МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Крымский федеральный университет имени В.И. Вернадского»

Таврический колледж

(структурное подразделение)

**ОТЧЕТ ПО ПРАКТИКЕ**

**Учебная практика по профессиональному модулю**

**ПМ.03 Участие в интеграции программных модулей**

Специальность **09.02.03 Программирование в компьютерных системах**

Обучающийся 4 курса группы **4ПКС18**

форма обучения⠀⠀⠀⠀очная⠀⠀⠀⠀

(очная, заочная)

⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀ ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀  
(фамилия, имя, отчество)

Место практики

Таврический колледж (структурное подразделение) ФГАОУ «КФУ им. В.И. Вернадского»

(наименование организации)

Срок практики с **16 марта 2023 г. по 22 марта 2023 г.**

Руководитель практики

от колледжа

преподаватель\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Руденко А.В. /

должность подпись (Ф.И.О.)

Зам директора

по учебно-производственной

практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Малюга Г.Г. /

подпись (Ф.И.О.)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(отлично, хорошо, удовлетворительно)

МП

г. Симферополь, 2023 г.

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ 3

ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5

1.1 Разработка технического задания 5

1.2 Разработка спецификаций 6

1.3 Разработка диаграмм 7

ГЛАВА 2. МОДУЛИ ПРОГРАММЫ 12

2.1 Модуль интерфейса 12

2.2 Модуль CoctailSort 13

2.3 Модуль QuickSort 14

2.4 Модуль SwapSort 17

2.5 Модуль InsertionSort 19

ЗАКЛЮЧЕНИЕ 23

СПИСОК ЛИТЕРАТУРЫ 25

ПРИЛОЖЕНИЕ А 26

**ВВЕДЕНИЕ**

Я, Сизой Максим Андреевич, проходил учебную практику на базе: Таврического колледжа (структурное подразделение) ФГАОУ «КФУ им. В.И. Вернадского».

Дата начала практики: 16 марта 2023 г.

Дата окончания практики: 22 марта 2023 г.

Дата сдачи отчёта по практике: 22 марта 2023 г.

Цель практики:

Формирование и развитие общих и профессиональных компетенций по модулю ПМ.03 Участие в интеграции программных модулей.

Задачи учебной практики:

Закрепление навыков разработки программного обеспечения;

Использование методов для получения кода с заданной функциональностью и степенью качества;

Разработка документации на программный продукт;

Знание моделей процесса разработки программного обеспечения, основных принципов процесса разработки программного обеспечения;

Знание основных подходов к интегрированию программных модулей, основных методов и средств эффективной разработки ПО;

Задание для выполнения:

1. Необходимо разработать программный комплекс по демонстрации работы алгоритмов сортировки массивов данных (реализовать не менее 4 алгоритмов сортировки, которые выбрать самостоятельно):
   1. Разработать техническое задание на программный продукт.
   2. Разработать спецификацию на программный продукт.
   3. Разработать функциональную диаграмму программного продукта, диаграмму потоков данных программных модулей продукта.
   4. Разработать функциональную схему программного продукта, составить блок-схемы программных модулей программного продукта.
   5. Разработать коды программных модулей программного продукта.
   6. Разработать пользовательский интерфейса программного продукта в визуальной среде.
   7. Выполнить интеграцию программных модулей в программный продукт.
   8. Разработать процедуру тестирования программного продукта. Выполнить тестирование программного продукта. Результат тестирования оформить протоколом тестирования.
   9. Разработать справочную систему программного продукта.
   10. Разработать руководства оператора (пользователя).
2. Создать аккаунт в GitHub. Создать папку проекта. В папку загрузить разработанный программный комплекс, всю разработанную документацию к проекту (п.п.1.1 – 1.10).
3. Составить отчет о выполнении.

**ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

1.1 Разработка технического задания

Наименование программы: "Сизой Максим 4ПКС18".

Программа предназначена для сортировки массивов различных размеров 4-мя разными методами сортировки.

Программа предоставляет свой пользовательский интерфейс

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.   
На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

Этапы разработки:  
1. Разработка программы

2. Разработка программной документации.

3. Испытание программы.

На стадии внедрения должен быть выполнен этап разработки подготовка и передача программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:   
1. постановка задачи;   
2. определение и уточнение требований к техническим средствам;   
3. определение требований к программе;  
4. определение стадий, этапов и сроков разработки программы и документации на неё;   
5. согласование и утверждение технического задания.   
На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.   
На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями к составу документации.   
На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:   
1. разработка, согласование и утверждение и методики испытаний;   
2. проведение приемо-сдаточных испытаний;   
3. корректировка программы и программной документации по результатам испытаний.   
На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

1.2 Разработка спецификаций

Программа “Сизой Максим 4ПКС18” реализована на языке программирования C# Windows Form, предназначена для выполнения задач, связанных с сортировкой различного размера массивов разными методами, среди них: метод пузырьком, метод выборки, метод перестановки, и метод вставки, в программе представлены следующие модули: Модуль интерфейса, модуль QuickSort, модуль SwapSort, модуль InsertionSort, модуль CoctailSort.

Предназначена для людей, у которых стоит потребность в сортировке массивов различных размеров.

