|  |
| --- |
| Министерство науки и высшего образования Российской Федерации |
| федеральное государственное бюджетное образовательное учреждение высшего образования |
| "Российский экономический университет имени Г.В. Плеханова" |
| **МОСКОВСКИЙ  ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ** |

МДК 01.04 Системное программирование

ПМ 01 Разработка программного обеспечения для компьютерных систем

**Специальность: 09.02.07 «Информационные системы и программирование»**

Квалификация: программист

|  |  |
| --- | --- |
| Выполнил студент: | Проверил преподаватель: |
| Группы П50-4-21 | Пахомов Д. А. |
| Игошев Р.В. | «\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 года |
| «\_\_» января 2023 год | Оценка: \_\_\_\_\_\_\_\_\_ |
|  |  |

**ОГЛАВЛЕНИЕ**

[ПРАКТИЧЕСКАЯ РАБОТА №1 3](#_Toc135667031)

[Выбор действия 3](#_Toc135667032)

[Сложение 4](#_Toc135667033)

[Вычитание 5](#_Toc135667034)

[Умножение 5](#_Toc135667035)

[Целочисленное деление 6](#_Toc135667036)

[Остаток от деления 7](#_Toc135667037)

[Число Фиббоначи 8](#_Toc135667038)

[Выход и блок default 8](#_Toc135667039)

[ПРАКТИЧЕСКАЯ РАБОТА №2 9](#_Toc135667040)

[Код программы: 9](#_Toc135667041)

[Результаты работы: 14](#_Toc135667042)

[ПРАКТИЧЕСКАЯ РАБОТА №3 19](#_Toc135667043)

[ПРАКТИЧЕСКАЯ РАБОТА №6 26](#_Toc135667044)

# ПРАКТИЧЕСКАЯ РАБОТА №1

Тема: Калькулятор

Цель работы: научиться базовым функциям языка программирования C++, узнать структуру кода, создать консольный калькулятор.

Ход работы:

Для начала следует запросить у пользователя действие, предварительно выведя их ассортимент. Кроме того, в данной программе будут присутствовать проверки на неверный ввод данных. Калькулятор сделан цикличным.

## Выбор действия

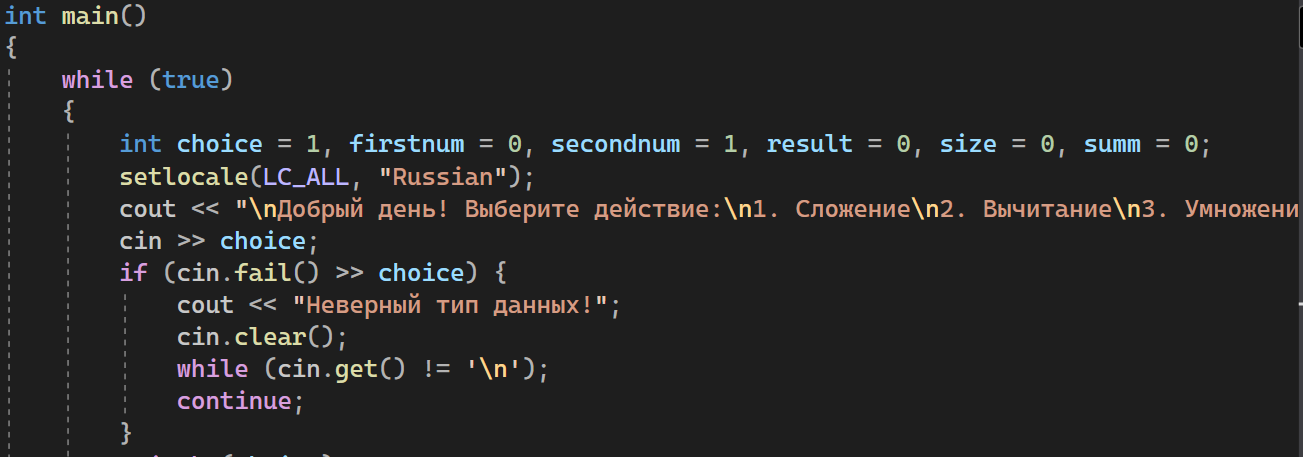


Рисунок 1 – Выбор действия

После выбора действий пользователем, необходимо проверять значение выбора и выполнять действия в зависимости от него. Первое на очереди – сложение.

## Сложение

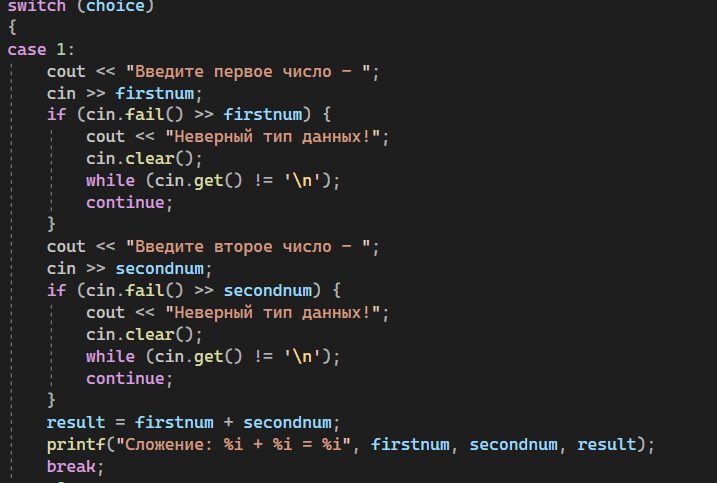


Рисунок 2 – Сложение

После сложения (где все еще присутствует защита от ввода неверного типа данных), приступаем к созданию вычитания.

## Вычитание

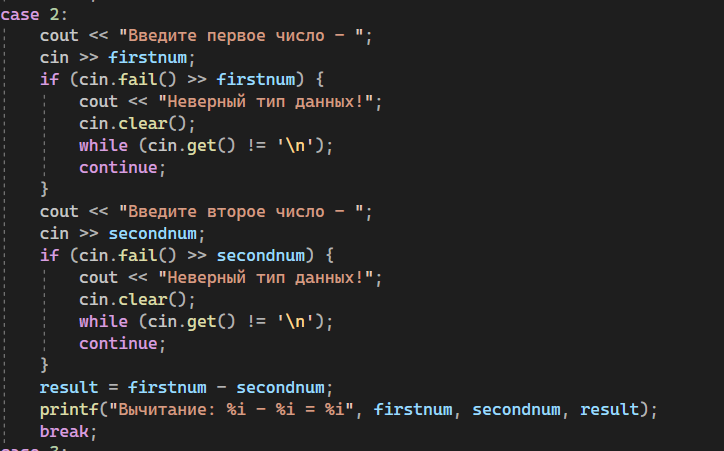


Рисунок 3 – Вычитание

После вычитания следует третье действие – умножение.

## Умножение

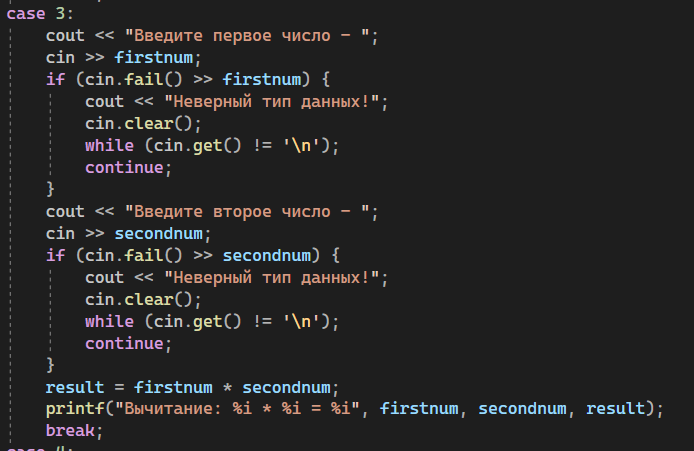


Рисунок 4 – Умножение

После умножение реализуем один из подвидов деления: целочисленное.

## Целочисленное деление

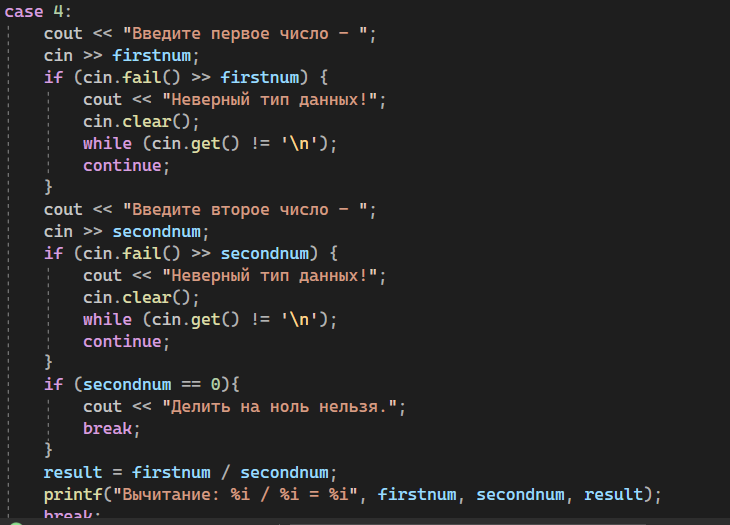


Рисунок 5 – Деление нацело

Теперь, попробуем создать второй вид деления, а вернее, остаток от деления:

## Остаток от деления

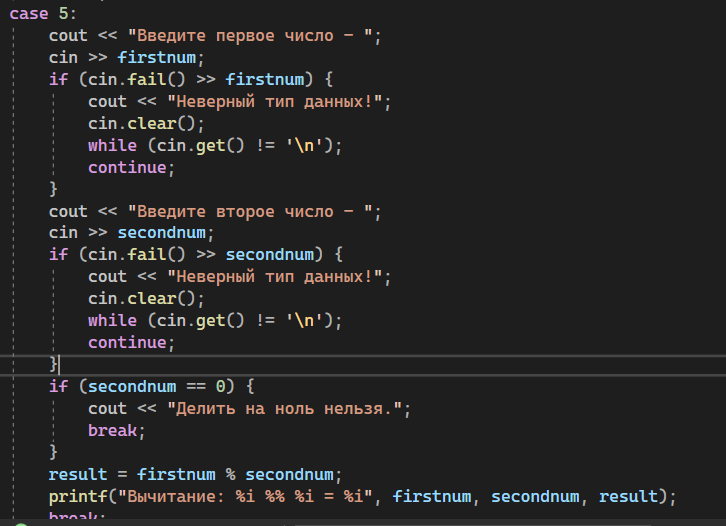


Рисунок 6 – Остаток от деления

После, попробуем найти число Фиббоначи. Это не простое выражение, что значит, что для этого придётся придумать алгоритм выполнения действий в нашем коде. В данном случае я использовал цикл и несколько арифметических действий.

## Число Фиббоначи

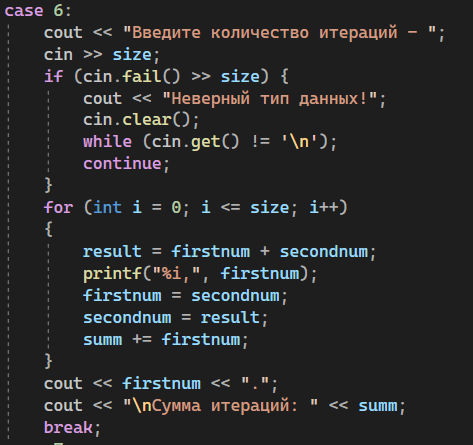


Рисунок 7 – Фиббоначи

Ну, и напоследок, самое простое – выход из программы и блок, выполняющийся в случае несоответствия выбора пользователя с имеющимися индексами возможным действий.

## Выход и блок default

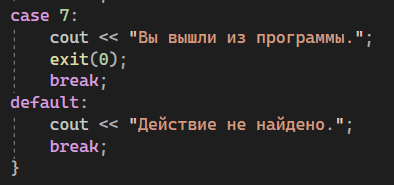


Рисунок 8 – Выход из программы и блок несовпадения значений

Вывод: научились базовым функциям языка программирования C++, узнали структуру кода, создали консольный калькулятор.

# ПРАКТИЧЕСКАЯ РАБОТА №2

Тема: «Указатели, Функции, Массивы»

Цель работы: научиться работать с указателями, функциями и массивами на языке программирования C++. Решить задачи, призванные закрепить навыки работы с вышеуказанными разделами языка.

