АБОРАТОРНАЯ РАБОТА №4

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ СТАТИЧЕСКИХ МАССИВОВ

1. Цель работы:

Исследование особенностей представления и обработки одномерных массивов в языках C/C++ с учетом связи указателей и массивов. Приобретение практических навыков реализации алгоритмов обработки одномерных динамических массивов средствами языков C/C++. Исследование особенностей обработки одномерных динамических массивов.

2. Постановка задачи и вариант задания:

Вариант 5: Последовательность состоит из целых чисел.

Необходимо найти количество элементов последовательности, отвечающих условию: $a_{i-1} < a_i > a_{i+1}$, при i = 2, 3 … n-1.

Упорядочить элементы последовательности a_p, a_{p+1}, …, a_q по убыванию, используя алгоритм сортировки методом прямого выбора.

3. Краткие теоретические сведения

Для выполнения лабораторной работы необходимо изучить особенности объявления, описания и вызова функций, а также понять работу динамических массивов в языках C/C++.

ХОД РАБОТЫ

4. Структурная схема алгоритма

Структурная схема алгоритма, а также всех функций представлены на рисунках 4.1 – 4.5:

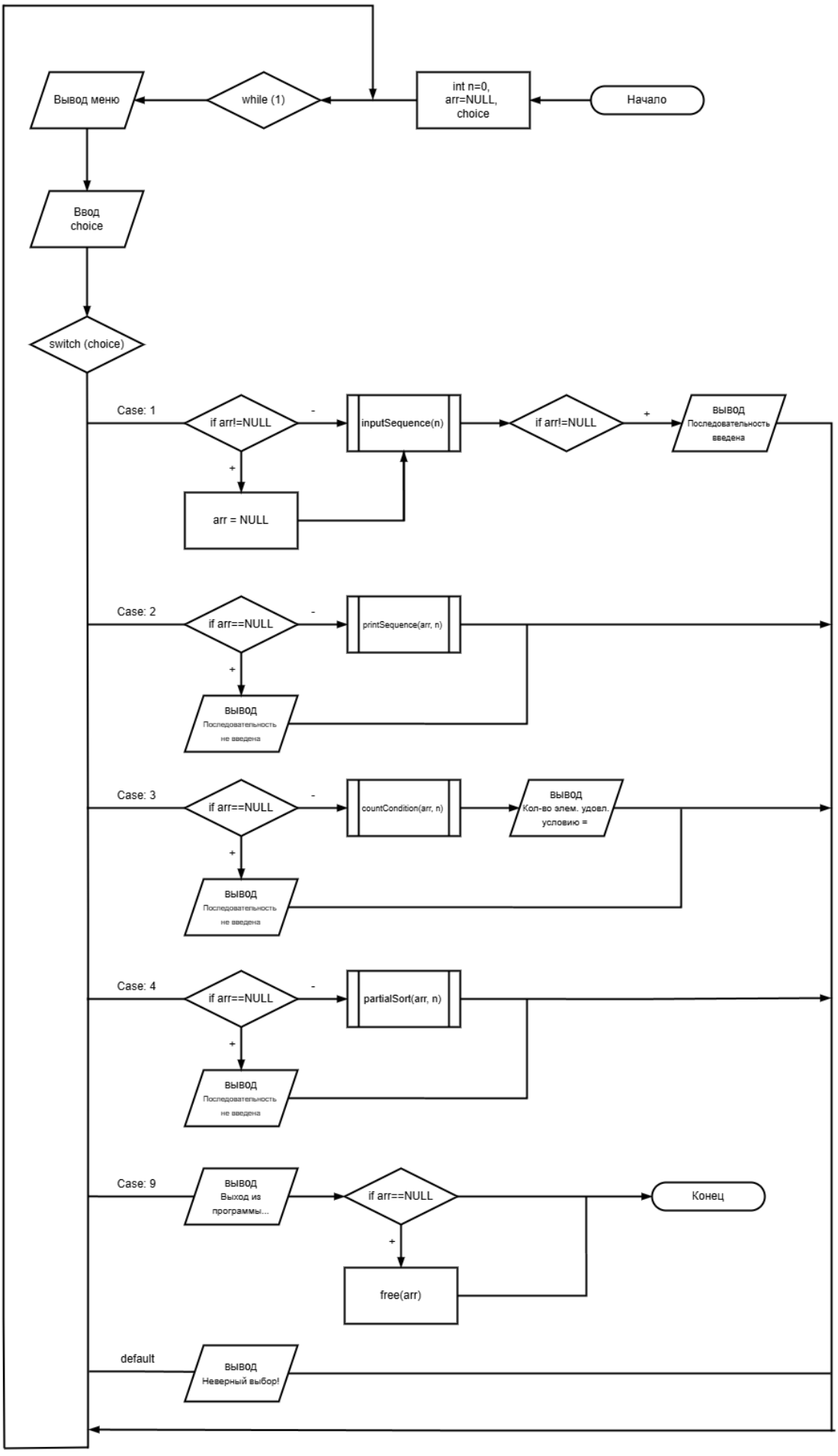

Рисунок 4.1 – Структурная схема алгоритма

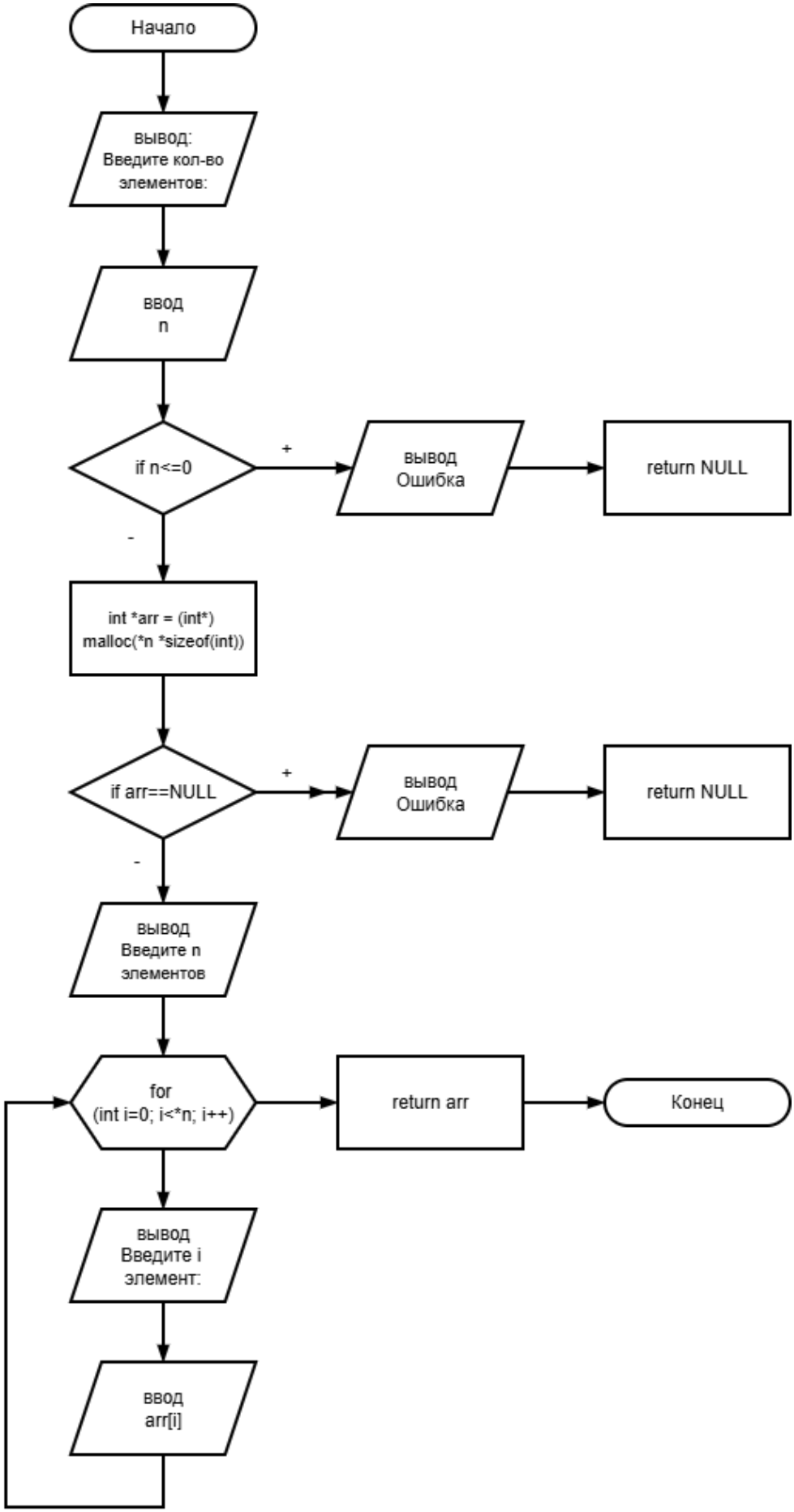

Рисунок 4.2 – Структурная схема функции inputSequence(n)

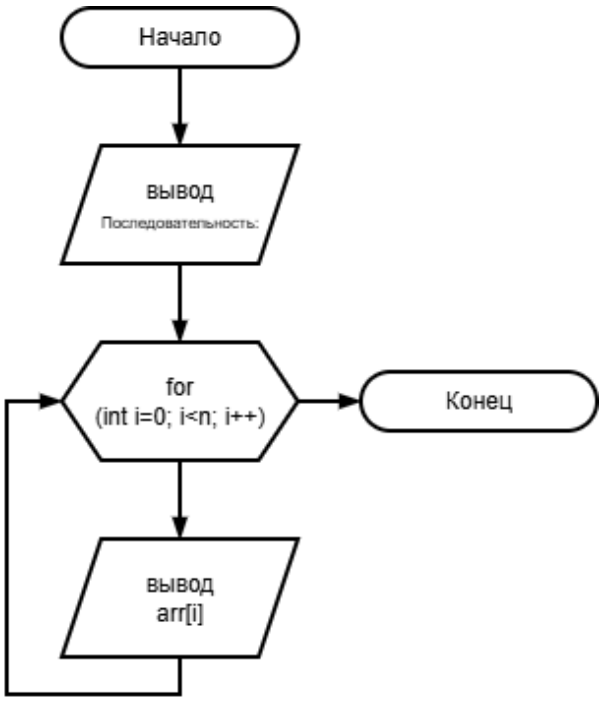

Рисунок 4.3 – Структурная схема функции printSequence(*arr, n)

