Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Высшая школа кибернетики и цифровых технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

ПО КУРСУ «ПРОГРАММИРОВАНИЕ»

«РАЗРАБОТКА МОДУЛЬНЫХ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ, УКАЗАТЕЛЕЙ И ССЫЛОК»

Выполнил: студент 2 курса группы ПО(б) – 31

Магонов Александр Александрович

Проверил: ассистент ВШ КЦТ

Крылов Владимир Андреевич

Хабаровск 2024 г.

Цель работы: изучить теоретический материал по лабораторной работе и реализовать на языке программирования С++ программы для решения заданий.

Задание:

1. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

- сумму отрицательных элементов массива;

- сумму элементов массива, расположенных между первым и последним нулевыми элементами. Если нулей больше 2, то взять первые 2;

- упорядочить элементы массива по возрастанию.

2) Дана целочисленная квадратная матрица.

- определить сумму элементов в тех столбцах, которые не содержат отрицательных значений;

- определить минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

Ход работы:

1. Теоретический материал приведён ниже:

Побочная диагональ матрицы — диагональ, проведённая из правого верхнего угла в левый нижний.

Сортировка пузырьком — простейший алгоритм сортировки, при котором каждый элемент сравнивается со следующим, пока массив не будет отсортирован.

2) Решение задач приведено ниже:

Задание №1

1. Создаём функцию подсчёта суммы отрицательных элементов массива.
2. Создаём функцию для подсчёта суммы элементов между первым и вторым нулевым элементами.
3. Создаём функцию пузырьковой сортировки массива.
4. Вводим размер массива и сам массив.
5. Выводим результат с вызовом функций.

Код программы для задания 1 (листинг 1).

Листинг 1 – код первого задания

*#include <iostream>*

**using** **namespace** std**;**

*// Функция для подсчета суммы отрицательных элементов массива*

**double** sumNegativeElements**(double** arr**[],** **int** size**)** **{**

**double** sum **=** **0;**

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

**if** **(**arr**[**i**]** **<** **0)** **{**

sum **+=** arr**[**i**];**

**}**

**}**

**return** sum**;**

**}**

*// Функция для подсчета суммы элементов между первым и вторым нулевыми элементами*

**double** sumBetweenZeros**(double** arr**[],** **int** size**)** **{**

**int** firstZeroIndex **=** **-1,** secondZeroIndex **=** **-1;**

*// Находим индекс первого нуля*

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

**if** **(**arr**[**i**]** **==** **0)** **{**

firstZeroIndex **=** i**;**

**break;**

**}**

**}**

*// Находим индекс второго нуля*

**for** **(int** i **=** firstZeroIndex **+** **1;** i **<** size**;** i**++)** **{**

**if** **(**arr**[**i**]** **==** **0)** **{**

secondZeroIndex **=** i**;**

**break;**

**}**

**}**

*// Проверяем существование нулей*

**if** **(**firstZeroIndex **==** **-1** **||** secondZeroIndex **==** **-1)** **{**

**return** **0;**

**}**

*// Находим сумму между*

**double** sum **=** **0;**

**for** **(int** i **=** firstZeroIndex **+** **1;** i **<** secondZeroIndex**;** i**++)** **{**

sum **+=** arr**[**i**];**

**}**

**return** sum**;**

**}**

*// Функция сортировки (пузырьком)*

**void** sortArray**(double** arr**[],** **int** size**)** **{**

**for** **(int** i **=** **0;** i **<** size **-** **1;** i**++)** **{**

**for** **(int** j **=** **0;** j **<** size **-** i **-** **1;** j**++)** **{**

**if** **(**arr**[**j**]** **>** arr**[**j **+** **1])** **{**

*// Меняем значения через третью переменную*

**double** temp **=** arr**[**j**];**

arr**[**j**]** **=** arr**[**j **+** **1];**

arr**[**j **+** **1]** **=** temp**;**

**}**

**}**

**}**

**}**

**int** main**()** **{**

**int** n**;**

cout **<<** "Введите размер массива: "**;**

cin **>>** n**;**

**double** arr**[**n**];**

cout **<<** "Введите элементы массива: "**;**

**for** **(int** i **=** **0;** i **<** n**;** i**++)** **{**

cin **>>** arr**[**i**];**

**}**

cout **<<** "Сумма отрицательных элементов массива: " **<<** sumNegativeElements**(**arr**,** n**)** **<<** endl**;**

cout **<<** "Сумма элементов между нулями: " **<<** sumBetweenZeros**(**arr**,** n**)** **<<** endl**;**

sortArray**(**arr**,** n**);**

cout **<<** "Отсортированный массив: "**;**

**for** **(int** i **=** **0;** i **<** n**;** i**++)** **{**

cout **<<** arr**[**i**]** **<<** " "**;**

**}**

**return** **0;**

**}**

Пример выполнения кода задания №1 приведён ниже на рисунке 1.

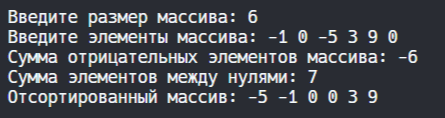


Рисунок 1 – пример выполнения задания №1

Задание №2

1. Создаём функцию для вычисления суммы элементов в столбцах, не содержащий отрицательных значений.
2. Создаём функцию для нахождения минимальной суммы модулей элементов диагоналей, параллельных побочных.
3. Вводим размер матрицы и саму матрицу.
4. Вызываем функции с результатом выводов.

