Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет (НИУ)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

о выполнении практического задания №1

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 5

Проверил:

ст. преподаватель кафедры СП

Петрова Л.Н.

Выполнил:

Студент группы КЭз-391

Галиулин Р.Р.

Челябинск
2025

Содержание

1. Описание задачи

Задание №1: Массив

Дан одномерный целочисленный массив порядка N. Найдите сумму положительных элементов массива, стоящих между первым и последним отрицательными элементами. Если таких элементов нет, вернуть значение 0.

Входные данные:

Размер массива N - целое положительное число больше нуля $N \in \mathbb{N}$

Массив $Z(Z_0...Z_i...Z_{N-1})$- состоящий из целых чисел $Z_i \in \mathbb{Z}$

Все данные выводятся с помощью стандартного потока вывода.

Выходные данные:

R - целое положительное число $R \in \mathbb{N}$ или 0

Данные вводятся с помощью стандартного потока ввода.

Задание №2: Строка

Отфильтровать из строки числа Вводится строка, содержащая буквы, целые неотрицательные числа и иные символы. Требуется все числа, которые встречаются в строке, поместить в отдельный целочисленный массив. Например, если дана строка «data 48 call 9 read13 blank0a», то в массиве должны оказаться числа 48, 9, 13 и 0.

Входные данные:

Строка символов, содержащая буквы, целые неотрицательные числа и иные символы

Все данные выводятся с помощью стандартного потока вывода.

Выходные данные:

Массив целых неотрицательных чисел

Данные вводятся с помощью стандартного потока ввода.

2. Листинги программ

Язык программирования: C++ 14. Среда разработки: Ubuntu 24.10 (6.11.0-13), gcc 14.2.0, neovim

Листинг 1: Задание 1: Массив

```cpp
//
// Галиулин РР.. КЭз-391
// Структуры и алгоритмы обработки данных
// Практическое занятие №1

// Дан одномерный целочисленный массив порядка N. Найдите сумму положительных
// элементов массива, стоящих между первым и последним отрицательными
// элементами.
// Если таких элементов нет, вернуть значение 0.

#include <iostream>
#include <vector>
#include <limits>

int SumBetweenNegatives(const std::vector<int>& arr) {
// Функция возвращает сумму положительных элементов массива, между первым и посденим
    отрицательными элементами

    int first_negative_idx = -1;
    int last_negative_idx = -1;
    int sum_positive = 0;

    // Находим индексы первого и последнего отрицательных элементов
    for (int i = 0; i < arr.size(); ++i) {
        if (arr[i] < 0) {
            if (first_negative_idx == -1) {
                first_negative_idx = i;
            }
            last_negative_idx = i;
        }
    }

    // Если в массиве нет отрицательных элементов или они расположены так, что
    // между ними нет положительных элементов
    if (first_negative_idx == -1 || last_negative_idx == -1 ||
        first_negative_idx >= last_negative_idx) {
        return 0;
    }

    // Суммируем положительные элементы между первым и последним отрицательными
    // элементами
    for (int i = first_negative_idx + 1; i < last_negative_idx; ++i) {
        if (arr[i] > 0) {
            sum_positive += arr[i];
        }
    }

    return sum_positive;
}

int main() {
   int n; //Колво- элементов массива

    std::cout << "Сумма положительных элементов между первым и последним "<<
```

```
    "отрицательными элементами" << std::endl;

    // Проверка ввода размера массива
    do {
        std::cout << "Введите положительное целое число для длины массива: ";
        std::cin >> n;
        if (std::cin.fail() || n <= 0) {
            std::cin.clear(); // Очистка флага ошибки
            // Игнорирование некорректного ввода
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n
    ');
            std::cout << "Некорректный ввод. Попробуйте снова." << std::endl;
        }
    } while (n <= 0);

    std::vector<int> arr(n);

    // Проверка ввода только целых чисел для элементов массива
    std::cout << "Введите элементы массива: " << std::endl;
    for (int i = 0; i < n; ++i) {
        while (true) {
            std::cout << "Элемент " << i + 1 << ": ";
            std::cin >> arr[i];
            if (std::cin.fail()) {
                std::cin.clear(); // Очистка флага ошибки
                // Игнорирование некорректного ввода
                std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
    '\n');
                std::cout << "Некорректный ввод. Введите целое число: ";
            } else {
                break; // Корректный ввод
            }
        }
    }

    int result = SumBetweenNegatives(arr);
    std::cout << "Сумма положительных элементов между первым и последним"
     "отрицательными элементами: " << result << std::endl;

    return 0;
}
```