1.3 Разработка диаграмм

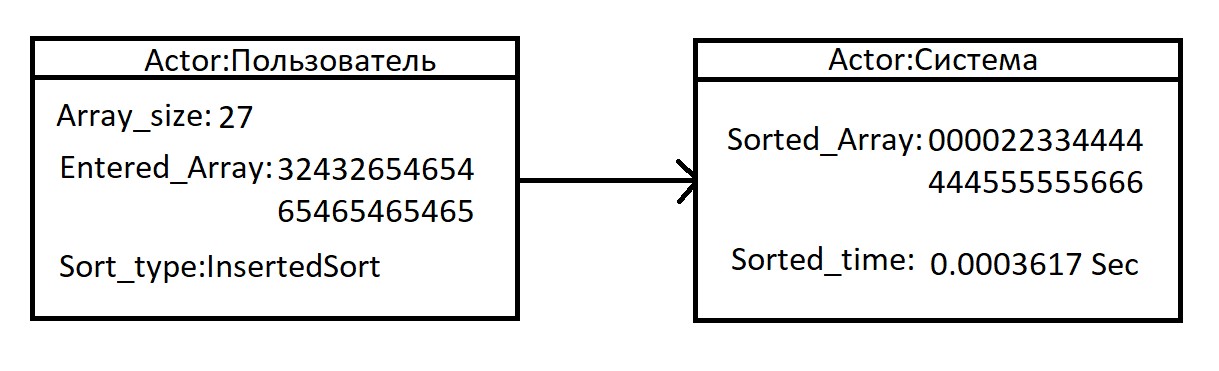


Рисунок 1 – Первая UML диаграмма

На данном рисунке представлена первая UML диаграмма с выполнением алгоритма сортировки массива методом вставки с размерностью массива 27.

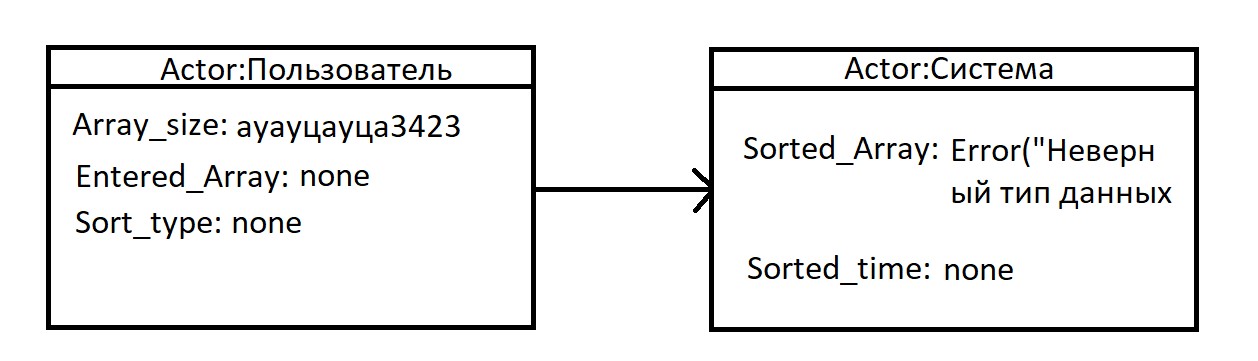


Рисунок 2 – Вторая UML диаграмма

а данном рисунке представлена вторая UML диаграмма с выполнением алгоритма с заведомо некорректными данными.

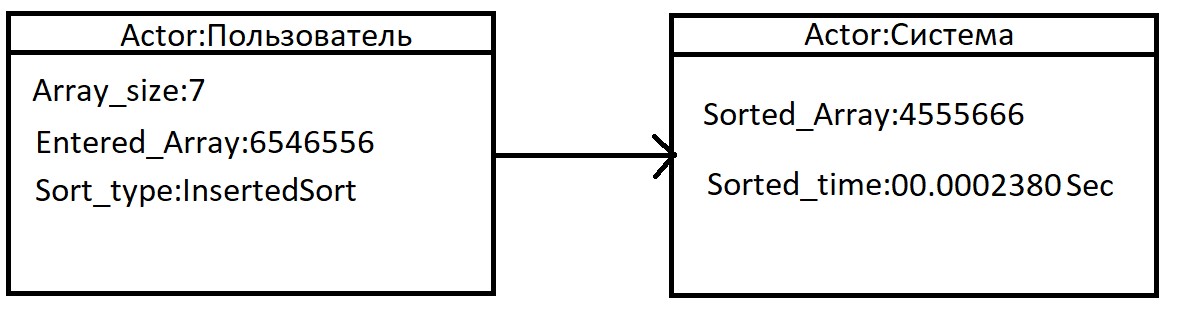


Рисунок 3 – Третья UML диаграмма

на данном рисунке представлена третья UML диаграмма с выполнением алгоритма сортировки массива методом вставки с размерностью массива 7.