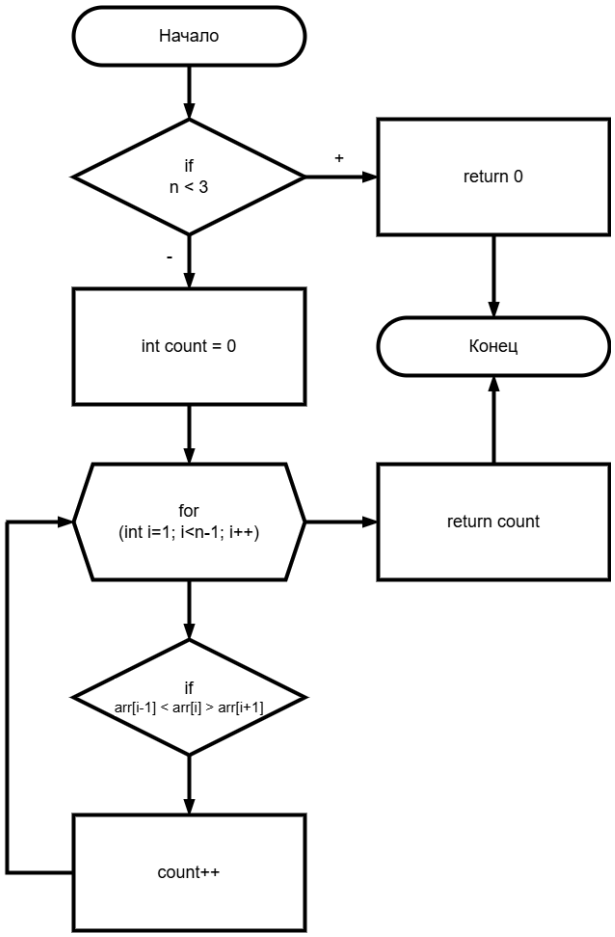

Рисунок 4.4 – Структурная схема функции countCondition(*arr, n)