## Код программы:

Для начала следует реализовать консольное меню, из которого будут вызываться все остальные функции.

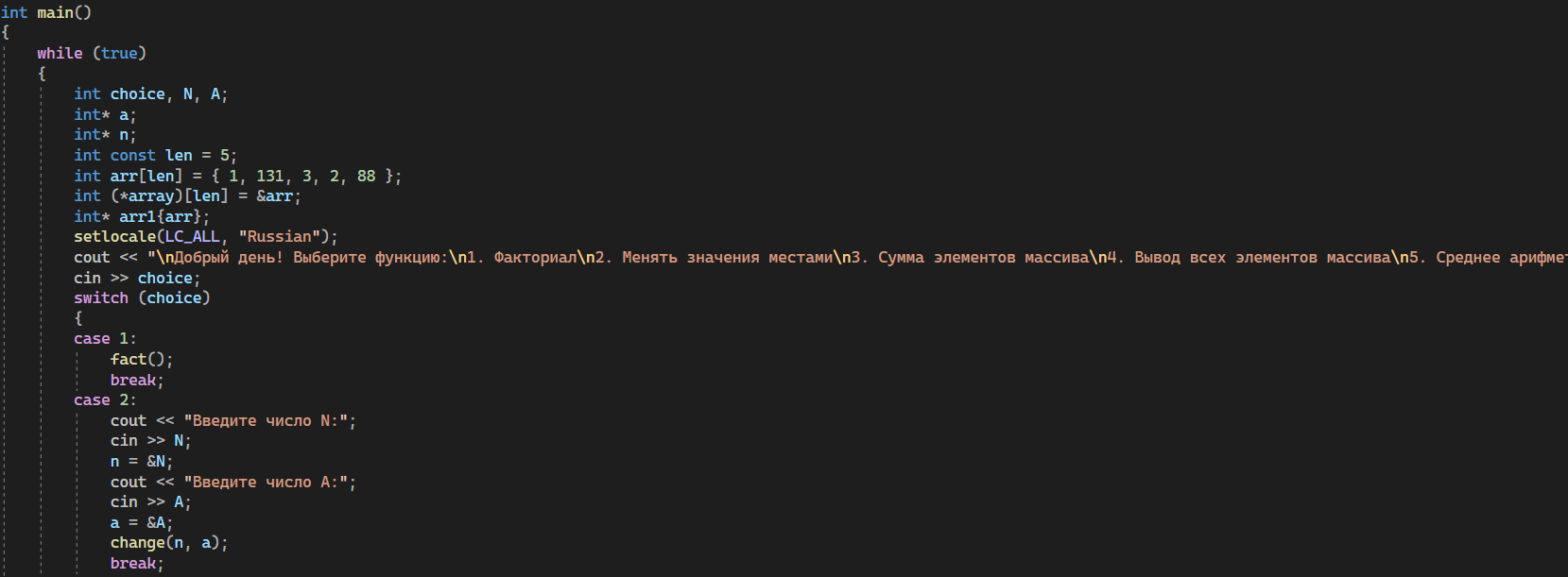


Рисунок 9 – Консольное меню

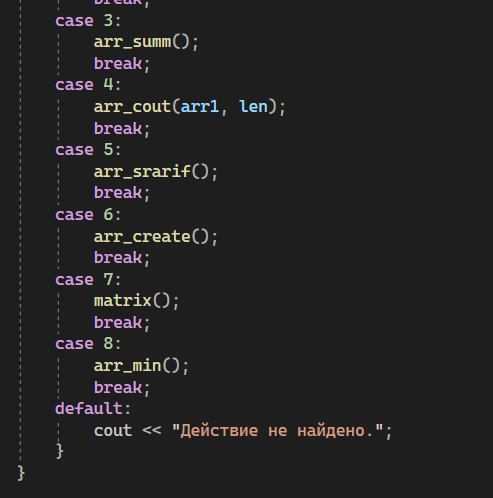


Рисунок 10 – Консольное меню

Реализуем расчет факториала:

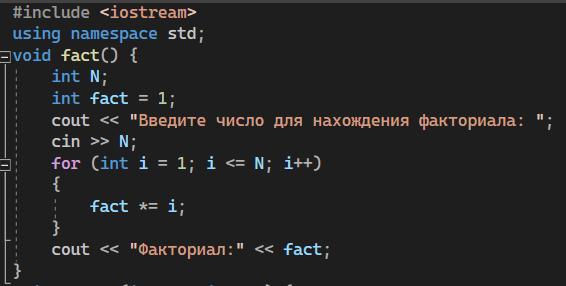


Рисунок 11 – Факториал

Теперь, изменение значений переменных через указатели.

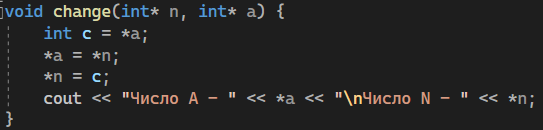


Рисунок 12 – Изменение значений переменных

Теперь, счетчик суммы элементов массива.

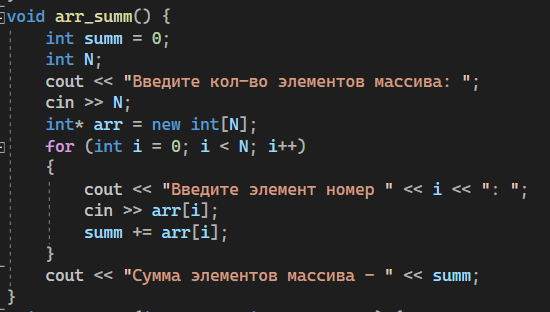


Рисунок 13 – Создание массива и их сумма

Теперь, вывод массива по его указателю.

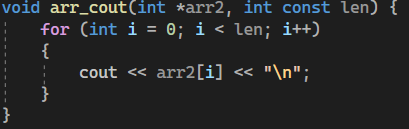


Рисунок 14 – Вывод элементов переданного указателя

Далее: среднее арифметическое массива.

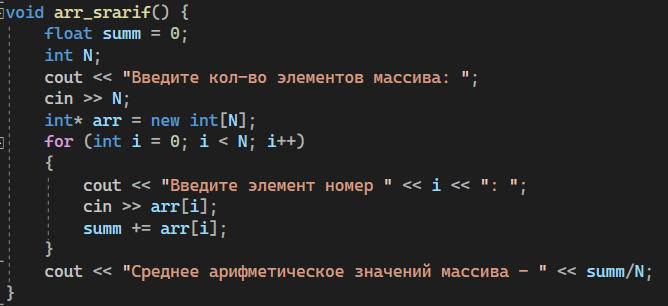


Рисунок 15 – Среднее арифметическое значений массива

Теперь, создание и вывод массива.

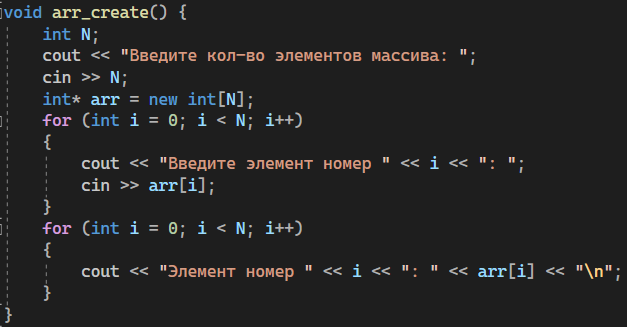


Рисунок 16 – Создание массива и вывод его

Теперь, попробуем реализовать функцию случайного заполнения двумерного массива и счета его суммы.

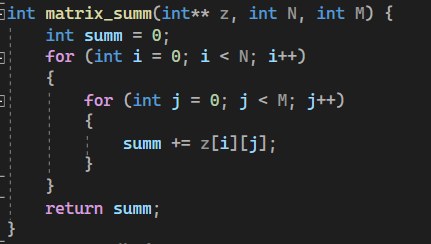


Рисунок 17 – Сумма матрицы

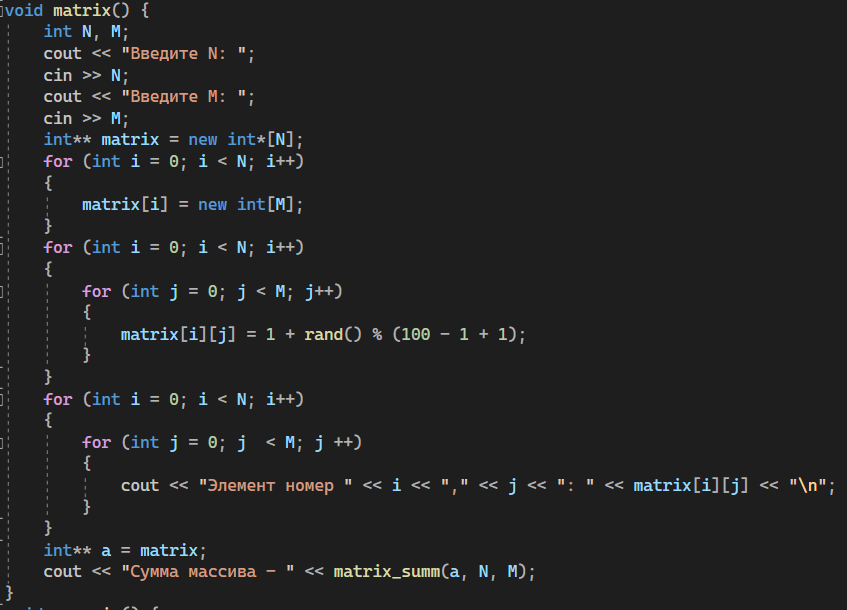


Рисунок 18 – Случайное заполнение двумерного массива и его сумма

Последнее задание это нахождение минимального числа в массиве.