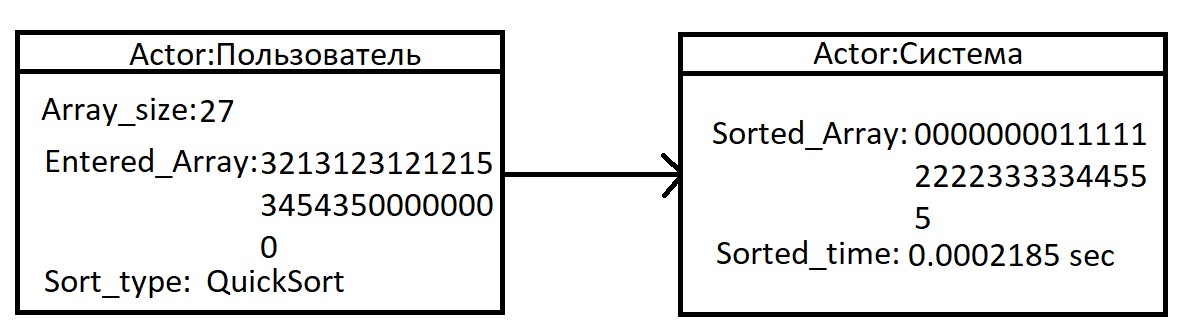


Рисунок 4 – Четвёртая UML диаграмма

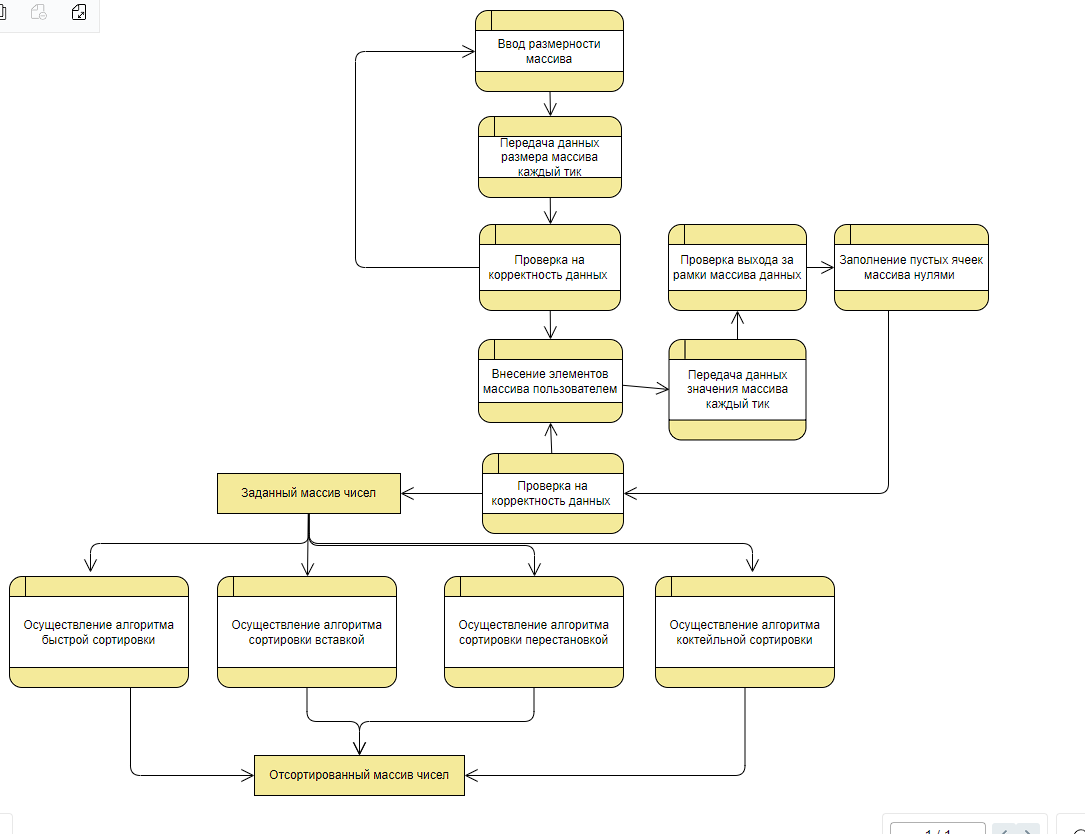
на данном рисунке представлена четвёртая UML диаграмма с выполнением алгоритма сортировки массива методом быстрой сортировки с размерностью массива 27.