Код программы для задания 1 (листинг 1).

Листинг 2 – код второго задания

*#include <iostream>*

*#include <cmath>*

**using** **namespace** std**;**

*// Функция для вычисления суммы элементов в столбцах, не содержащих отрицательных значений*

**int** noNegativesCol**(int** matrix**[][100],** **int** size**)** **{**

**int** sum **=** **0;**

**for** **(int** col **=** **0;** col **<** size**;** col**++)** **{**

**bool** hasNegative **=** false**;** *// Флаг для проверки наличия отрицательного элемента*

**int** columnSum **=** **0;** *// Сумма текущего столбца*

**for** **(int** row **=** **0;** row **<** size**;** row**++)** **{**

**if** **(**matrix**[**row**][**col**]** **<** **0)** **{** *// Проверка на отрицательное значение*

hasNegative **=** true**;**

**break;** *// Пропускаем столбец, если есть отрицательный элемент*

**}**

columnSum **+=** matrix**[**row**][**col**];**

**}**

**if** **(!**hasNegative**)** **{** *// Если столбец не содержит отрицательных*

sum **+=** columnSum**;** *// Добавляем его сумму к общему результату*

**}**

**}**

**return** sum**;**

**}**

*// Функция для нахождения минимальной суммы модулей элементов диагоналей, параллельных побочной*

**int** diagonals**(int** matrix**[][100],** **int** size**)** **{**

**int** minSum **=** INT\_MAX**;** *// Изначально огромное число*

*// Диагонали выше побочной*

**for** **(int** offset **=** **1;** offset **<** size**;** offset**++)** **{**

**int** sum **=** **0;** *// offset – смещение от побочной диагонали*

**for** **(int** row **=** **0,** col **=** size **-** **1** **-** offset**;** row **<** size **&&** col **>=** **0;** row**++,** col**--)** **{**

**if** **(**col **<** **0)** **break;** *// Проверяем границы*

sum **+=** abs**(**matrix**[**row**][**col**]);**

**}**

**if** **(**sum **<** minSum**)** **{**

minSum **=** sum**;**

**}**

**}**

*// Диагонали ниже побочной*

**for** **(int** offset **=** **1;** offset **<** size**;** offset**++)** **{**

**int** sum **=** **0;**

**for** **(int** row **=** offset**,** col **=** size **-** **1;** row **<** size **&&** col **>=** **0;** row**++,** col**--)** **{**

sum **+=** abs**(**matrix**[**row**][**col**]);**

**}**

**if** **(**sum **<** minSum**)** **{**

minSum **=** sum**;**

**}**

**}**

**return** minSum**;**

**}**

**int** main**()** **{**

**int** size**;**

cout **<<** "Введите размер матрицы: "**;**

cin **>>** size**;**

**int** matrix**[100][100];** *// Объявление квадратной матрицы размером до 100x100*

cout **<<** "Введите элементы матрицы:" **<<** endl**;**

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

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

cin **>>** matrix**[**i**][**j**];** *// Ввод элементов матрицы*

**}**

**}**

*// Вызов функций и вывод результатов*

cout **<<** "Сумма элементов в столбцах без отрицательных значений: "

**<<** noNegativesCol**(**matrix**,** size**)** **<<** endl**;**

cout **<<** "Минимальная сумма модулей элементов диагоналей, параллельных побочной: "

**<<** diagonals**(**matrix**,** size**)** **<<** endl**;**

**return** **0;**

**}**

Пример выполнения кода задания №2 приведён ниже на рисунке 2.

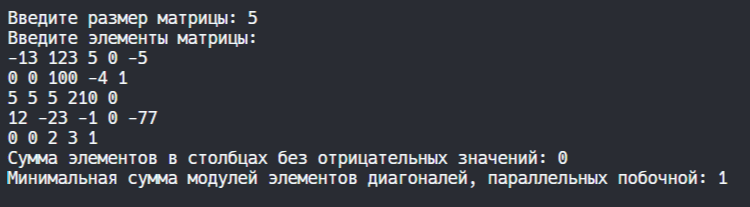


Рисунок 2 – пример выполнения задания №2

Вывод: в ходе работы были освоены навыки разработки и реализации модульных программ, предполагающих работу с функциями, развит навык алгоритмического мышления, применимый к решению подобных и последующих задач. Были решены задачи на работу с матрицами. Побочная диагональ матрицы — диагональ, проведённая из правого верхнего угла в левый нижний. Также, использовался алгоритм сортировки пузырьком. Сортировка пузырьком — простейший алгоритм сортировки, при котором каждый элемент сравнивается со следующим, пока массив не будет отсортирован. В данной лабораторный работе задачи решены на языке программирования C++, однако сами алгоритмы могут быть выполнены и на других языках, суть решения задачи от этого не меняется. Были решены две задачи на функции и действия с ними, изучены различные операции. Благодаря алгоритмическому мышлению и программной реализации алгоритмов, повышается эффективность работы и качество конечного продукта – лабораторной работы, в нашем случае. Данные знания помогут при решении и оформлении дальнейших лабораторных работ.