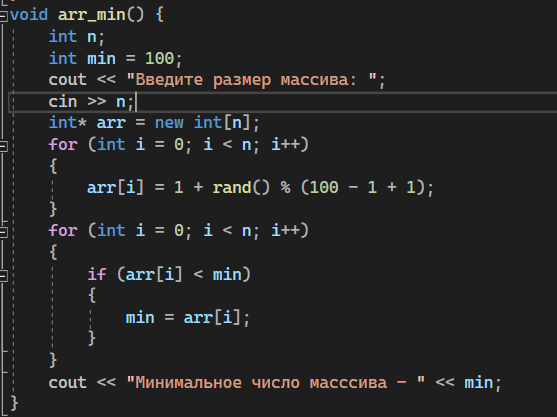


Рисунок 19 – Минимальное число массива

## Результаты работы:

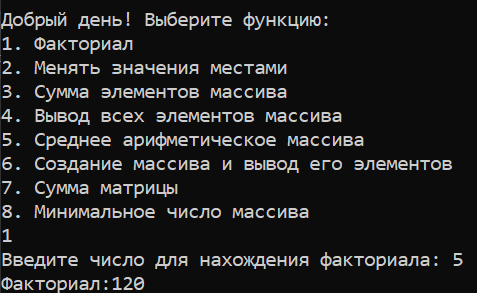


Рисунок 20 - Факториал

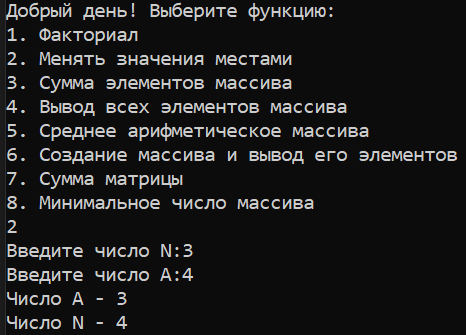


Рисунок 21 – Изменение значений переменных

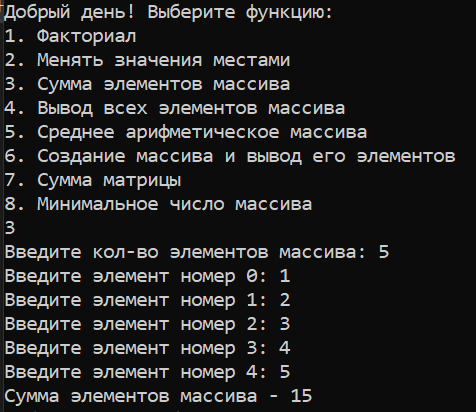


Рисунок 22 – Сумма элементов массива

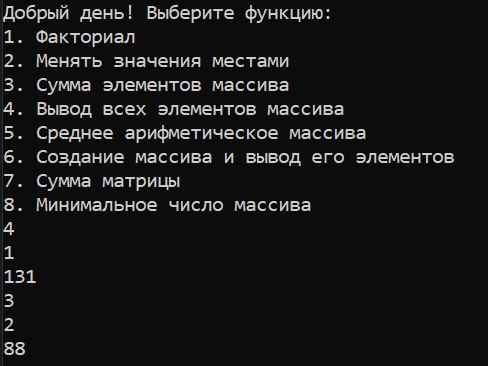


Рисунок 23 – Вывод элементов массива

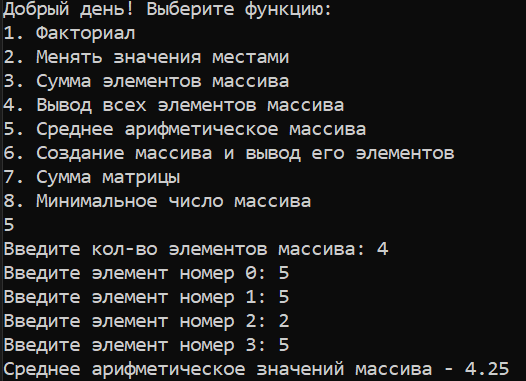


Рисунок 24 – Среднее арифметическое значений массива



Рисунок 25 – Создание массива и вывод его элементов

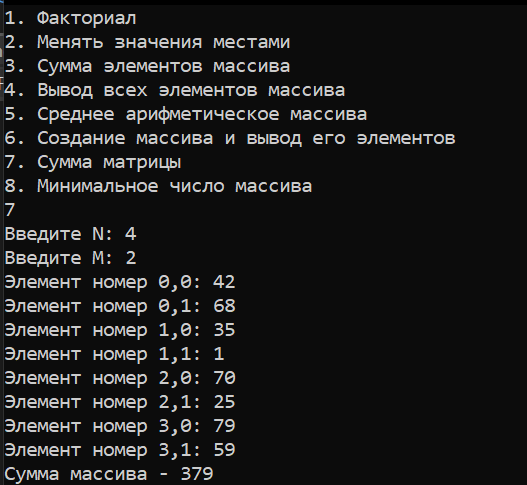


Рисунок 26 – Случайное заполнение матрицы и ее сумма

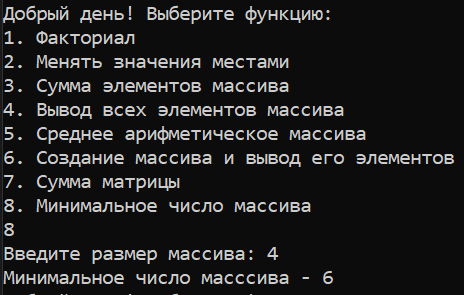


Рисунок 27 – Минимальное число массива, заполненного случайными числами

Вывод: научились работать с указателями, функциями и массивами на языке программирования C++. Решили задачи, призванные закрепить навыки работы с вышеуказанными разделами языка.

# ПРАКТИЧЕСКАЯ РАБОТА №3

Тема: «Структуры, классы»

Цель работы: решить задание, попутно научившись работать со структурами, классами и ООП в языке C++.

Вначале, создадим структуру:

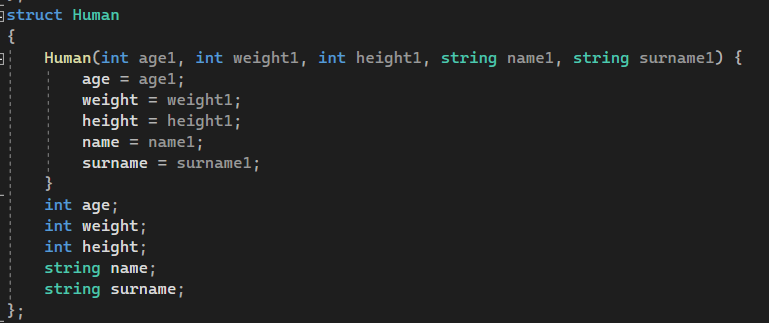


Рисунок 28 – Структура

Теперь, реализуем класс, от которого будет наследоваться будущий класс.

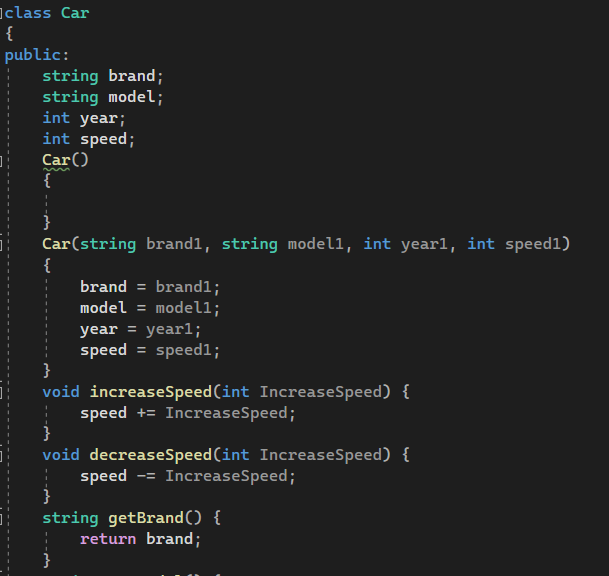


Рисунок 29 – Родительский класс

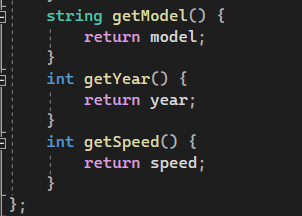


Рисунок 30 – Родительский класс

Настало время и реализовать дочерний класс, принимающий родительский класс Car.

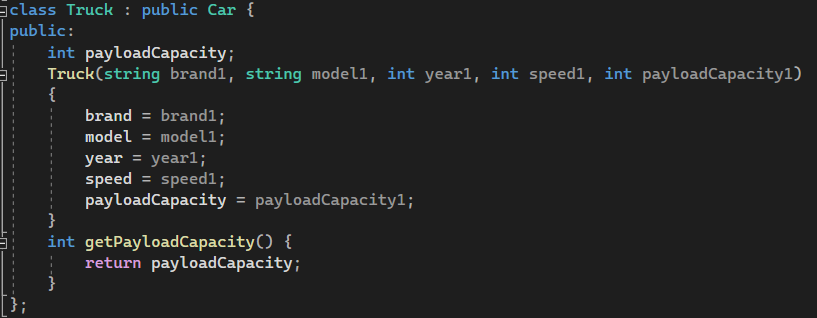


Рисунок 31 – Дочерний класс

Теперь попробуем поработать с нашими структурами и классами, создавая их экземпляры и всячески их сортируя и выводя.

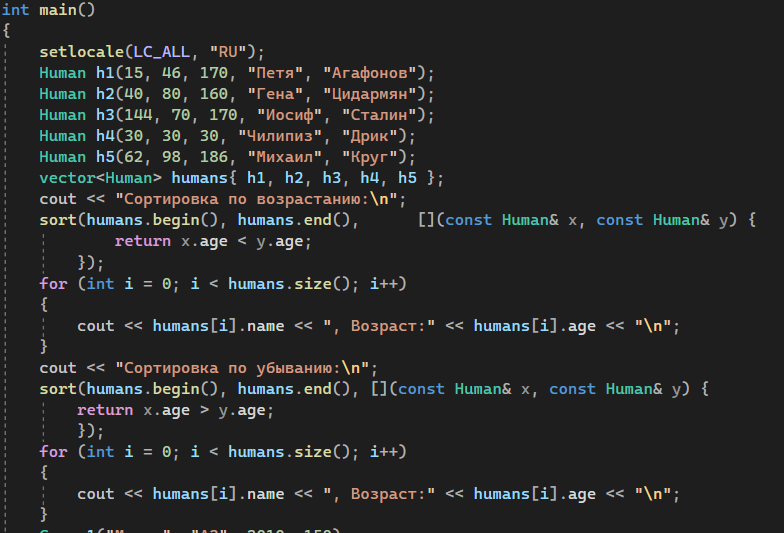


Рисунок 32 – Работа с классами и структурами

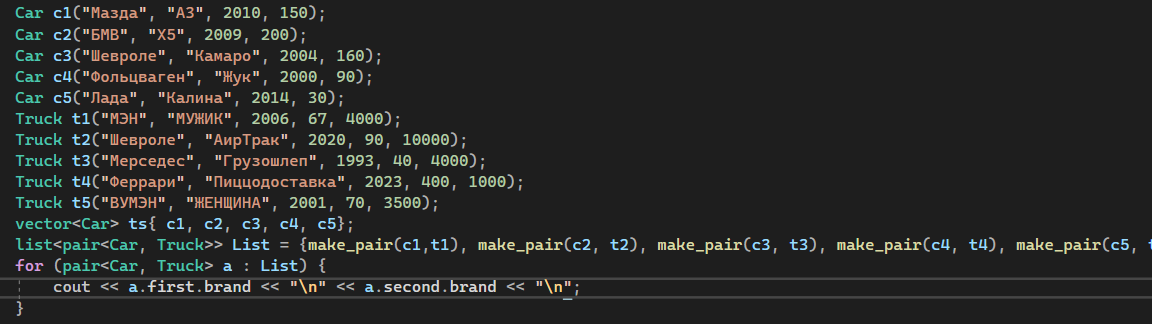


Рисунок 33 – Работа с классами и структурами

Результат работы:



Рисунок 34 – Результат работы

Вывод: решили задание, попутно научившись работать со структурами, классами и ООП в языке C++.

# ПРАКТИЧЕСКАЯ РАБОТА №4

«Файлы, json»

Цель работы: научиться работать с файлами и json в языке C++.

Для начала, реализуем функцию main, из которой будут вызываться все остальные части нашей программы.

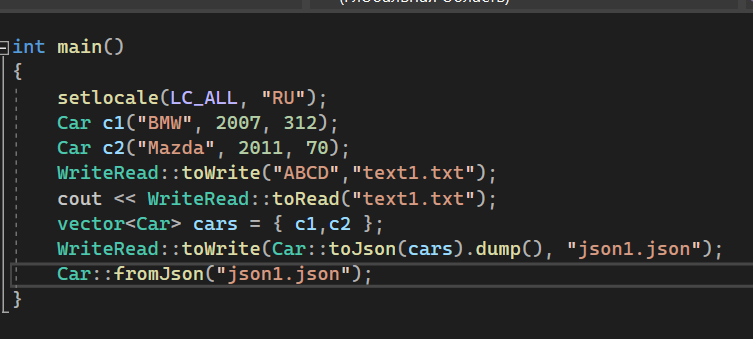


Рисунок 35 – Функция Main

Теперь – создадим класс для работы с файлами. У него будет две функции: на чтение файла и на запись в файл.

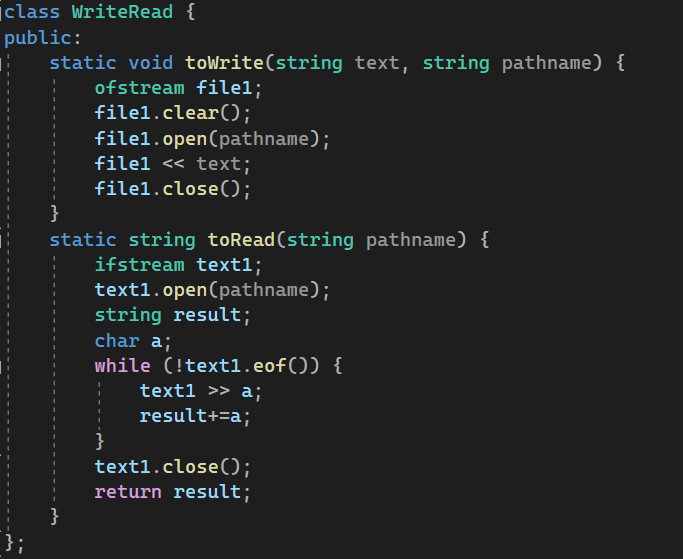


Рисунок 36 – Класс WriteRead

На очереди – класс Car, с чьими данными мы и будем работать в процессе работы с json.

