textBox4->Clear(); // очистка текстового окна

this->textBox5->Clear(); // очистка текстового окна

ClearAll(); // вызываем функцию очистки бинарного дерева

delete[] arr; // уничтожение массива

arr = nullptr; // объявление массива нулевым указателем

delete[] arr2; // уничтожение массива

arr2 = nullptr; // объявление массива нулевым указателем

r = 0; // зануляем переменную

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

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

try

{

this->textBox3->Clear(); // очистка текстового окна

functions first; // инициализируем переменную структурным типом из функций ("functions.h")

first.quickSort(this->arr2, 0, (rows \* cols) - 1); // обращаемся к переменной и вызываем нужную функцию, т.е. функцию сортировки Хоара

for (int i = 0; i < (rows \* cols); i++) // циклом проходимся по всему массиву

{

this->textBox3->Text += arr2[i].ToString() + "\t"; // конвертация элементов массива в строку и перенос в текстовое поле с добавлением управляющего символа табуляции

}

}

catch (System::FormatException^ exception)

{

MessageBox::Show("Неверный формат числа", "Ошибка", MessageBoxButtons::OK); // вывод сообщения ошибки формата данных на экран

}

catch (System::NullReferenceException^ exception)

{

MessageBox::Show("Для начало создайте массив", "Ошибка", MessageBoxButtons::OK);

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) { // создание массива

try

{

this->textBox2->Clear(); // очистка текстового окна

this->textBox3->Clear(); // очистка текстового окна

this->textBox5->Clear(); // очистка текстового окна

if (arr != nullptr) // если участок в ДООП уже выделен

{

delete[] arr; // уничтожение массива

arr = nullptr; // объявление массива нулевым указателем

rows = 0; // зануляем размер строки

cols = 0; // зануляем размер столбца

}

r = 0;

this->rows = Convert::ToInt32(this->textBox1->Text); // конвертация введённым пользователем размера строки в целочисисленное значение

this->cols = Convert::ToInt32(this->textBox4->Text); // конвертация введённым пользователем размера столбца в целочисисленное значение

if (rows >= 31 || cols >= 31) // ограничение на размер строк и столбцов

{

MessageBox::Show("Размер массива должен быть не больше 30 элементов!"); // вывод сообщения на экран

return; // возвращаем

}

int\*\* arr = new int\* [cols]; // создаём указатель второго уровня на двумерный динамический массив, а его элементы - указатели на строки

if (arr == nullptr) // если выделить участок в ДООП под массив не удалось

// ДООП (Динамической Области Оперативной Памяти)

{

MessageBox::Show("Не удаётся создать массив на " + "[" + rows.ToString() + "]" + "[" + cols.ToString() + "]" + " элементов."); // вывод сообщения на экран

return; // возвращаем

}

for (int i = 0; i < cols; i++) // выделяем память под столбцы массива

{

arr[i] = new int[rows]; // выделяем память под стобцы массива, которые состоят из "обычных" элементов типа int

if (arr[i] == nullptr) // если под очередной столбец не удалось выделить место в ДООП

{

MessageBox::Show("Не удаётся создать массив для строки №" + i); // вывод сообщения на экран

for (int j = 0; j < i; j++) // освободим участки ДООП, которые успели ранее выделить под предыдущие столбцы массива

{

delete[] arr[j]; // удаляем одномерный массив-столбец

}

delete[] arr; // удаляем указатель и двумерный массив

arr = nullptr; // зануляем указатель, что бы он наверняка не содержал адрес области ДООП

return; // возвращаем

}

}

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

{

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

{

arr[i][j] = NULL;

}

}

arr2 = new int[rows \* cols]; // создание одномерного динамический массив

for (int i = 0; i < cols; i++) // циклом проходимся по столбцам массива

{

for (int j = 0; j < rows; j++) // вложенным циклом проходимся по строкам массива

{

arr2[rows \* i + j] = arr[i][j];

}

}

for (int i = 0; i < (rows \* cols); i++) // цикл будет выполняться до тех пор, пока он не пройдётся по всему массиву