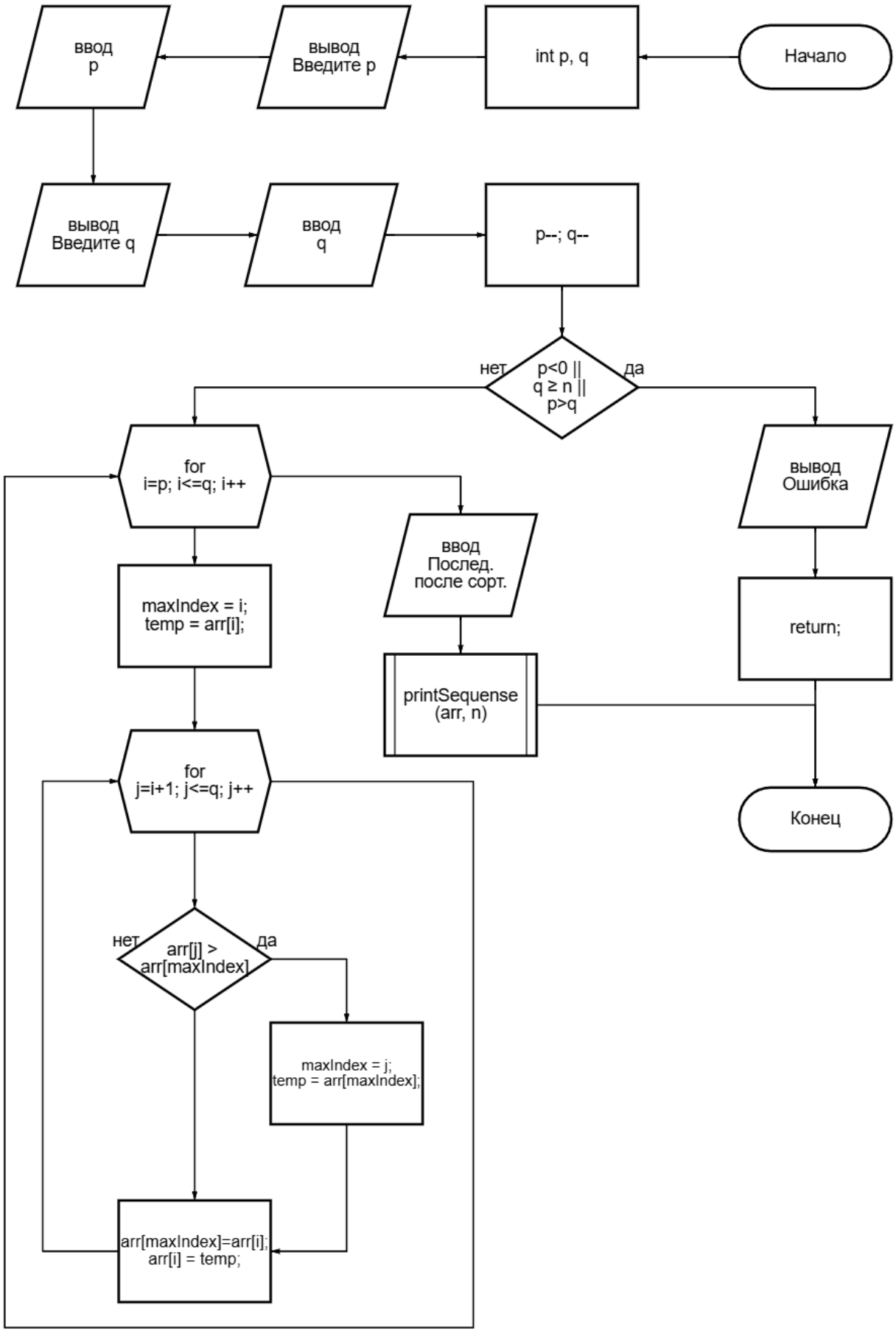

Рисунок 4.5 – Структурная схема функции partialSort(*arr, n)

5. Текст кода на языке С. Тестирование и отладка программы

```
#include <stdio.h>
#include <math.h>

// Объявление функций
int countCondition(int *, int );
int *inputSequence(int *);
void printSequence(int *, int);
void partialSort(int *, int);

int main() {
    int *arr = NULL, n = 0, choice;

    while (1) {
        printf("\n======= М Е Н Ю =======\n");
        printf("1. Ввести новую последовательность\n");
        printf("2. Вывести текущую последовательность\n");
        printf("3. Найти количество элементов по условию\n");
        printf("4. Отсортировать часть последовательности\n");
        printf("9. Выход\n");
        printf("-> ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                // Освобождаем старую память, если она была
выделена
                if (arr != NULL) {
                    free(arr);
                }

                arr = inputSequence(&n);
                if (arr != NULL) {
                    printf("Последовательность успешно
введена.\n");
                }
                break;

            case 2:
                if (arr == NULL) {
                    printf("Последовательность не введена!\n");
                }
                else {
                    printSequence(arr, n);
                }
                break;

            case 3:
                if (arr == NULL) {
                    printf("Последовательность не введена!\n");
                }
                else {
                    int count = countCondition(arr, n);
```

```
                        printf("Количество                элементов,
удовлетворяющих условию ai-1 < ai > ai+1: %d\n", count);
                    }
                    break;

                case 4:
                    if (arr == NULL) {
                        printf("Последовательность не введена!\n");
                    }
                    else {
                        partialSort(arr, n);
                    }
                    break;

                case 9:
                    printf("Выход из программы...\n");
                    // Освобождаем память перед выходом
                    if (arr != NULL) {
                        free(arr);
                    }
                    return 0;
                    break;

                default:
                    printf("Неверный выбор! Попробуйте снова.\n");
            }