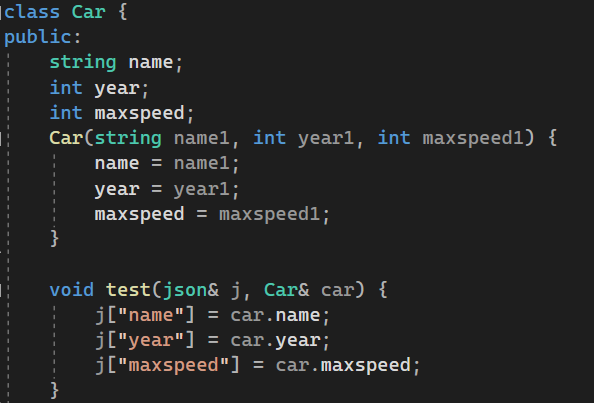


Рисунок 37 – Класс Car

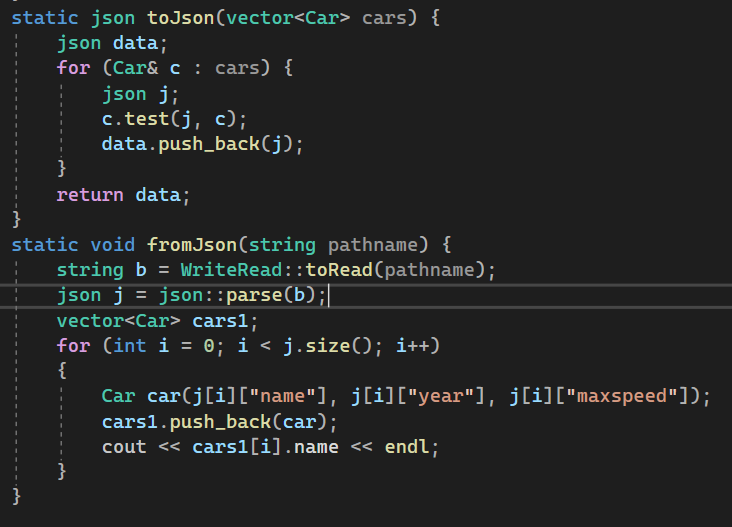


Рисунок 38 – Функции класса Car

Вывод: научились работать с файлами и json в языке C++.

# ПРАКТИЧЕСКАЯ РАБОТА №5

Тема «UDP/TCP»

Цель: научиться работать с протоколами UDP и TCP используя язык программирования C++ и выполнив задание, настроенное на отработку знаний по этой теме.

Для начала реализуем клиента. Здесь будет вызов функций для обмена сообщениями:

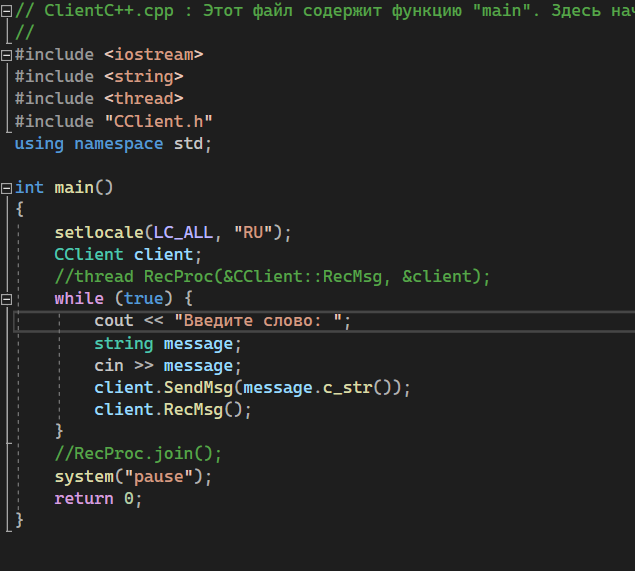


Рисунок 39 – Функция main клиента

Разметка класса клиента выглядит следующим образом:

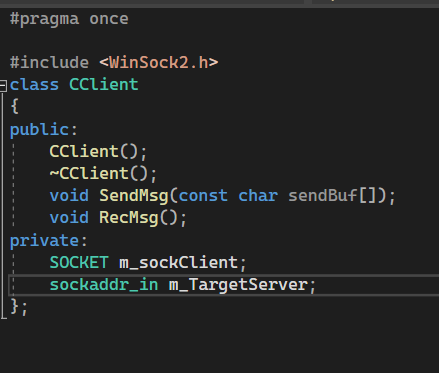


Рисунок 40 – Разметка класса клиента

Теперь, добавим логику самого класса:

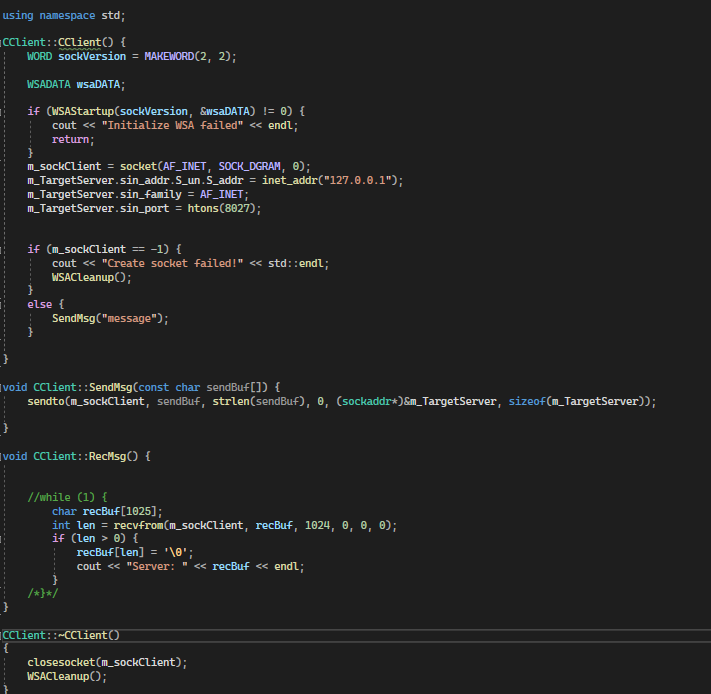


Рисунок 41 – Выполнение функций клиента

Ну, и начнем работать с сервером:

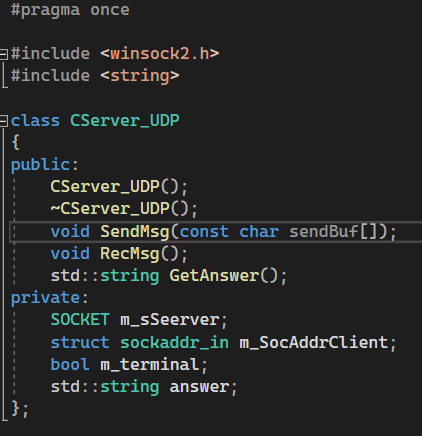


Рисунок 42 – Разметка класса сервера

Логика:

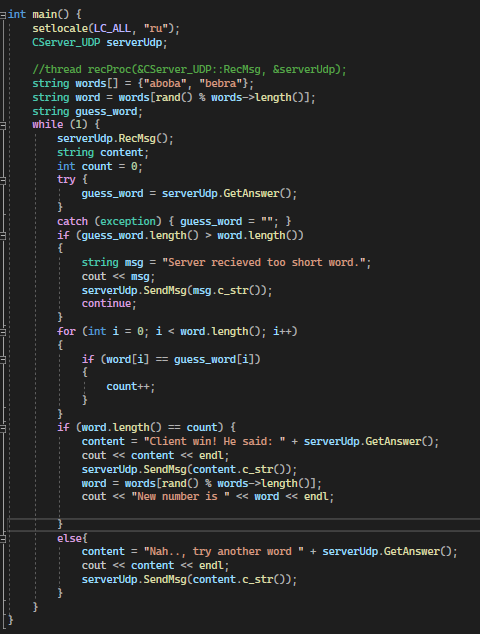


Рисунок 43 – Функция main сервера

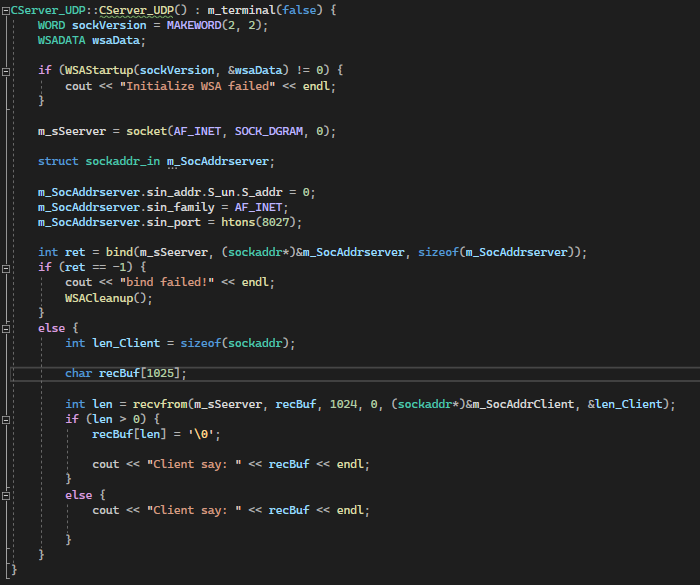


Рисунок 44 – Функция main сервера



Рисунок 45 – Функции класса серверов

Ну, и теперь, попробуем запустить нашу программу и посмотреть на результат:

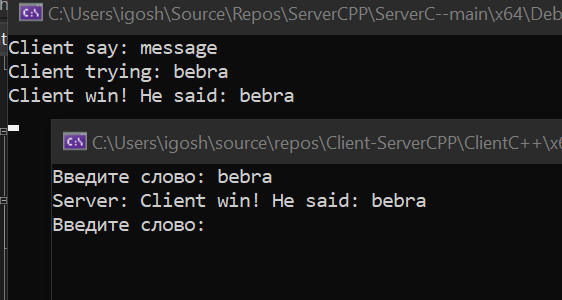


Рисунок 46 – Результат работы

Вывод: научились работать с протоколами UDP и TCP используя язык программирования C++ и выполнив задание, настроенное на отработку знаний по этой теме.

ПРАКТИЧЕСКАЯ РАБОТА №6

Тема: «Библиотеки»

Цель работы: научиться работать с библиотеками на языке С++, выполнив практическое задание, затрагивающее динамические и статические библиотеки.

Рассмотрим структуру проекта:

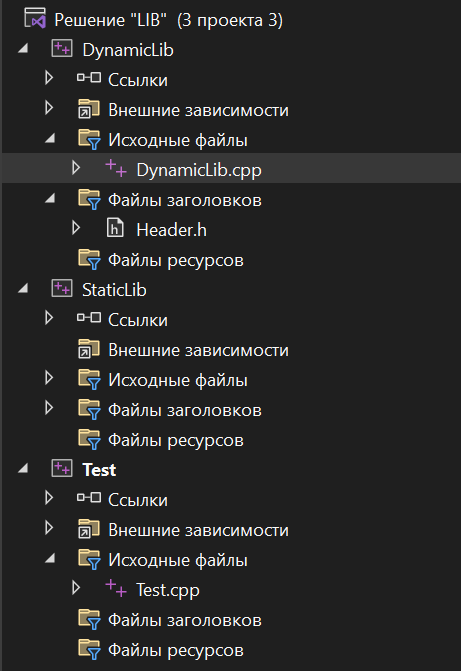


Рисунок 47 – Структура проекта

Начнем с динамической библиотеки.

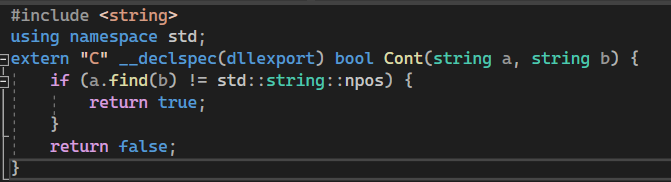


Рисунок 48 – Функции динамической библиотеки

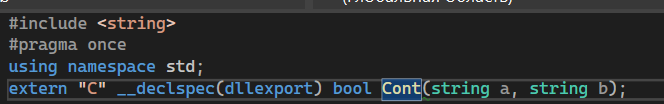


Рисунок 49 – Разметка динамической библиотеки

Теперь - статическая

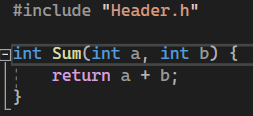


Рисунок 50 – Функции статической библиотеки



Рисунок 51 – Разметка статической библиотеки

Теперь же создадим функцию, из которой будут использоваться все созданные библиотеки.