{

this->textBox2->Text += arr2[i].ToString() + "\t"; // конвертация элементов массива в строку и перенос в текстовое поле с добавлением управляющего символа табуляции

}

}

catch (System::FormatException^ exception)

{

MessageBox::Show("Неверный формат числа", "Ошибка", MessageBoxButtons::OK); // вывод сообщения ошибки формата данных на экран

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) { // кнопка "Добавления элемента в массив"

try

{

this->textBox2->Clear(); // очистка текстового окна

this->textBox3->Clear(); // очистка текстового окна

if ((r < rows \* cols) && (rows \* cols > 0)) // проверка что размер массива больше нуля

{

arr2[r] = Convert::ToInt32(this->textBox5->Text); // конвертация введённым пользоваталем числа в целочисленное значение

r++; // r += 1

}

else

{

MessageBox::Show("Массив размерности [" + rows.ToString() + "]" + "[" + cols.ToString() + "] заполнен или не создан."); // вывод сообщения на экран

}

for (int i = 0; i < (rows \* cols); i++) // цикл будет выполняться до тех пор, пока он не пройдётся по всему массиву

{

this->textBox2->Text += arr2[i].ToString() + "\t"; // конвертация элементов массива в строку и перенос в текстовое поле с добавлением управляющего символа табуляции

}

}

catch (System::FormatException^ exception)

{

MessageBox::Show("Неверный формат числа", "Ошибка", MessageBoxButtons::OK); // вывод сообщения ошибки формата данных на экран

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

// Алгоритм чтения и записи файлов

private: System::Void чтениеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // чтение файла

textBox2->Clear(); // очистка текстового окна

textBox3->Clear(); // очистка текстового окна

delete[] arr; // уничтожение массива

arr = nullptr; // объявление массива нулевым указателем

delete[] arr2; // уничтожение массива

arr2 = nullptr; // объявление массива нулевым указателем

r = 0; // зануляем переменную

String^ FileName = ""; // переменная строкового класса для хранения пути к файлу

if (openFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK) // открываем диалоговое окно если пользователь нажал "ОК"

{

FileName = openFileDialog1->FileName; // записываем путь к выбранному файлу

}

try

{

StreamReader^ file = File::OpenText(FileName); // открываем файл

textBox2->Text = file->ReadToEnd(); // записываем содержимое файла в текстовое поле

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

private: System::Void записьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) { // запись файла

StreamWriter^ f = gcnew StreamWriter("test.txt", true); // создаём переменную для работы с файлами и сразу же создаём файл для работы

f->Write(textBox2->Text); // записываем в файл текст из текстового поля

f->Close(); // закрываем файл

MessageBox::Show(this, "Файл успешно сохранён", "Сообщение", MessageBoxButtons::OK, MessageBoxIcon::Information); // вывод сообщения на экран

textBox2->Clear(); // очистка текстового окна

}

// Алгоритм бинарного дерева и поиска

void ClearAll() // функция очистки бинарного дерева

{

panel1->Controls->Clear(); // очистка панели

panel1->Refresh(); // обновление панели

richTextBox1->Clear(); // очистка текстового окна

bTree->Clear(); // очистка бинарного дерева

}

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) { // бинарный поиск

//try

//{

// functions second; // объявление экземпляра структуры

// int key = 0; // целочисленный ключ ячейки, т.е. искомая переменная

// int index = 0; // целочисленный индекс ячейки с искомым значением

// key = Convert::ToInt32(this->textBox6->Text); // конвертация введённым пользоваталем ключа ячейки в целочисленное значение

// index = second.Search\_Binary(arr2, 0, rows \* cols, key); // функция бинарного поиска

// if (index >= 0) // если индекс больше или равен нулю

// {

// MessageBox::Show("Индекс элемента " + key + " в массиве равен: " + index); // вывод сообщения на экран

// }

// else // иначе

// {

// MessageBox::Show("Извините, но такого элемента в массиве нет"); // вывод сообщения на экран

// }

//}

//catch (System::FormatException^ exception)

//{

// MessageBox::Show(exception->Message); // вывод сообщения ошибки формата данных на экран

//}

//catch (System::Exception^ exception)