        }
    }

    // Функция для подсчета элементов, удовлетворяющих условию ai-1
< ai > ai+1
    int countCondition(int *arr, int n) {
        if (n < 3) return 0;

        int count = 0;
        for (int i = 1; i < n - 1; i++) {
            if (arr[i-1] < arr[i] && arr[i] > arr[i+1]) {
                count++;
            }
        }
        return count;
    }

    // Функция для ввода последовательности
    int *inputSequence(int *n) {
        printf("Введите количество элементов последовательности: ");
        scanf("%d", n);

        if (*n <= 0) {
            printf("Количество      элементов      должно      быть
положительным!\n");
            return NULL;
        }
```

```
        // Выделяем память с использованием malloc
        int *arr = (int*) malloc(*n * sizeof(int));

        if (arr == NULL) {
            printf("Ошибка выделения памяти!\n");
            return NULL;
        }

        printf("Введите %d элементов последовательности:\n", *n);
        for (int i = 0; i < *n; i++) {
            printf("Элемент %d: ", i + 1);
            scanf("%d", &arr[i]);
        }

        return arr;
    }

    // Функция для вывода последовательности
    void printSequence(int *arr, int n) {
        printf("Последовательность: ");
        for (int i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");
    }

    // Функция для обработки частичной сортировки
    void partialSort(int *arr, int n) {
        int p, q;

        printf("Введите индекс p (от 1 до %d): ", n);
        scanf("%d", &p);

        printf("Введите индекс q (от %d до %d): ", p+1, n);
        scanf("%d", &q);

        q--;  p--;  //  Корректировка  значений  (т.к.  нумерация
начинается с 0)

        // Проверка корректности индексов
        if (p < 0 || q >= n || p > q) {
            printf("Некорректные индексы!\n");
            return;
        }

        // Сортируем часть массива
        for (int i = p; i <= q; i++) {
            int maxIndex = i, temp = arr[i];
            // Находим максимальный элемент в оставшейся части
            for (int j = i + 1; j <= q; j++) {
                if (arr[j] > arr[maxIndex]) {
                    maxIndex = j;
```

```
                    temp = arr[maxIndex];
                }
                arr[maxIndex] = arr[i];
                arr[i] = temp;
            }
        }

        printf("Последовательность после сортировки элементов с %d по
%d:\n", p + 1, q + 1);
        printSequence(arr, n);
    }
```