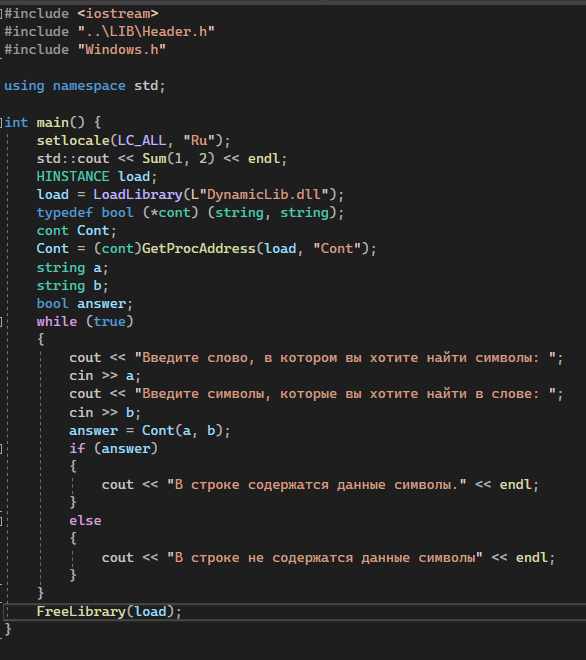


Рисунок 52 – Функции основного файла

Результат работы:

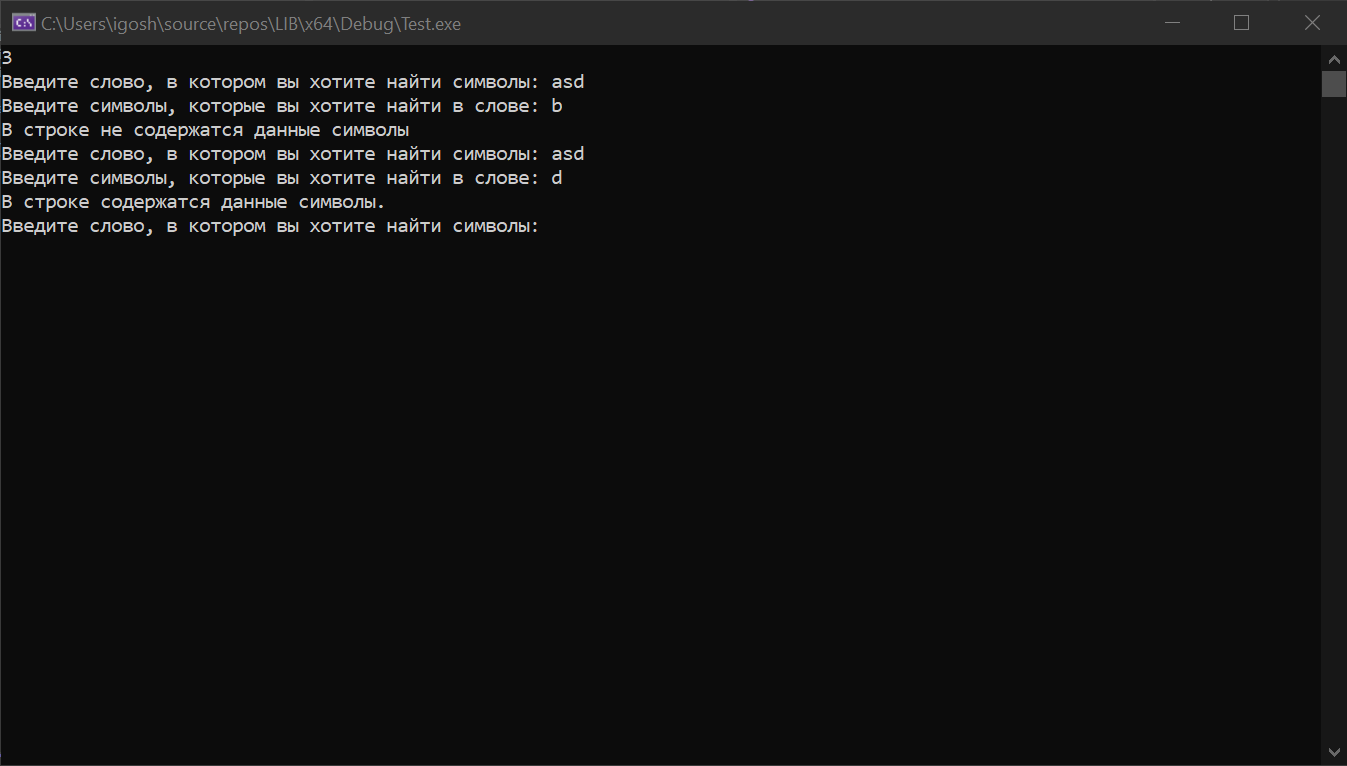


Рисунок 53 – Результат работы

Вывод: научились работать с библиотеками на языке С++, выполнив практическое задание, затрагивающее динамические и статические библиотеки.

ПРАКТИЧЕСКАЯ РАБОТА №6

Тема: «Умные указатели. unique\_ptrT. shared\_ptrT. Обработка исключений. Вложенные try-catch. Создание своих типов исключений. Тип exception. Типы исключений.»

Цель работы: Научиться работать с умными указателями в языке C++, с исключениями, их вложениями и типами исключений, выполнив практическое задание для отработки вышеуказанных знаний.

Задание:

* Создайте свой тип исключения с названием PersonException, который будет использоваться для обработки ошибок, связанных с данными о человеке.
* Создайте класс Person с приватными полями: имя (string), возраст (int), идентификатор (int).
* В классе Person создайте конструктор, который принимает значения для имени, возраста и идентификатора, и инициализирует соответствующие поля объекта.
* В конструкторе класса Person проверьте корректность переданных данных. Если данные некорректны, бросьте исключение типа PersonException.
* Создайте функцию main, которая будет запрашивать данные о человеке у пользователя, создавать объект Person и выводить информацию о нем на экран.
* Используйте умные указатели для хранения объекта Person.
* Вложенными try-catch блоками обработайте ошибки, которые могут возникнуть при создании объекта Person и при работе с умными указателями.

Выполнение задания:

Для начала реализуем свой тип ошибки. Называться он будет PersonException.

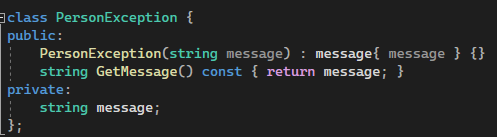


Рисунок 54 – Свой тип ошибок

Теперь – свой класс.

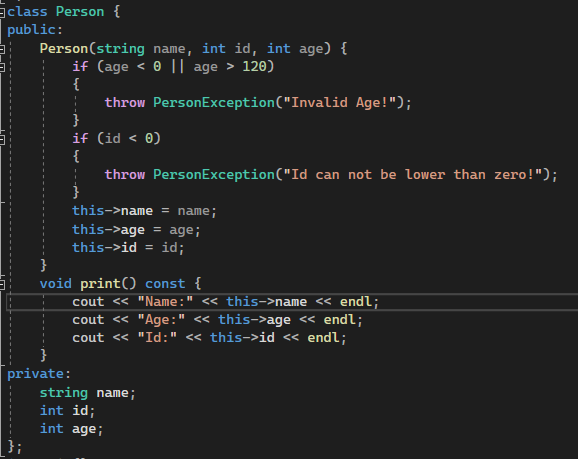


Рисунок 55 – Класс Person

Ну, и вызов всех классов в функции main.

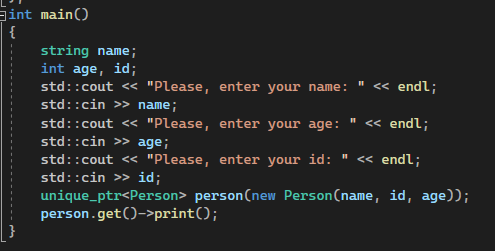


Рисунок 56 – Функция Main

Результаты работы:

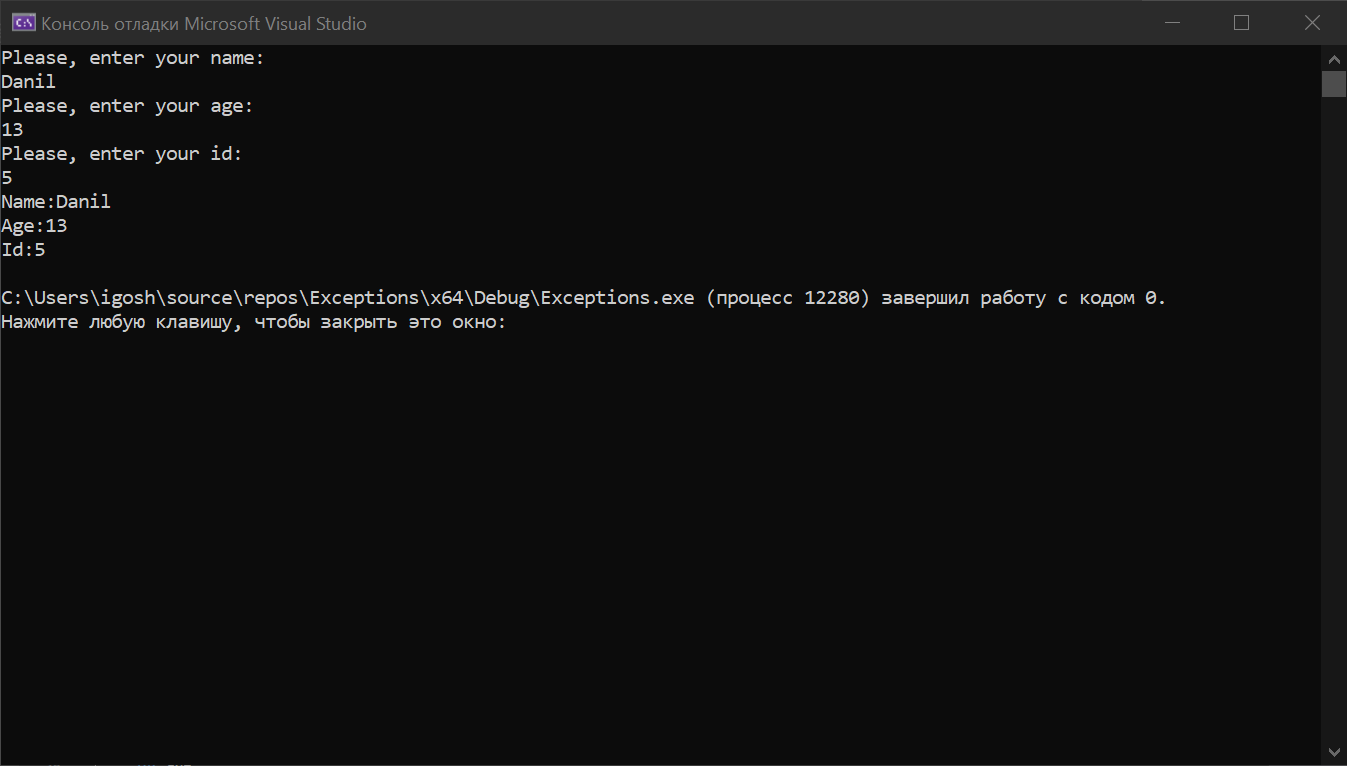


Рисунок 57 – Результат работы

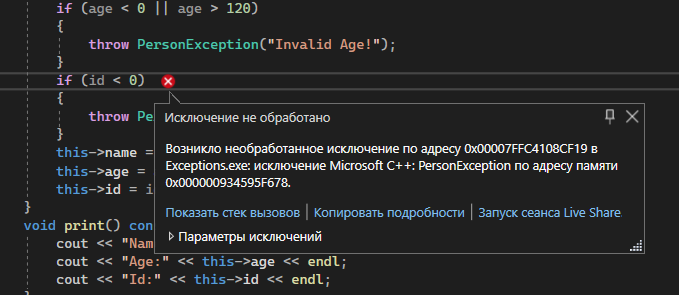


Рисунок 58 – Выдача ошибки возраста

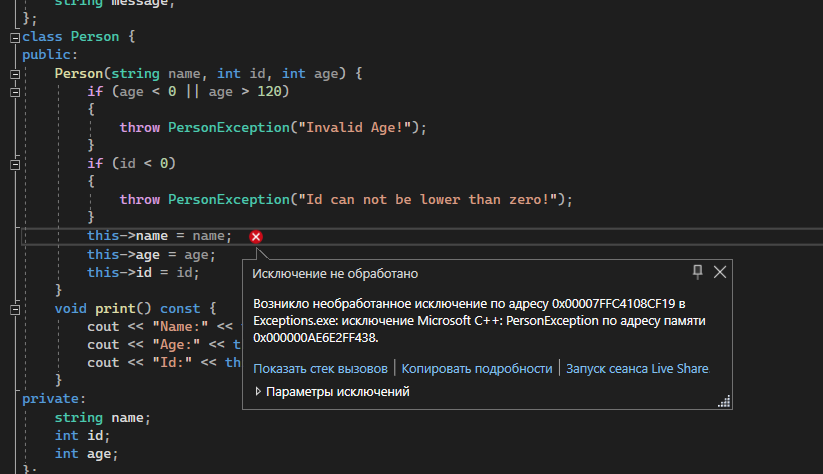


Рисунок 59 – Выдача ошибки id

Вывод: Научились работать с умными указателями в языке C++, с исключениями, их вложениями и типами исключений, выполнив практическое задание для отработки вышеуказанных знаний.