//{

// MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

//}

if (textBox6->Text->Length != 0) // проверка на то что длина текстового окна не равна нулю

{

int number = 0; // инициализируем переменную целочисленного типа

if (!Int32::TryParse(textBox6->Text, number)) // парсим

{

MessageBox::Show(L"Осторожно! Ошибка преобразования"); // вывод сообщения на экран

return; // возвращаем

}

bTree->Search(number) ? MessageBox::Show("Элемент найден") : MessageBox::Show("Элемент не найден"); // тернарный оператор, если искомый элемент найден - выводим успех, если не найден - выводим неудачу

}

else

{

MessageBox::Show("Введите элемент для поиска!", "Ошибка", MessageBoxButtons::OK); // вывод сообщения на экран

}

}

private: System::Void button8\_Click(System::Object^ sender, System::EventArgs^ e) { // визуализация бинарного дерева

try

{

Graphics^ panelGraphics = panel1->CreateGraphics();

int startX = 0, startY = 0;

int curX = 0, curY = 0;

if (arr2 != nullptr)

{

for (int i = 0; i < rows \* cols; i++)

{

bTree->AddItem(arr2[i]);

}

}

else

{

MessageBox::Show("Для начала создайте массив", "Внимание");

}

for (auto& item : bTree->PrintTree())

{

curX = item->theX;

curY = item->theY;

Label^ label = gcnew Label();

label->Font = gcnew System::Drawing::Font("verdana", 12);

label->Width = 40;

label->BackColor = Color::YellowGreen;

if (curX > panel1->Width - label->Width)

curX = panel1->Width - label->Width - 5;

if (curY < 0)

curY = 5;

label->Location = Point(curX, curY);

label->Text = Convert::ToString((int)item->Data);

if (startX != startY != 0)

{

panelGraphics->DrawLine(gcnew Pen(Color::Brown, 4), Point(startX, startY), Point(curX, curY));

}

panelGraphics->FillRectangle(Brushes::Black, curX - 2, curY - 2, 40, label->Height);

panel1->Controls->Add(label);

if (item->testRoot)

{

startX = curX;

startY = curY;

}

richTextBox1->AppendText(String::Format(" {0} {1}\r\n", item->Data, item->Count));

}

}

catch (System::Exception^ exception)

{

MessageBox::Show(exception->Message); // вывод сообщения ошибки на экран

}

}

};

}

//Код файла BinaryTree.h

#include <vector>

#include <string>

/\*

BinaryTree.h - представляет класс бинарного дерева, включая его методы визуализации

\*/

template <class T>

class BinaryTree

{

private:

struct Node

{

T Data;

int Count;

int theX, theY;

Node\* Left, \* Right;

bool testRoot;

Node(T \_Data) : Data(\_Data), testRoot(false)

{

Count = theX = theY = 0;

Left = Right = NULL;

}

};

Node\* Root;

void Insert(T \_Data, Node\*& paramnode, int \_theX, int \_theY)

{

if (paramnode == NULL)

{

paramnode = new Node(\_Data);

paramnode->testRoot = true;

paramnode->Count = 1;

paramnode->theX = \_theX;

paramnode->theY = \_theY;

}

else

{

if (\_Data == paramnode->Data)

paramnode->Count++;

else if (\_Data < paramnode->Data)

{

paramnode->testRoot = false;

Insert(\_Data, paramnode->Left, paramnode->theX - 70, paramnode->theY - 30);

}

else if (\_Data > paramnode->Data)

{

paramnode->testRoot = false;

Insert(\_Data, paramnode->Right, paramnode->theX + 70, paramnode->theY - 30);

}

}

}

std::vector<Node\*> PrintTree(Node\* paramnode)

{

std::vector<Node\*> myvector, othervector;

if (paramnode != NULL)

{

myvector.push\_back(paramnode);

othervector = PrintTree(paramnode->Left);

myvector.insert(myvector.begin(), othervector.begin(), othervector.end());

othervector = PrintTree(paramnode->Right);

myvector.insert(myvector.begin(), othervector.begin(), othervector.end());

}

return myvector;