Рисунок 5 – Диаграмма потока данных

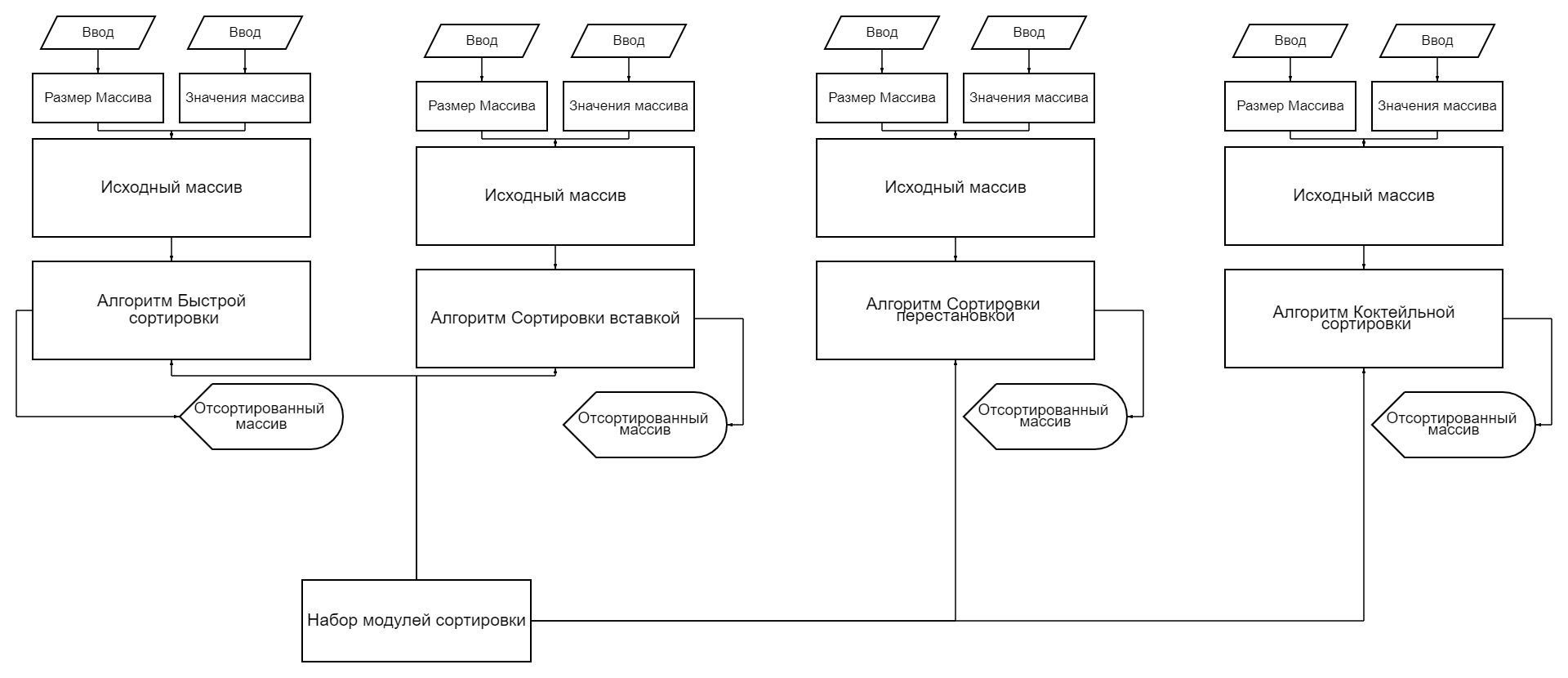
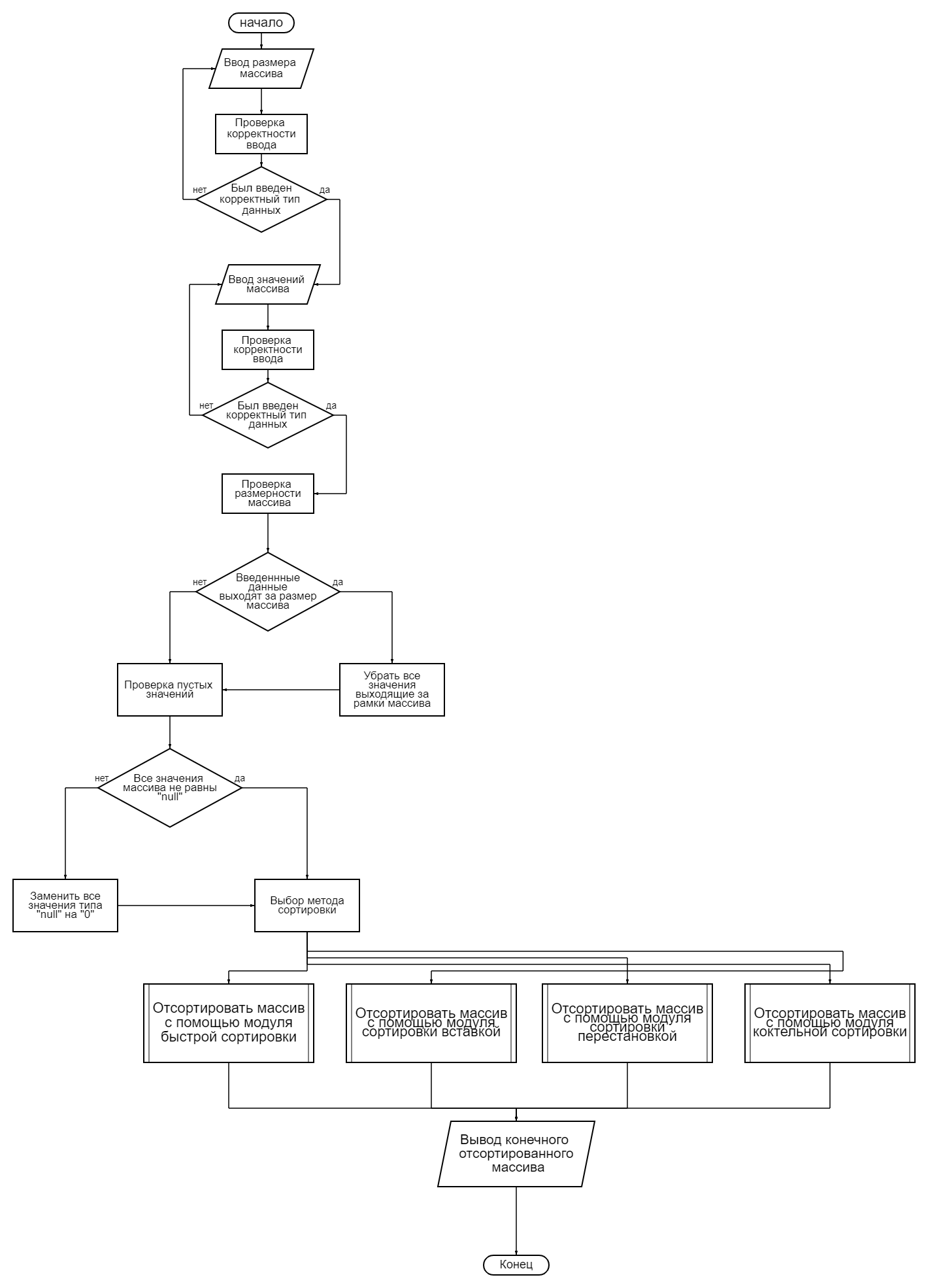
На данном рисунке представлена диаграмма потока данных в которой показывается какой именно путь данные проходят от начала программы до конечного результата, а именно: Отсортированного массива чисел.