# ПРАКТИЧЕСКАЯ РАБОТА №6

## Тема: «Типы контейнеров. Итераторы. Array. List. Forward\_list. Deque. Адаптеры контейнеров. Stack. Queue. Priority\_queue»

Цель работы: Ознакомиться и научиться работать с итераторами, контейнерами Array, List, Forward\_list, Deque и адаптерами контейнеров Stack, Queue, Priority\_queue.

Для начала создадим структуры. У нас будет структура книги и структура сравнения объектов книги.

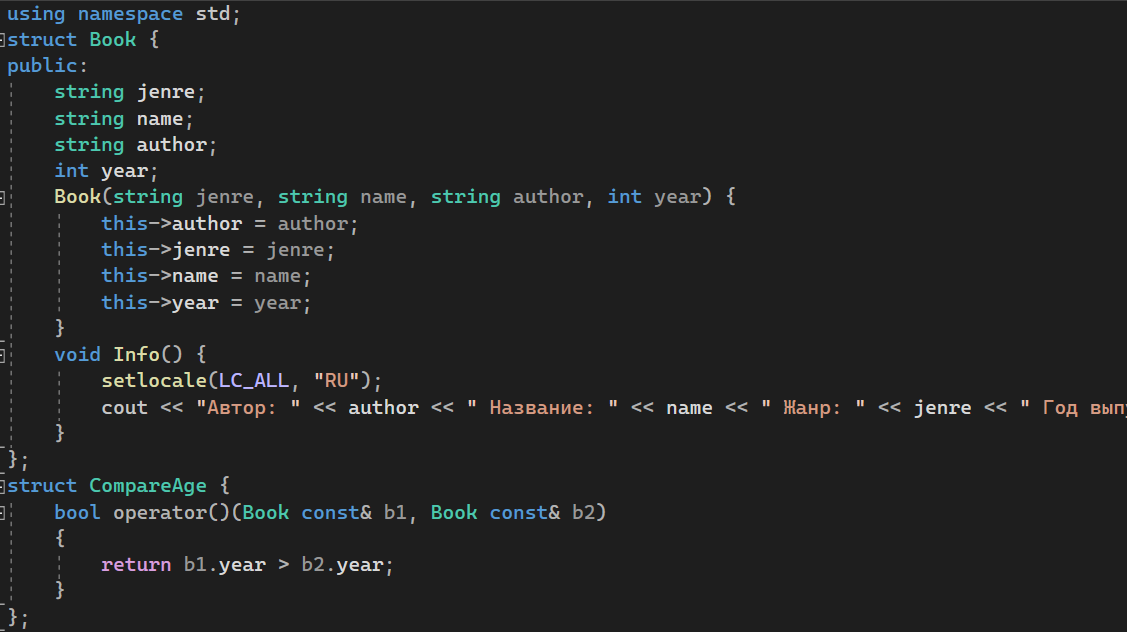


Рисунок 60 – Структура книги и сравнения

Теперь, создадим логику в функции main.

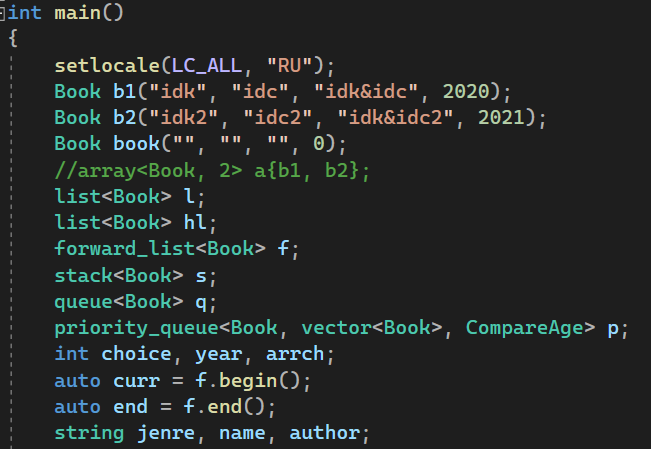


Рисунок 61 – Объявление вспомогательных массивов и переменных

Здесь мы будем добавлять книги:



Рисунок 62 – Добавить книгу

Здесь – выводить книги с последней добавленной.



Рисунок 63 - Вывод информации о книгах с последней добавленной

А теперь создадим и вывод с первой добавленной.

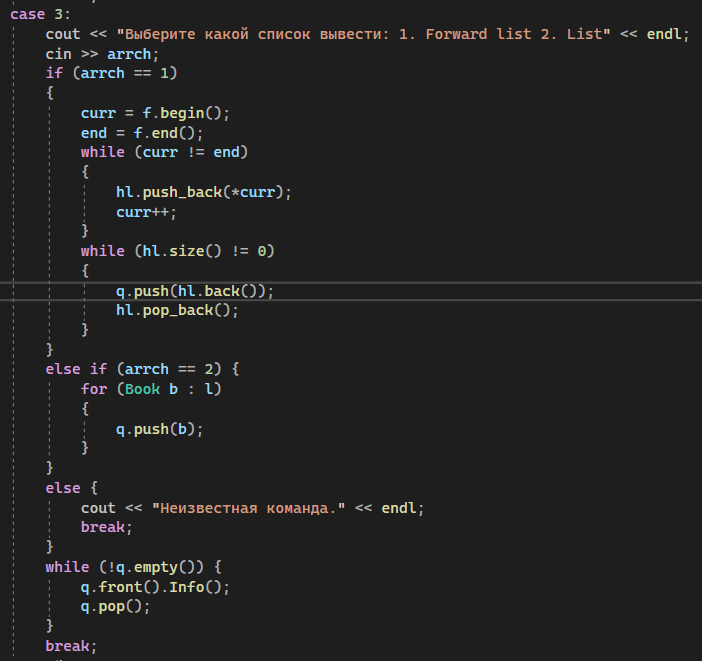


Рисунок 64 - Вывод информации о книгах с первой добавленной

Теперь же, выведем книги, отсортированные по году выпуска:

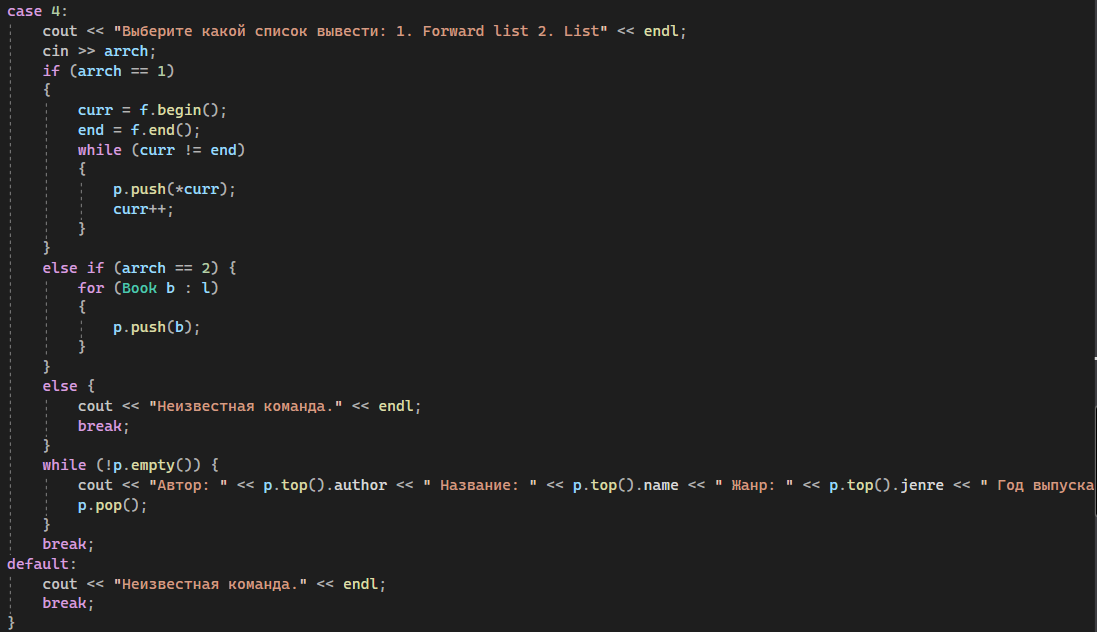


Рисунок 65 - Вывод информации о книгах начиная с самой старой по дате выпуска

Результаты работы:

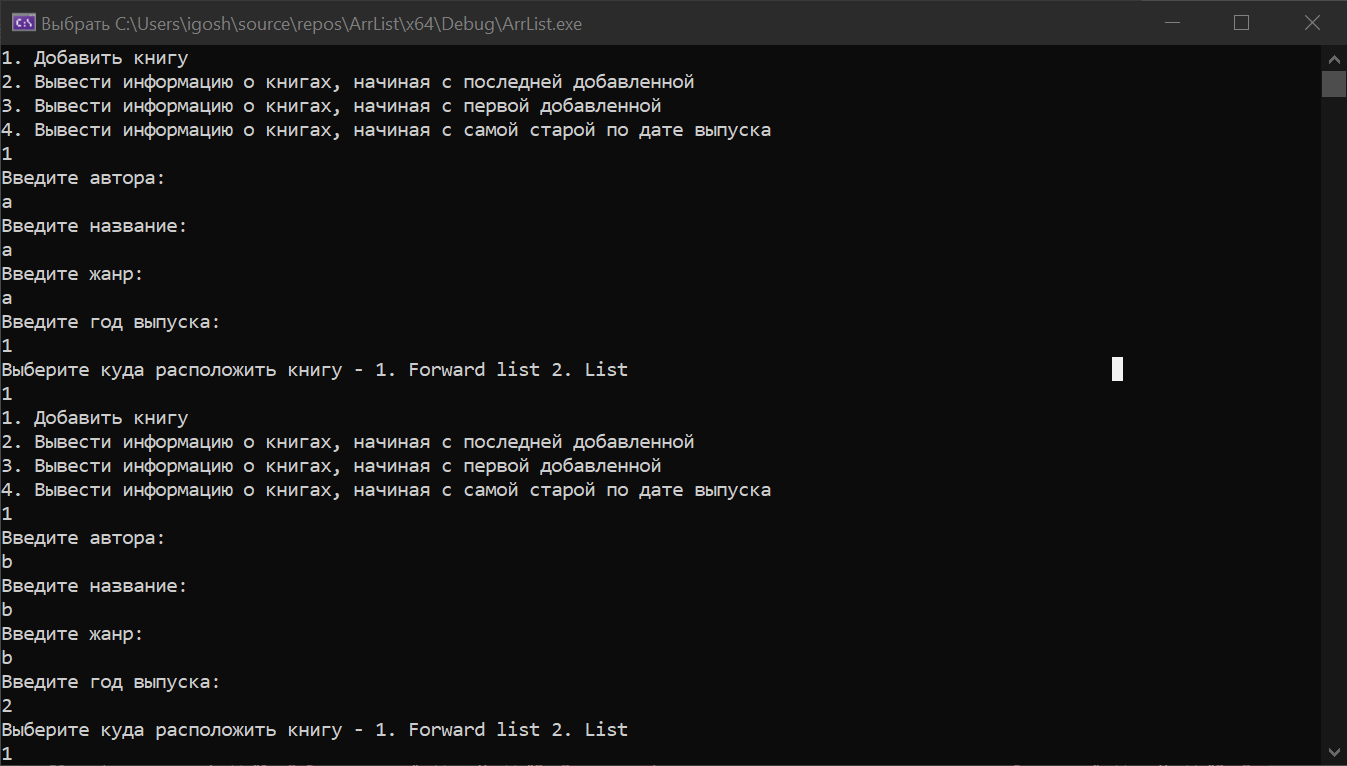


Рисунок 66 – Добавление книг в forward list

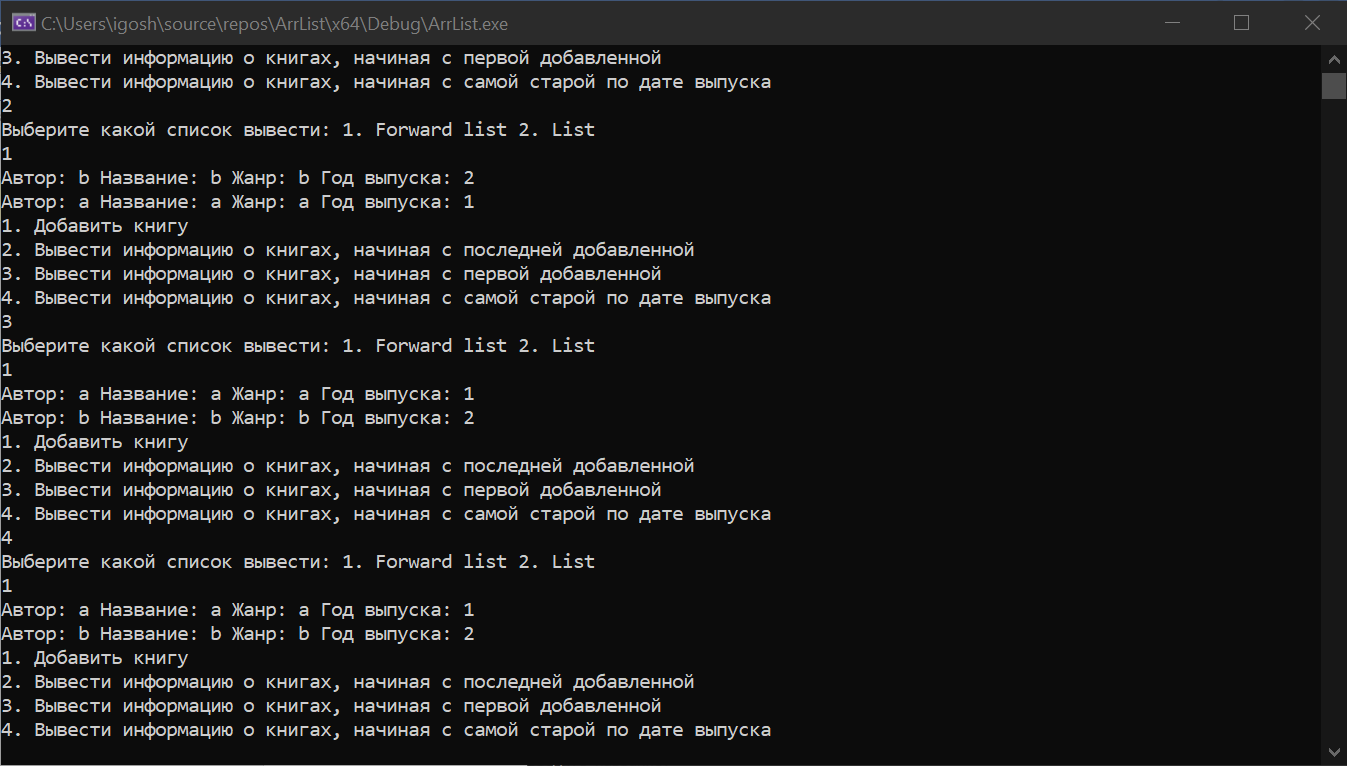


Рисунок 67 – Вывод forward list

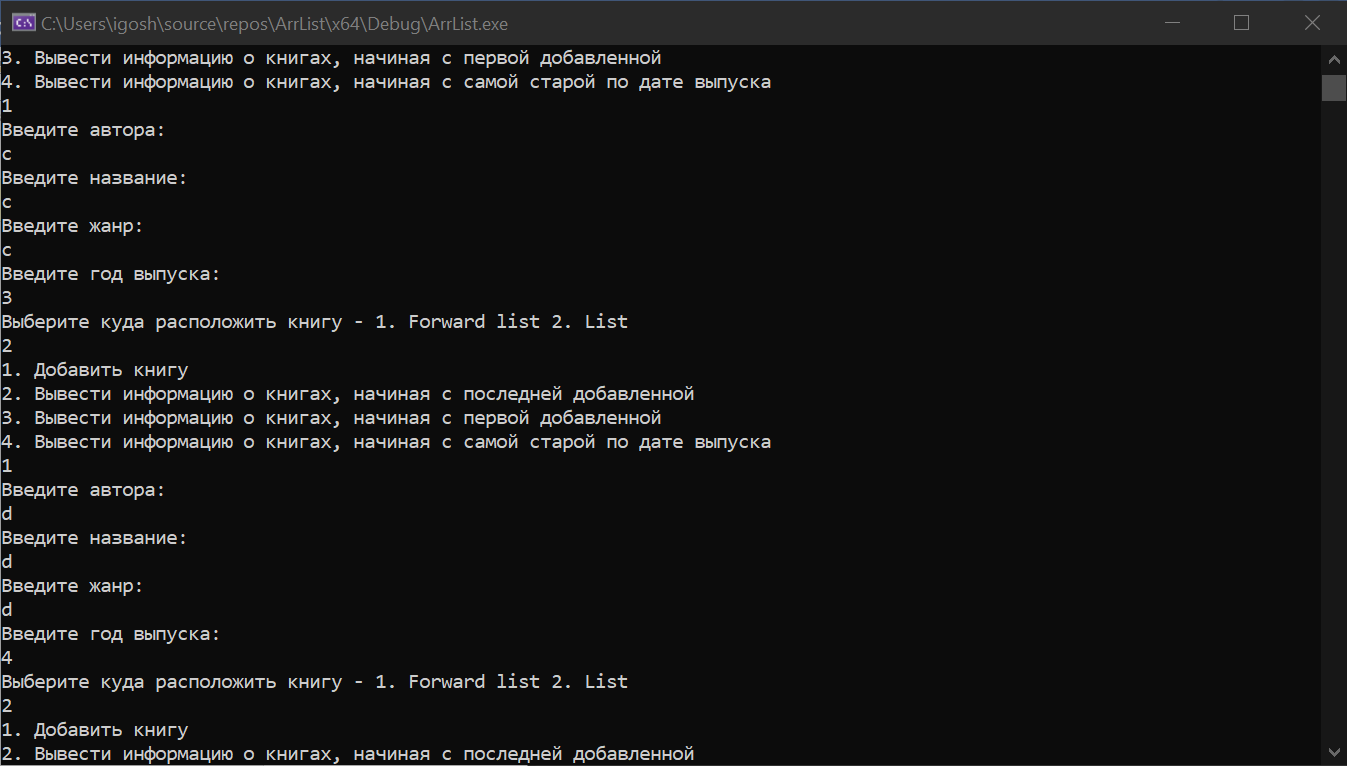


Рисунок 68 – Добавление книг в list

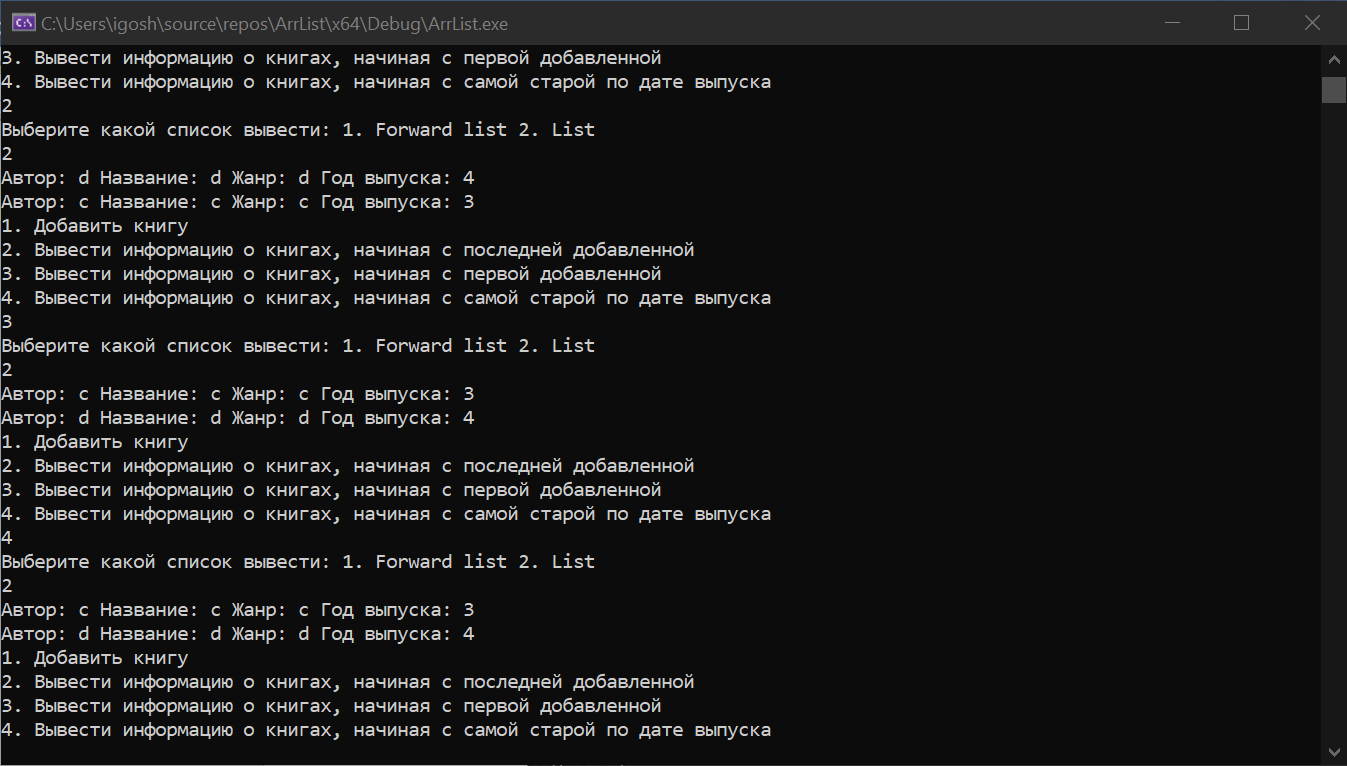


Рисунок 69 – Вывод list

Вывод: Ознакомились и научились работать с итераторами, контейнерами Array, List, Forward\_list, Deque и адаптерами контейнеров Stack, Queue, Priority\_queue.

# ПРАКТИЧЕСКАЯ РАБОТА №7

## Тема: «Ассоциативные контейнеры. Сбалансированные деревья поиска. Map. Multimap. Set. Multiset.»

Цель работы: Ознакомиться и научиться работать с ассоциативными контейнерами Map, Multimap, Set, Multiset и сбалансированные деревья поиска.

Начнём с создания класса работы с файлами.

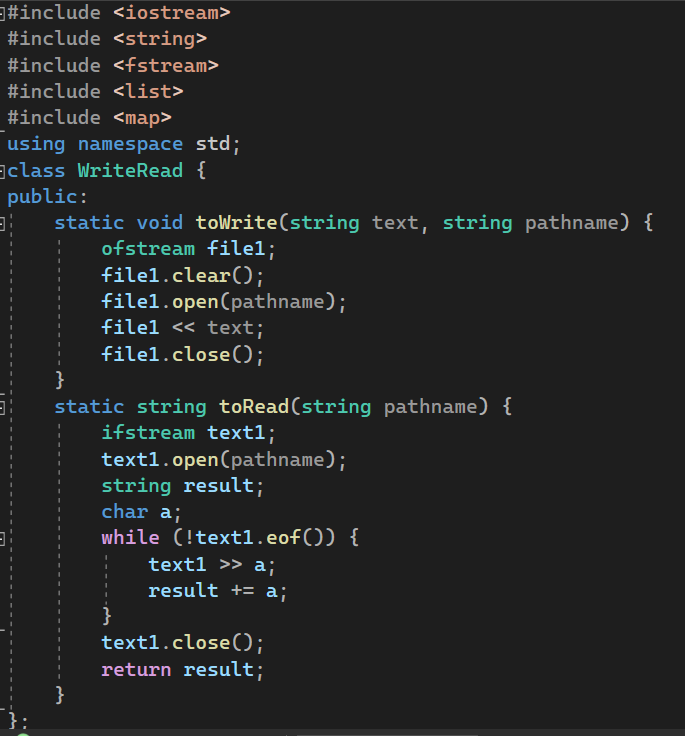


Рисунок 70 – Класс работы с файлами

Реализуем логику функции main.

