**Задание 3 (Одномерные массивы)**

Создать проект Task3 с реализацией функций варианта задания. Тестирование функций осуществлять с помощью gtest. Для каждой функции библиотеки необходимо создать один или при необходимости несколько тестов, проверяющих корректность её работы. См. примеры решения задач 3.1, 3.2, 3.3 в файле «Материалы к Заданию 3.pdf».

Массив, одномерный массив, ввод данных из текстового и бинарного файла, вывод данных в текстовый и бинарный файл, операции над массивами, УКАЗАТЕЛИ

**Пункты задания**

1. Реализовать функцию, создающую одномерный целочисленный динамический массив, вводя элементы массива из заранее созданного текстового файла (имя файла передавать в функцию в качестве параметра **const char\***). Размер массива определять по количеству данных в файле.

Функция должна возвращать указатель на созданный массив и его размер.

Pair<int\*,int >

2. Реализовать функцию вывода одномерного целочисленного массива в текстовый файл. Имя файла, массив и его размерность передавать в функцию в качестве параметров.

3. Реализовать функции ввода элементов одномерного целочисленного массива из заранее созданного бинарного файла и вывода массива в бинарный файл (подумайте, как при вводе определить количество элементов типа int в бинарном файле).

4. Реализовать функции обработки одномерных массивов в соответствии с вариантом задания. Массивы и их размерности передавать в функции в качестве параметров. Если из функции необходимо возвратить более одного параметра, использовать возврат массива или передачу параметров по ссылке. Значения границ интервалов (a и b) и другие необходимые данные передавать в функции в качестве параметров.

В одномерном массиве, состоящем из n вещественных элементов, Задание 3 Язык программирования С++

1. вычислить сумму элементов массива, расположенных после последнего отрицательного элемента;

2. упорядочить по возрастанию отдельно элементы, стоящие на чётных местах, и элементы, стоящие на нечётных местах.

5. Для перебора элементов массива продемонстрировать использование разных видов оператора цикла (while и for), а также индексирования и адресной арифметики.

**Правила оформления репозитория**

Для сдачи проекта использовать структуру в файловой системе:

CppTask3/Task3/\* – файлы с реализацией пунктов задания (имена файлов давать в соответствии с вариантами);

CppTask3/Task3.Tests/\* – файлы с тестами пунктов задания (имена файлов давать в соответствии с вариантами);

CppTask3/CMakeLists.txt - файл с правилом сборки (изменить в соответствии с вариантом).

**Введение в указатели:**

Указатели в C++ — это переменные, которые хранят адрес памяти другой переменной. Они позволяют вам работать непосредственно с памятью, обеспечивая гибкость и эффективность в программировании. Рассмотрим основные концепции указателей:

**1. Объявление указателей:**

A black and grey background with white text

Description automatically generated

Это объявляет указатель с именем `**ptr**`, который может хранить адрес памяти целочисленной переменной.

**2. Инициализация указателей:**

A black and grey rectangular object with white text

Description automatically generated

Здесь `&number` представляет собой адрес памяти переменной `number`. Указатель `ptr` теперь указывает на этот адрес памяти.

**3. Доступ к значениям через указатели:**

A black screen with white text

Description automatically generated

Оператор `\*` используется для доступа к значению, хранящемуся по адресу, на который указывает указатель.

**4. Динамическое выделение памяти:**

A black and white screen with white text

Description automatically generated

Динамическое выделение памяти позволяет создавать переменные во время выполнения программы и управляется с использованием указателей.

**5. Удаление динамически выделенной памяти:**

A black and grey rectangular object with white text

Description automatically generated

Очень важно освобождать динамически выделенную память с помощью оператора `delete`, чтобы избежать утечек памяти.

В C++, бинарный файл - это файл, который содержит данные в бинарном формате, то есть данные хранятся в компьютерном представлении, без изменений или преобразований, которые могли бы произойти при выводе в текстовый формат.

При работе с бинарными файлами данные считываются и записываются в их внутреннем двоичном представлении. Это означает, что можно сохранить структуры данных, массивы и другие объекты в бинарных файлах и затем прочитать их обратно в память, сохраняя все байты данных точно так, как они были записаны.

Использование бинарных файлов имеет преимущества в том, что они обеспечивают более эффективное использование памяти и позволяют сохранять и восстанавливать сложные структуры данных, такие как массивы, структуры и классы, без необходимости преобразования данных в текстовый формат.

Для работы с бинарными файлами в C++, используются классы **std::ifstream** для чтения и **std::ofstream** для записи. Бинарный режим работы с файлами указывается с помощью флага **ios::binary**.