Рисунок 6 – Функциональная схема продукта

На данном рисунке представлена функциональная схема продукта где наглядно описана функциональная составляющая программного продукта

Рисунок 7 – Блок схема программы



На данном рисунке представлена блок схема реализованной программы где наглядно описана логика программного продукта со всеми необходимыми этапами.

**ГЛАВА 2. МОДУЛИ ПРОГРАММЫ**

2.1 Модуль интерфейса

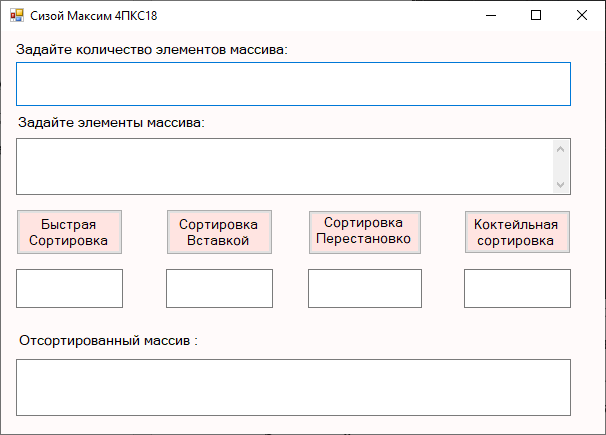


Рисунок 8 – Модуль интерфейса

На данном рисунке представлен общий интерфейс программы, в нем реализованы такие кнопки как:” Быстрая Сортировка”,” Сортировка Вставкой”,” Сортировка Перестановкой», «Коктейльная сортировка”, а также окна: “Задайте количество элементов Массива:”, “Задайте элементы массива:”, “Отсортированный массив:”.

2.2 Модуль CoctailSort

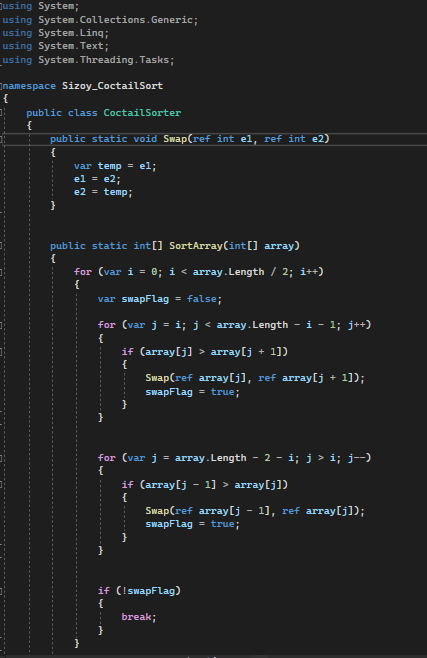


Рисунок 9 – Модуль CoctailSort

На данном рисунке представлена реализация метода коктейльной сортировки, а именно, весь необходимый для реализации внутри программы код.

Шейкерная сортировка (Cocktail sort), она же сортировка перемешиванием, она же двунаправленная сортировка — по сути всего лишь оптимизированный алгоритм [пузырьковой сортировки](https://medium.com/@alivander/%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0-%D0%BF%D1%83%D0%B7%D1%8B%D1%80%D1%8C%D0%BA%D0%BE%D0%BC-javascript-54462b2989a6). Напомним — ее суть была в том, что за каждый проход по массиву в его конец “всплывал” максимальный элемент.

А значит в основе опять лежит сравнение двух соседних элементов. Единственное отличие состоит лишь в том, что теперь это происходит в двух направлениях поочередно, постепенно сужая диапазон сортировки. В итоге за один проход в конец массива “всплывает” максимальный элемент из диапазона, а за следующий проход — в начало массива минимальный (мы рассматриваем сортировку по возрастанию). Эти элементы можно больше не рассматривать и таким образом диапазон сужается с двух сторон. [1]

2.3 Модуль QuickSort

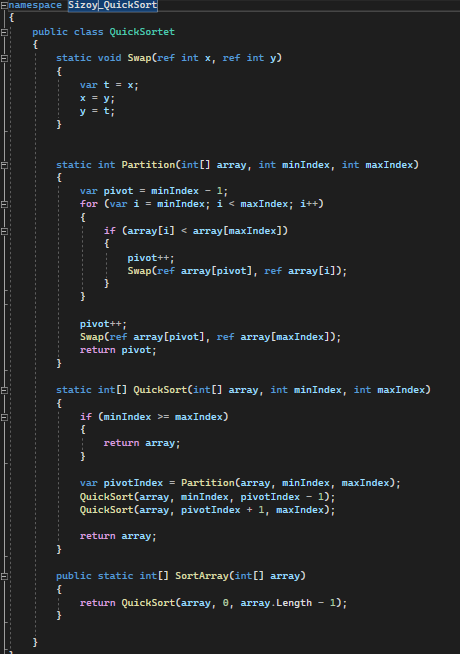


Рисунок 10 – Модуль QuickSort

На данном рисунке представлена реализация метода быстрой сортировки, а именно, весь необходимый для реализации внутри программы код.