На рисунке 4.6 отображены результаты работы кода согласно всем условиям по заданию для языка C.

```
======= M E H Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 1
Введите количество элементов последовательности: 6
Введите 6 элементов последовательности:
Элемент 1: 32
Элемент 2: 34
Элемент 3: 24
Элемент 4: 26
Элемент 5: 25
Элемент 6: 12
Последовательность успешно введена.

======= M E H Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 2
Последовательность: 32 34 24 26 25 12

======= M E H Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 3
Количество элементов, удовлетворяющих условию ai-1 < ai > ai+1: 2

======= M E H Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 4
Введите индекс p (от 1 до 6): 2
Введите индекс q (от 3 до 6): 4
Последовательность после сортировки элементов с 2 по 4:
Последовательность: 32 34 26 24 25 12

======= M E H Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 9
Выход из программы...
```

Рисунок 4.6 – Результат работы программы на языке С

6. Текст кода на языке C++. Тестирование и отладка программы

```
#include <iostream>
#include <math.h>

using namespace std;

// Объявление функций
int countCondition(int *arr, int n);
int *inputSequence(int *n);
void printSequence(int *arr, int n);
void partialSort(int *arr, int n);

int main()
{
    int *sequence = NULL;
    int n = 0;
    int choice;

    while (1)
    {
        cout << endl << "======= М Е Н Ю =======" << endl;
        cout << "1. Ввести новую последовательность" << endl;
        cout << "2. Вывести текущую последовательность" << endl;
        cout << "3. Найти количество элементов по условию" <<
endl;
        cout << "4. Отсортировать часть последовательности" <<
endl;
        cout << "9. Выход" << endl;
        cout << "-> ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            // Освобождаем старую память, если она была выделена
            if (sequence != NULL)
            {
                free(sequence);
            }

            sequence = inputSequence(&n);
            if (sequence != NULL)
            {
                cout << "Последовательность успешно введена." <<
endl;
            }
            break;

        case 2:
            if (sequence == NULL)
            {
                cout << "Последовательность не введена!" << endl;
            }
```

```
                else
                {
                    printSequence(sequence, n);
                }
                break;

            case 3:
                if (sequence == NULL)
                {
                    cout << "Последовательность не введена!" << endl;
                }
                else
                {
                    int count = countCondition(sequence, n);
                    cout << "Количество элементов, удовлетворяющих
условию ai-1 < ai > ai+1: " << count << endl;
                }
                break;

            case 4:
                if (sequence == NULL)
                {
                    cout << "Последовательность не введена!" << endl;
                }
                else
                {
                    partialSort(sequence, n);
                }
                break;

            case 9:
                cout << "Выход из программы..." << endl;
                // Освобождаем память перед выходом
                if (sequence != NULL)
                {
                    free(sequence);
                }
                return 0;
                break;

            default:
                cout << "Неверный выбор! Попробуйте снова." << endl;
            }
        }
    }

    // Функция для подсчета элементов, удовлетворяющих условию ai-1
< ai > ai+1
    int countCondition(int *arr, int n)
    {
        if (n < 3)
            return 0;
```

```
        int count = 0;
        for (int i = 1; i < n - 1; i++)
        {
            if (arr[i - 1] < arr[i] && arr[i] > arr[i + 1])
            {
                count++;
            }
        }