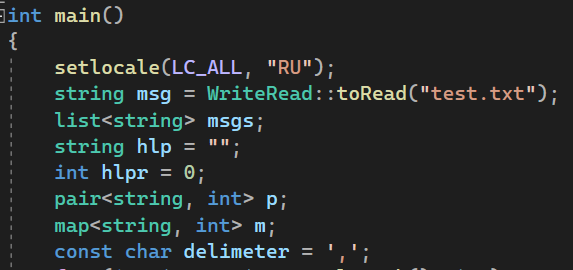


Рисунок 71 – Объявление переменных и массивов

Здесь мы разделяем текст файла.

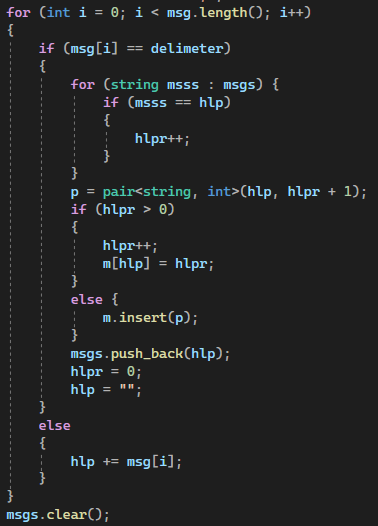


Рисунок 72 – Цикл для разделения полученного текста и помещение их в ассоциативный контейнер map

Здесь – заполняем список.

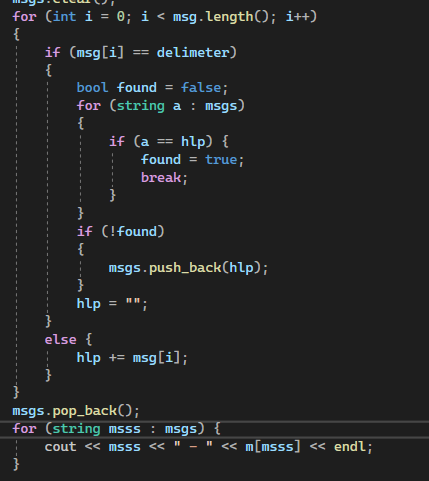


Рисунок 73 – Добавление сообщений в список и вывод их самих и их кол-во

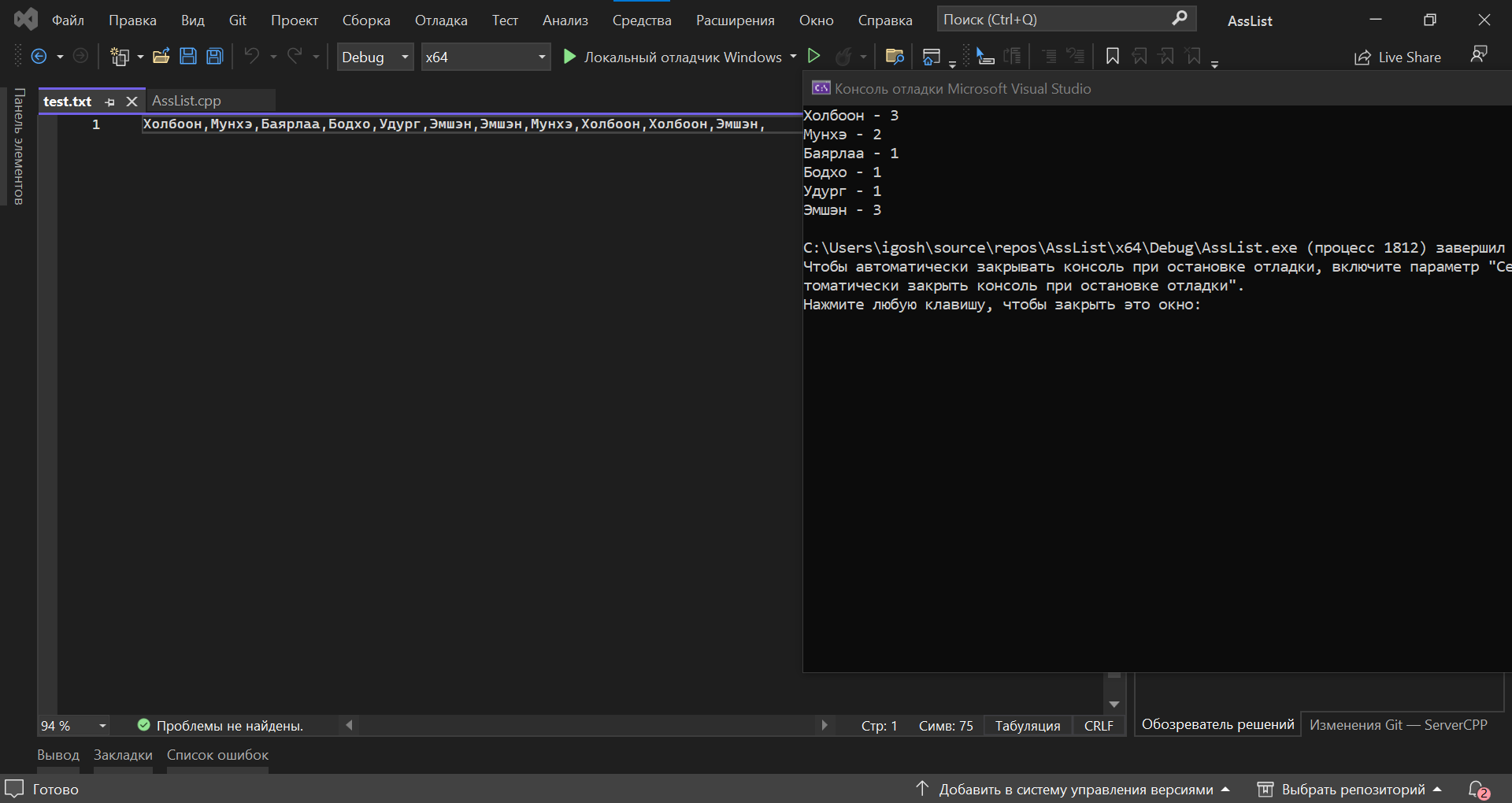


Рисунок 74 – Результат работы

Вывод: Ознакомились и научились работать с ассоциативными контейнерами Map, Multimap, Set, Multiset и сбалансированные деревья поиска.

ПРАКТИЧЕСКАЯ РАБОТА №8

Тема: «Многопоточное программирование. Потоки. Thread. Потоки с параметрами. Возврат результата из потока по ссылке. Лямбда-выражения. Лямбда функции. Анонимные функции. Класс function. Полиморфная обёртка функции. Захват контекста. Возврат результата»

Цель работы: Ознакомиться и научиться работать с многопоточным программированием, потоками, потоками с параметрами, возвратом результата из потока по ссылке, лямбда-выражениями, лямбда функциями, анонимными функциями, классом function, полиморфной обёрткой функции, захватом контекста, возвратом результата выполнения потока и потоками с методами класса.

Для начала реализуем класс с функцией сортировки слиянием.



Рисунок 75 – Класс сортировки слиянием

Теперь реализуем заполнение массива пользователем:

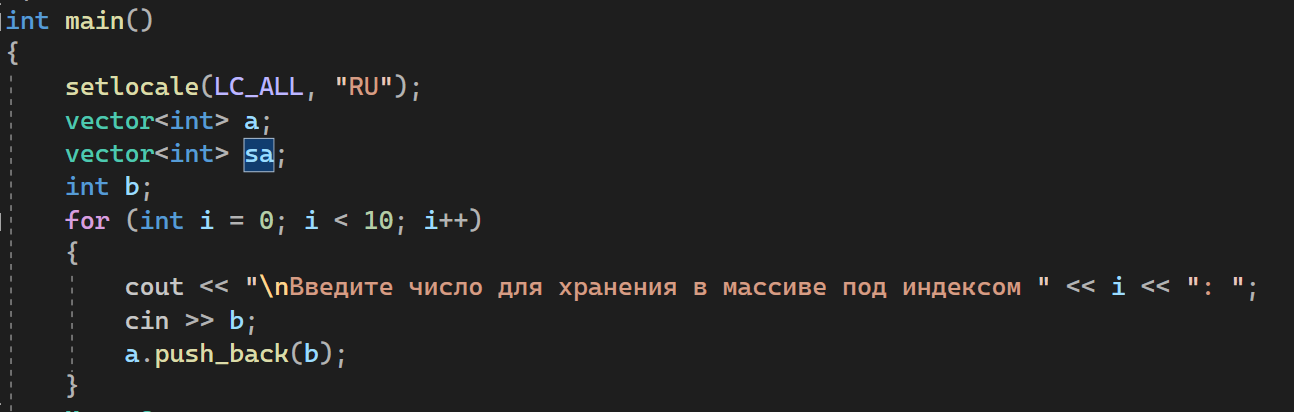


Рисунок 76 – Заполнение массива

Ну и напишем вызов функции через экземпляр класса в потоке используя лямбда-функцию. Также, join() позволяет нам ждать окончания сортировки, после чего выводится массив.

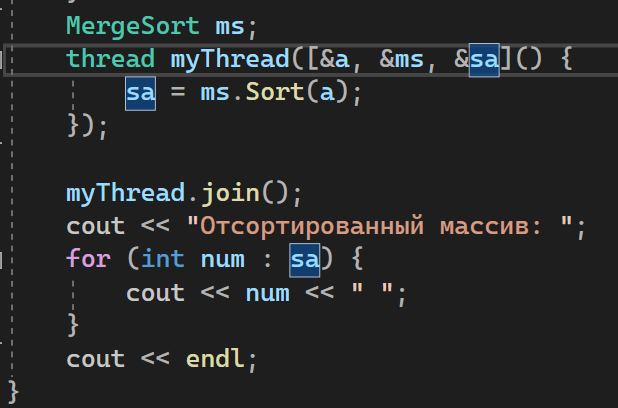


Рисунок 77 – Вызов сортировки слиянием через лямбда-функцию в потоке и ожидание его окончания и вывод массива

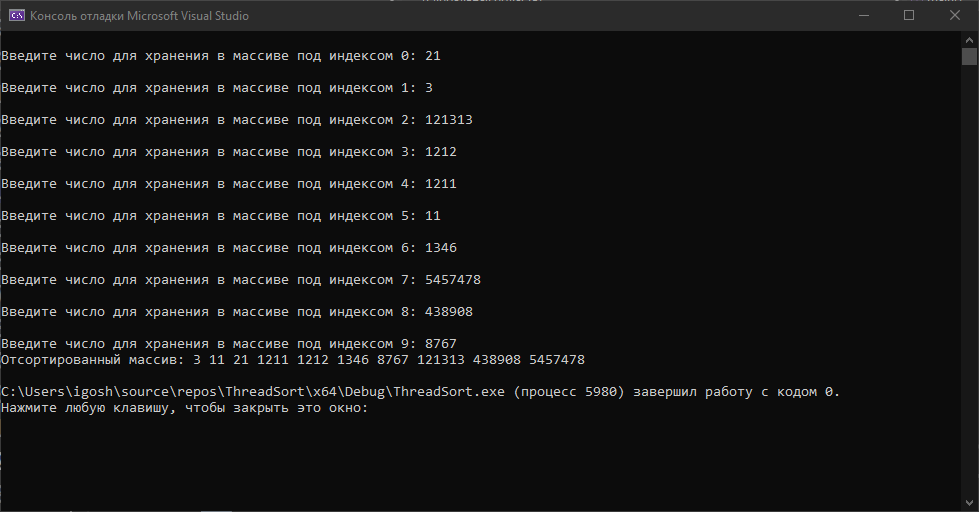


Рисунок 78 – Результат работы

Вывод: Ознакомились и научились работать с многопоточным программированием, потоками, потоками с параметрами, возвратом результата из потока по ссылке, лямбда-выражениями, лямбда функциями, анонимными функциями, классом function, полиморфной обёрткой функции, захватом контекста, возвратом результата выполнения потока и потоками с методами класса.