}

Node\* Search(T \_Data, Node\* paramNode)

{

if (paramNode != NULL)

{

if (\_Data == paramNode->Data)

return paramNode;

if (\_Data < paramNode->Data)

return Search(\_Data, paramNode->Left);

else

return Search(\_Data, paramNode->Right);

}

else return NULL;

}

public:

BinaryTree()

:Root(NULL)

{

}

~BinaryTree()

{

DestroyTree(Root);

}

void Clear()

{

DestroyTree(Root);

}

void AddItem(T value)

{

int startX = 250;

int startY = 250;

Insert(value, Root, startX, startY);

}

std::vector<Node\*> PrintTree()

{

return PrintTree(Root);

}

bool Search(T \_Data)

{

return Search(\_Data, Root);

}

void DestroyTree(Node\* paramnode)

{

if (paramnode != NULL)

{

DestroyTree(paramnode->Left);

DestroyTree(paramnode->Right);

delete paramnode;

paramnode = NULL;

}

Root = NULL;

}

};

//Код файла functions.h

/\*

RegisterForm.h содержит панель регистрации пользователей

\*/

public struct functions

{

void swap(int& first, int& second) // функция обмена значениями между двумя элементами.

// Чтобы обменять значения у ОРИГИНАЛОВ значений в main'е,

// эта функция принимает ссылки на элементы, то есть константные

// указатели на адреса элементов

{

int t = first;

first = second;

second = t;

}

void quickSort(int\* m, int left, int right) // функция быстрой сортировки (рекурсивной или сортировки Хоара)

{

int leftArrow = left, rightArrow = right, pivot = m[(left + right) / 2];

do

{

while (m[rightArrow] > pivot)

{

rightArrow--;

}

while (m[leftArrow] < pivot)

{

leftArrow++;

}

if (leftArrow <= rightArrow)

{

swap(m[leftArrow], m[rightArrow]);

leftArrow++;

rightArrow--;

}

} while (rightArrow >= leftArrow);

if (left < rightArrow)

{

quickSort(m, left, rightArrow);

}

if (leftArrow < right)

{

quickSort(m, leftArrow, right);

}

}

};

**Удостоверяющий лист**

электронного документа – курсовой проект

Тема КП \_Программа для сортировки и бинарного поиска в массиве\_\_\_\_\_\_\_\_\_

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

Обозначение КП  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ КП Т.0910006.401\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработчик \_Зеневич А.О. \_\_  Руководитель \_\_Шаляпин Ю.В.\_\_

(Ф.И.О.)   (Ф.И.О.)

Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав ЭД | Разработчик | Руководитель |
| Пояснительная записка (на  бумажном носителе формата  А4), файл ПЗ\_Зеневич\_Т091.docx |  |  |
| Папка с проектом «cousework» |  |  |
| ГЧ  №1, файлы СортировкиХоара.png |  |  |
| ГЧ  №2, файл БинарныйПоиск.png |  |  |
| Программная документация:  файлы MainForm.h, MainForm.cpp, LoginForm.h, LoginForm.cpp, RegisterForm.h, RegisterForm.cpp, HELP.h, HELP.cpp, BinaryTree.h, User.h, functions.h |  |  |
| Установочный пакет программного средства «setup.exe» |  |  |
| Тип носителя: оптический диск |  |  |

**Этикетка**

**Курсовой проект**

Тема \_\_\_ Программа для сортировки и бинарного поиска в массиве\_\_\_\_\_\_\_

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

КП Т.0910006.401

Разработан \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утвержден\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработчик: *\_\_\_\_\_* Зеневич А.О.\_\_\_

Руководитель: \_\_\_Шаляпин Ю.В.\_\_\_\_

Технические средства

Программные средства: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Состав документа:**

Пояснительная записка – файл ПЗ.docx

Программные документы – файлы MainForm.h, MainForm.cpp, LoginForm.h,

LoginForm.cpp, RegisterForm.h, RegisterForm.cpp, HELP.h, HELP.cpp, BinaryTree.h,

User.h, functions.h

Графическая часть – файлы СортировкиХоара.png, БинарныйПоиск.png