Листинг 2: Задание 2: Строка

```
// Галиулин РР.. КЭз-391
// Структуры и алгоритмы обработки данных
// Практическое занятие №1

// Отфильтровать из строки числа
// Вводится строка, содержащая буквы, целые неотрицательные числа и
// иные символы.
// Требуется все числа, которые встречаются в строке, поместить в отдельный
// целочисленный массив. Например, если дана строка
// «data 48 call 9 read13 »blank0a,
// то в массиве должны оказаться числа 48, 9, 13 и 0.

#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <cctype>
```

```cpp


std::vector<int> extract_numbers_from_string(const std::string& str) {
/*Функция извлечения чисел из строки*/
    std::vector<int> extract_numbers;
    std::string temp_number;

    for (char c : str) {
        if (std::isdigit(c)) {
            temp_number += c;
        } else {
            if (!temp_number.empty()) {
                extract_numbers.push_back(std::stoi(temp_number));
                temp_number.clear();
            }
        }
    }

    // Добавиление последнего числа, если строка оканчивается числом
    if (!temp_number.empty()) {
        extract_numbers.push_back(std::stoi(temp_number));
    }

    return extract_numbers;
}

int main() {
    std::cout <<"Числа, которые встречаются в строке, поместить в отдельный "
                "целочисленный массив" << std::endl;

    std::string input;
    std::cout << "Введите строку: ";
    std::getline(std::cin, input);

    std::vector<int> numbers = extract_numbers_from_string(input);

    std::cout << "Извлеченные числа: ";
    for (int num : numbers) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

3. Контрольные тесты

3.1 Задание №1: Массив

Исходные данные	Результат
5 -1 3 5 -4 2	8
5 1 2 3 4 5	0
5 -5 -6 3 4 5	0

Таблица 1: Таблица с результатами контрольных тестов Задания №1

3.2 Задание №2: Строка

Исходные данные	Результат
data 48 call 9 read13 blank0a	48 9 13 0
0qwe1rty2asdf6	0 1 2 6
аворп+-+45оаавыgfgdfd+*56	45 56

Таблица 2: Таблица с результатами контрольных тестов Задания №2

4. Контрольные вопросы

4.1 Как найти нужный элемент в массиве?

Ответ на этот вопрос зависит от того имеем ли мы дело с отсортированным или не отсортированным массивом.

В первом случае возможно применить, например:

- Бинарный поиск ($O(\log n)$)- разделение массива пополам на каждом шаге и выбор нужной половины для дальнейшего поиска
- Интерполяционный поиск ($O(\log \log n)$) - вариация бинарного поиска, где вместо деления пополам используется предположение о положении элемента на основе равномерного распределения значений, но в худшем случае также ($O(\log n)$).

Если массив не отсортирован:

- Линейный поиск ($O(n)$)- последовательная проверка каждого элемента массива от начала до конца до нахождения искомого элемента или конца массива.

4.2 Поиск эффективнее происходит в упорядоченном или произвольном массиве?

Более эффективен в упорядоченном массиве - $O(\log n)$ против $O(n)$

4.3 Как поменять местами два элемента массива?

Поменять местами два элемента массива, нужно временно сохранить значение одного из них во временную переменную (того же типа что и элемент массива), затем присвоить другому элементу его значение, а после этого вернуть сохраненное значение первому элементу.

Возможно также применить побитовую операцию XOR для двух элементов массива, например так:

```
    x[a] ^= x[b];
    x[b] ^= x;
    x[a] ^= x[b];

```

в этом случае временная переменная не потребуется, но код становится не наглядным и лучше применять там где это действительно нужно.

4.4 Как создать одномерный динамический массив?

Общая схема заключается в выделении в динамической памяти, некоторой области памяти, увеличения или изменения этой области (если потребуется) и освобождения области.

В различных языка программирования это происходит по разному, явно и неявно.

Пример явной работы с динамической памятью

Пример C

```
int *mas = (int*)malloc(sizeof(int) * n);  // Создание массива из n элементов типа int
...
mas = (int*)realloc(mas, sizeof(int) * m); // Изменение размера массива с n на m с сохранением содержимого
...
free(mas); // Освобождение памяти после использования массива

```

И пример не явной работы:

Пример C++

```

// Объявляем массив mas, изначально содержащий числа 1 - 5
std::vector<int> mas = {1, 2, 3, 4, 5};

mas.reserve(100);      // Зарезервировать место для хранения не менее 100 элементов, не изменяя фактический размер. Содержимое остаётся прежним.

mas.resize(50);        // Задать явный размер - ровно 50 элементов. Недостающие элементы получат значение по" умолчанию", лишние элементы будут удалены.


```

и ещё более не явно:

Пример Python

```

```

```
dynamic_list = []

# Добавление элементов в список
for i in range(1, 11):
dynamic_list.append(i)

# Расширяем список
for i in range(11, 21):
dynamic_list.append(i)
```

4.5 Как выделить память под одномерный динамический массив?

В языке C вызвать функцию malloc(), calloc() или realloc():

```
    int *mas = (int*)malloc(sizeof(int) * n);
```

В языке C++ вызвать функцию malloc() и т.д., new() или вызвать конструктор типа vector:

```
    int *mas1= (int*)malloc(sizeof(int) * n);
    int *mas2= new int [n];
    std::vector<int> mas3= {1, 2, 3, 4, 5};
```

Более высокоуровневые конструкции скрывают фактическую реализацию и можно сказать что так или иначе вызывается malloc() (более точно ёще более низкоуровневая функция)

4.6 Как определить размер динамического массива в текущий момент времени?

Сделать это строго говоря не возможно, так указатель на динамический массив указывает только на первый элемент. В связи с чем необходимо хранить размер массива в отдельной переменной.

В языке C это необходимо делать явно. В таких языках как C++, Python типы которые являются динамическими массивами: vector, list, позволяют вызвать метод который возвращает размер массива.

4.7 Какими способами можно освободить всю память, занимаемую динамическими массивами?

В C это делается вызовом функции **free()** для освобождения памяти выделенной **malloc**, **calloc** или **realoc**.

В C++ для этого служит оператор **delete[]**, который в отличии от **delete** освобождает память выделенную под массив. Лучше использовать более безопасные типы вроде **vector**.

В общем смысле все функции и операторы вызывают, функции ядра операционной системы которое уже занимается реализации хранения данных в динамической памяти используя различные структуры данных для хранения информации о выделении и освобождении областей памяти.