        return count;
    }

    // Функция для ввода последовательности
    int *inputSequence(int *n)
    {
        cout << "Введите количество элементов последовательности: ";
        scanf("%d", n);

        if (*n <= 0)
        {
            cout << "Количество элементов должно быть положительным!"
<< endl;
            return NULL;
        }

        // Выделяем память с использованием malloc
        int *arr = (int *)malloc(*n * sizeof(int));

        if (arr == NULL)
        {
            cout << "Ошибка выделения памяти!" << endl;
            return NULL;
        }

        cout << "Введите " << *n << " элементов последовательности:
" << endl;
        for (int i = 0; i < *n; i++)
        {
            cout << "Элемент " << i + 1 << ": ";
            cin >> arr[i];
        }

        return arr;
    }

    // Функция для вывода последовательности
    void printSequence(int *arr, int n)
    {
        cout << "Последовательность: ";
        for (int i = 0; i < n; i++)
        {
            cout << arr[i] << " ";
        }
        cout << "" << endl;
```

```
    }

    // Функция для обработки частичной сортировки
    void partialSort(int *arr, int n)
    {
        int p, q;

        cout << "Введите индекс p (от 1 до " << n << ": ";
        cin >> p;

        cout << "Введите индекс q (от " << p + 1 << " до " << n <<
"): ";
        cin >> q;

        q--;
        p--; // Корректировка значений (т.к. нумерация начинается с
0)

        // Проверка корректности индексов
        if (p < 0 || q >= n || p > q)
        {
            cout << "Некорректные индексы!" << endl;
            return;
        }

        // Сортируем часть массива
        for (int i = p; i <= q; i++)
        {
            int maxIndex = i, temp = arr[i];
            // Находим максимальный элемент в оставшейся части
            for (int j = i + 1; j <= q; j++)
            {
                if (arr[j] > arr[maxIndex])
                {
                    maxIndex = j;
                    temp = arr[maxIndex];
                }
                arr[maxIndex] = arr[i];
                arr[i] = temp;
            }
        }

        cout << "Последовательность после сортировки элементов с " <<
p + 1 << " по " << q + 1 << ":" << endl;
        printSequence(arr, n);
    }
```

На рисунке 4.7 отображены результаты работы кода согласно трем условиям по заданию для языка C++.

```
======= М Е Н Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 1
Введите количество элементов последовательности: 8
Введите 8 элементов последовательности:
Элемент 1: 24
Элемент 2: 12
Элемент 3: 2007
Элемент 4: 45
Элемент 5: 128
Элемент 6: 64
Элемент 7: 77
Элемент 8: 1
Последовательность успешно введена.

======= М Е Н Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 2
Последовательность: 24 12 2007 45 128 64 77 1

======= М Е Н Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 3
Количество элементов, удовлетворяющих условию ai-1 < ai > ai+1: 3

======= М Е Н Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 4
Введите индекс p (от 1 до 8): 2
Введите индекс q (от 3 до 8): 5
Последовательность после сортировки элементов с 2 по 5:
Последовательность: 24 45 128 45 12 64 77 1

======= М Е Н Ю =======
1. Ввести новую последовательность
2. Вывести текущую последовательность
3. Найти количество элементов по условию
4. Отсортировать часть последовательности
9. Выход
-> 9
Выход из программы...
```

Рисунок 4.7 – Результат работы программы на языке C++

ВЫВОД

В ходе лабораторной работы были исследованы особенности представления и обработки одномерных динамических массивов в языках C/C++. На практике были успешно реализованы алгоритмы работы с массивами, включая их динамическое создание, обработку и освобождение памяти. Экспериментально подтверждена и изучена неразрывная связь между указателями и массивами, лежащая в основе управления памятью. Приобретены навыки эффективного использования динамического выделения памяти.