Алгоритм быстрой сортировки является рекурсивным, поэтому для простоты процедура на вход будет принимать границы участка массива от l включительно и до r не включительно. Понятно, что для того, чтобы отсортировать весь массив, в качестве параметра l надо передать 0, а в качестве r — n, где по традиции n обозначает длину массива.  
  
В основе алгоритма быстрой сортировке лежит процедура partition. Partition выбирает некоторый элемент массива и переставляет элементы участка массива таким образом, чтобы массив разбился на 2 части: левая часть содержит элементы, которые меньше этого элемента, а правая часть содержит элементы, которые больше или равны этого элемента. Такой разделяющий элемент называется пивотом.  
  
Реализация partiion'а:

partition(l, r):

pivot = a[random(l ... r - 1)]

m = l

for i = l ... r - 1:

if a[i] < pivot:

swap(a[i], a[m])

m++

return m

Пивот в нашем случае выбирается случайным образом. Такой алгоритм называется рандомизированным. На самом деле пивот можно выбирать самым разным образом: либо брать случайный элемент, либо брать первый / последний элемент учаcтка, либо выбирать его каким-то «умным» образом. Выбор пивота является очень важным для итоговой сложности алгоритма сортировки, но об этом несколько позже. Сложность же процедуры partition — O(n), где n = r — l — длина участка.  
  
Теперь используем partition для реализации сортировки:  
  
Реализация partiion'а:

sort(l, r):

if r - l = 1:

return

m = partition(l, r)

sort(l, m)

sort(m, r)

Крайний случай — массив из одного элемента обладает свойством упорядоченности. Если массив длинный, то применяем partition и вызываем процедуру рекурсивно для двух половин массива.  
  
Если прогнать написанную сортировку на примере массива 1 2 2, то можно заметить, что она никогда не закончится. Почему так получилось?  
  
При написании partition мы сделали допущение — все элементы массива должны быть уникальны. В противном случае возвращаемое значение m будет равно l и рекурсия никогда не закончится, потому как sort(l, m) будет вызывать sort(l, l) и sort(l, m). Для решения данной проблемы надо массив разделять не на 2 части (< pivot и >= pivot), а на 3 части (< pivot, = pivot, > pivot) и вызывать рекурсивно сортировку для 1-ой и 3-ей частей. [2]

2.4 Модуль SwapSort

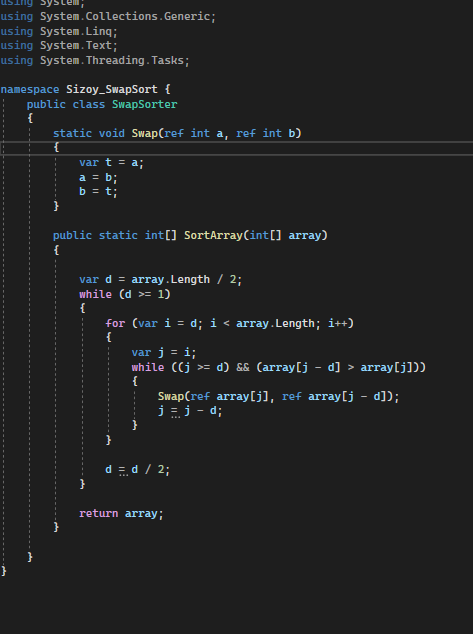


Рисунок 11 – Модуль SwapSort

На данном рисунке представлена реализация метода выборки, весь необходимый для реализации внутри программы код.

Сортировка простой выборкой. Данный метод реализует практически "дословно" сформулированную выше стратегию выборки. Порядок алгоритма простой выборки - O(N2). Количество пересылок - N. [3]

2.5 Модуль InsertionSort

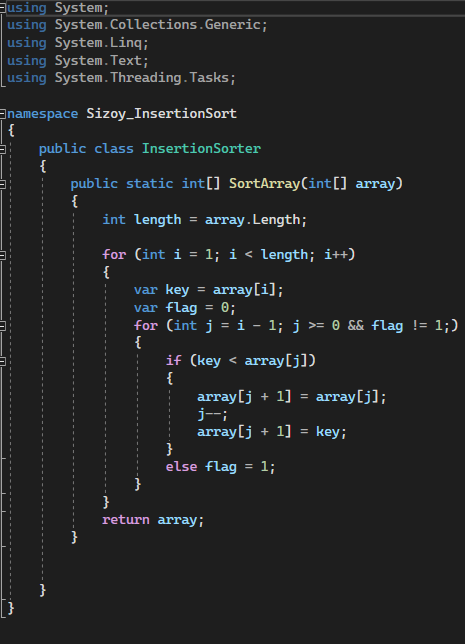


Рисунок 12 – Модуль InsertionSort

На данном рисунке представлена реализация метода вставки, весь необходимый для реализации внутри программы код.

Сортировка вставками - алгоритм, при котором каждый последующий элемент массива сравнивается с предыдущими элементами (отсортированными) и вставляется в нужную позицию.

Общая идея алгоритма: Сравниваем второй элемент с первым элементом массива и при необходимости меняем их местами. Условно эти элементы (первый и второй) будут являться отсортированным массивом, остальные элементы - неотсортированным. [4]

Сортировка массива — расположение его элементов в некотором заданном порядке. В отсортированном массиве поиск элемента можно осуществлять, не просматривая весь массив. Например, в случае сортировки в порядке возрастания минимальный элемент массива всегда будет находиться на первом месте. Задача сортировки, как и любая другая задача, может решаться множеством способов, каждый из которых имеет как достоинства, так и недостатки. [5]

Первые прототипы современных методов сортировки появились уже в XIX веке.

К 1890 году для ускорения обработки данных [переписи населения](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BF%D0%B8%D1%81%D1%8C_%D0%BD%D0%B0%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B5%D0%B4%D0%B8%D0%BD%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%A8%D1%82%D0%B0%D1%82%D1%8B_%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B8) американец [Герман Холлерит](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%80%D0%BC%D0%B0%D0%BD_%D0%A5%D0%BE%D0%BB%D0%BB%D0%B5%D1%80%D0%B8%D1%82) создал первый статистический [табулятор](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80) — электромеханическую машину, предназначенную для автоматической обработки информации, записанной на [перфокартах](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%84%D0%BE%D0%BA%D0%B0%D1%80%D1%82%D0%B0). У машины Холлерита имелся специальный «сортировальный ящик» из 26 внутренних отделений.

При работе с машиной от оператора требовалось вставить перфокарту и опустить рукоятку. Благодаря пробитым на перфокарте отверстиям замыкалась определённая [электрическая цепь](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%86%D0%B5%D0%BF%D1%8C), и на единицу увеличивалось показание связанного с ней циферблата. Одновременно с этим открывалась одна из 26 крышек сортировального ящика, и в соответствующее отделение перемещалась перфокарта, после чего крышка закрывалась.

Данная машина позволила обрабатывать около 50 карт в минуту, что ускорило обработку данных в 3 раза. К переписи населения 1900 года Холлерит усовершенствовал машину, автоматизировав подачу карт. Работа сортировальной машины Холлерита основывалась на методах [поразрядной сортировки](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0). В патенте на машину обозначена сортировка «по отдельности для каждого столбца», но не определён порядок. В другой аналогичной машине, запатентованной в 1894 году Джоном Гором, упоминается сортировка со столбца десятков. Метод сортировки, начиная со столбца единиц, впервые появляется в литературе в конце 1930-х годов. К этому времени сортировальные машины уже позволяли обрабатывать до 400 карт в минуту.

К 1952 году на практике уже применялись многие методы внутренней сортировки, но теория была развита сравнительно слабо. В октябре 1952 года Даниэль Гольденберг привёл пять методов сортировки с анализом наилучшего и наихудшего случаев для каждого из них. В 1954 году Гарольд Сьюворд развил идеи Гольденберга, а также проанализировал методы внешней сортировки. Говард Демут в 1956 году рассмотрел три абстрактные модели задачи сортировки: с использованием циклической памяти, линейной памяти и памяти с произвольным доступом. Для каждой из этих задач автор предложил оптимальные или почти оптимальные методы сортировки, что помогло связать теорию с практикой. Из-за малого числа людей, связанных с вычислительной техникой, эти доклады не появлялись в «открытой литературе». Первой большой обзорной статьёй о сортировке, появившейся в печати в 1955 году, стала работа Дж. Хоскена, в которой он описал всё имевшееся на тот момент оборудование специального назначения и методы сортировки для ЭВМ, основываясь на брошюрах фирм-изготовителей. В 1956 году Э. Френд в своей работе проанализировал математические свойства большого числа алгоритмов [внутренней](https://ru.wikipedia.org/wiki/%D0%92%D0%BD%D1%83%D1%82%D1%80%D0%B5%D0%BD%D0%BD%D1%8F%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0) и [внешней сортировки](https://ru.wikipedia.org/wiki/%D0%92%D0%BD%D0%B5%D1%88%D0%BD%D1%8F%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0), предложив некоторые новые методы.

После этого было предложено множество различных алгоритмов сортировки: например, вычисление адреса в 1956 году; слияние с вставкой, [обменная поразрядная сортировка](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0), каскадное слияние и [метод Шелла](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%A8%D0%B5%D0%BB%D0%BB%D0%B0) в 1959 году, [многофазное слияние](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D0%B5%D0%BC) и вставки в дерево в 1960 году, осциллирующая сортировка и [быстрая сортировка Хоара](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0) в 1962 году, [пирамидальная сортировка Уильямса](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%80%D0%B0%D0%BC%D0%B8%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0) и обменная сортировка со слиянием Бэтчера в 1964 году. В конце 60-х годов произошло и интенсивное развитие теории сортировки. Появившиеся позже алгоритмы во многом являлись вариациями уже известных методов. Получили распространение адаптивные методы сортировки, ориентированные на более быстрое выполнение в случаях, когда входная последовательность удовлетворяет заранее установленным критериям.

**ЗАКЛЮЧЕНИЕ**

Я, Сизой Максим Андреевич, проходил учебную практику на базе: Таврического колледжа (структурное подразделение) ФГАОУ «КФУ им. В.И. Вернадского».

Дата начала практики: 16 марта 2023 г.

Дата окончания практики: 22 марта 2023 г.

Дата сдачи отчёта по практике: 22 марта 2023 г.

Была выполнена следующая цель практики:

Формирование и развитие общих и профессиональных компетенций по модулю ПМ.03 Участие в интеграции программных модулей.

Были выполнены следующие задачи учебной практики:

Закрепление навыков разработки программного обеспечения;

Использование методов для получения кода с заданной функциональностью и степенью качества;

Разработка документации на программный продукт;

Знание моделей процесса разработки программного обеспечения, основных принципов процесса разработки программного обеспечения;

Знание основных подходов к интегрированию программных модулей, основных методов и средств эффективной разработки ПО;

Был выполнен весь план задания для выполнения:

1. Необходимо разработать программный комплекс по демонстрации работы алгоритмов сортировки массивов данных (реализовать не менее 4 алгоритмов сортировки, которые выбрать самостоятельно):
   1. Разработать техническое задание на программный продукт.
   2. Разработать спецификацию на программный продукт.
   3. Разработать функциональную диаграмму программного продукта, диаграмму потоков данных программных модулей продукта.
   4. Разработать функциональную схему программного продукта, составить блок-схемы программных модулей программного продукта.
   5. Разработать коды программных модулей программного продукта.
   6. Разработать пользовательский интерфейса программного продукта в визуальной среде.
   7. Выполнить интеграцию программных модулей в программный продукт.
   8. Разработать процедуру тестирования программного продукта. Выполнить тестирование программного продукта. Результат тестирования оформить протоколом тестирования.
   9. Разработать справочную систему программного продукта.
   10. Разработать руководства оператора (пользователя).
2. Создать аккаунт в GitHub. Создать папку проекта. В папку загрузить разработанный программный комплекс, всю разработанную документацию к проекту (п.п.1.1 – 1.10).
3. Составить отчет о выполнении.

Ссылка на репозиторий с выполненной работой:

[https://github.com/Zibulya](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2FZibulya&cc_key=)

**СПИСОК ЛИТЕРАТУРЫ**

1. Мамулов А.О. Коктейльная сортировка [электронный ресурс] medium.com – URL: https://medium.com/@alivander/шейкерная-сортировка-javascript-a2b8af562ee. – (дата обращения 20.03.2023)
2. Бельчиков М.Н. Сортировка массивов методом быстрой сортировки [электронный ресурс] techiedelight.com – URL: https://www.techiedelight.com/ru/quicksort/. – (дата обращения 20.03.2023)
3. Мельников А.И. Методы сортировки [электронный ресурс] sumk.ulstu.ru – URL: http://sumk.ulstu.ru/docs/mszki/Zavgorodnii/9.3.2.html. - (дата обращения 20.03.2023)
4. Березов М.Н. Сортировка вставками [электронный ресурс] study.urfu.ru – URL: https://study.urfu.ru/Aid/Publication/7482/1/Baraz.pdf. - (дата обращения 20.03.2023)
5. Пильщиков М.К. Выборочный метод статистического анализа [электронный ресурс] habr.com – URL: [https://habr.com/ru/post/415935/](https://vk.com/away.php?to=https%3A%2F%2Fhabr.com%2Fru%2Fpost%2F415935%2F&cc_key=). - (дата обращения 20.03.2023)

**ПРИЛОЖЕНИЕ А**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Diagnostics;

using Sizoy\_QuickSort;

using Sizoy\_SwapSort;

using Sizoy\_CoctailSort;

using Sizoy\_InsertionSort;

namespace ArraySorter

{

public partial class Form1 : Form

{

int n = 0;

public Form1()

{

InitializeComponent();

}

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

{

var sw = new Stopwatch();

sw.Start();

string myArr = textBox4.Text;

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

string texter = string.Join("",CoctailSorter.SortArray(result));

textBox3.Text = texter;

sw.Stop();

textBox7.Text = sw.Elapsed.ToString();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

textBox4.Text = "";

try

{

n = Convert.ToInt32(textBox1.Text);

}

catch

{

textBox1.Text = "";

MessageBox.Show(

"Неверный тип данных",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1);

}

int[] x = new int[n];

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

{

textBox4.Text = textBox4.Text + x[i];

}

}

private void button2\_Click(object sender, EventArgs e)

{

var sw = new Stopwatch();

sw.Start();

string myArr = textBox4.Text;

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

string texter = string.Join("", SwapSorter.SortArray(result));

textBox3.Text = texter;

sw.Stop();

textBox6.Text = sw.Elapsed.ToString();

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

int[] x = new int[n];

if (x.Length > textBox4.TextLength)

{

int a = x.Length - textBox4.TextLength;

while (a > 0)

{

a--;

textBox4.Text = textBox4.Text + 0;

}

}

else

if (x.Length < textBox4.TextLength)

{

int a = textBox4.TextLength - x.Length;

textBox4.Text = textBox4.Text.Remove(textBox4.Text.Length - a);

a = 0;

}

}

private void button3\_Click(object sender, EventArgs e)

{

var sw = new Stopwatch();

sw.Start();

string myArr = textBox4.Text;

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

int resl = result.Length;

string texter = string.Join("", InsertionSorter.SortArray(result));

sw.Stop();

textBox5.Text = sw.Elapsed.ToString();

textBox3.Text = texter;

}

private void button4\_Click(object sender, EventArgs e)

{

var sw = new Stopwatch();

sw.Start();

string myArr = textBox4.Text;

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

int resl = result.Length;

string texter = string.Join("", QuickSortet.SortArray(result));

sw.Stop();

textBox2.Text = sw.Elapsed.ToString();

textBox3.Text = texter;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

}

private void textBox7\_TextChanged(object sender, EventArgs e)

{

}

}

}