



МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №1.1

Тема:

«Оценка вычислительной сложности алгоритма»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент:

Подоплелов А.С.

Группа:

ИНБО-21-23

Москва - 2024

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ.....	3
РЕШЕНИЕ ЗАДАНИЯ 1.....	4
1.1 ФОРМУЛИРОВКА ЗАДАЧИ.....	4
1.2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ ЗАДАЧИ.....	4
1.3 РЕАЛИЗАЦИЯ АЛГОРИТМА В ВИДЕ ФУНКЦИИ И ОТЛАДКА НА МАССИВЕ ПРИ $n = 10$, $n = 100$. ВКЛЮЧЕНИЕ В ФУНКЦИЮ ПОДСЧЕТА СУММАРНОГО КОЛИЧЕСТВА ВЫПОЛНЕННЫХ СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ ЭЛЕМЕНТОВ ПРИ РЕШЕНИИ ЗАДАЧИ УДАЛЕНИЯ КЛЮЧЕВЫХ ЭЛЕМЕНТОВ.....	8
1.4 РЕАЛИЗАЦИЯ ФУНКЦИИ: ЗАПОЛНЕНИЕ МАССИВА ДАТЧИКОМ СЛУЧАЙНЫХ ЧИСЕЛ, ВЫВОД МАССИВА НА ЭКРАН МОНИТОРА.....	10
1.5 ТЕСТИРОВАНИЕ АЛГОРИТМА В СИТУАЦИЯХ: А) СЛУЧАЙНОЕ ЗАПОЛНЕНИЕ МАССИВА; Б) ВСЕ ЭЛЕМЕНТЫ МАССИВА ДОЛЖНЫ БЫТЬ УДАЛЕНЫ; В) НИ ОДИН ЭЛЕМЕНТ НЕ УДАЛЯЕТСЯ. СРАВНЕНИЕ РЕЗУЛЬТАТОВ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ЭТИХ СЛУЧАЕВ.....	11
1.6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ.....	12
1.7 ЁМКостная сложность.....	12
1. 8 ВЫВОД ПО ПЕРВОМУ ЗАДАНИЮ.....	12
РЕШЕНИЕ ЗАДАНИЯ 2.....	14
2.1 ФОРМУЛИРОВКА ЗАДАЧИ.....	14
2.2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ ЗАДАЧИ.....	14
2.3 РЕАЛИЗАЦИЯ ФУНКЦИИ.....	16
2.4 ТЕСТИРОВАНИЕ АЛГОРИТМА.....	16
2.5 ЁМКостная сложность.....	18
2.6 ВЫВОД ПО ВТОРОМУ ЗАДАНИЮ.....	18
ОБЩИЙ ВЫВОД.....	19
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ.....	20

ЦЕЛЬ РАБОТЫ

Приобретение практических навыков по определению:

- эмпирическому определению вычислительной сложности алгоритмов на теоретическом;
- выбору эффективного алгоритма решения вычислительной задачи из нескольких.

РЕШЕНИЕ ЗАДАНИЯ 1

Выбрать эффективный алгоритм вычислительной задачи из двух предложенных, используя теоретическую и практическую оценку вычислительной сложности каждого из алгоритмов, а также емкостную сложность.

1.1 ФОРМУЛИРОВКА ЗАДАЧИ

Необходимо реализовать два предложенных алгоритма, которые удаляют из массива значение, заданное пользователем. Оценить их вычислительную сложность теоретически и практически, а также сделать вывод об их эффективности.

1.2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ ЗАДАЧИ

а) Описание работы двух алгоритмов с помощью блок-схем.

Описание работы первого алгоритма (См. Рисунок 1).

Идем с помощью цикла по элементам массива. В случае нахождения элемента, который равен «key», сдвигаем все последующие элементы справа налево, тем самым заменяя «i» элемент массива на «i+1». После чего уменьшаем значение количества элементов в массиве «n» на 1. По завершении работы алгоритма значение переменной «n» равно размеру итогового массива без удаленных значений.

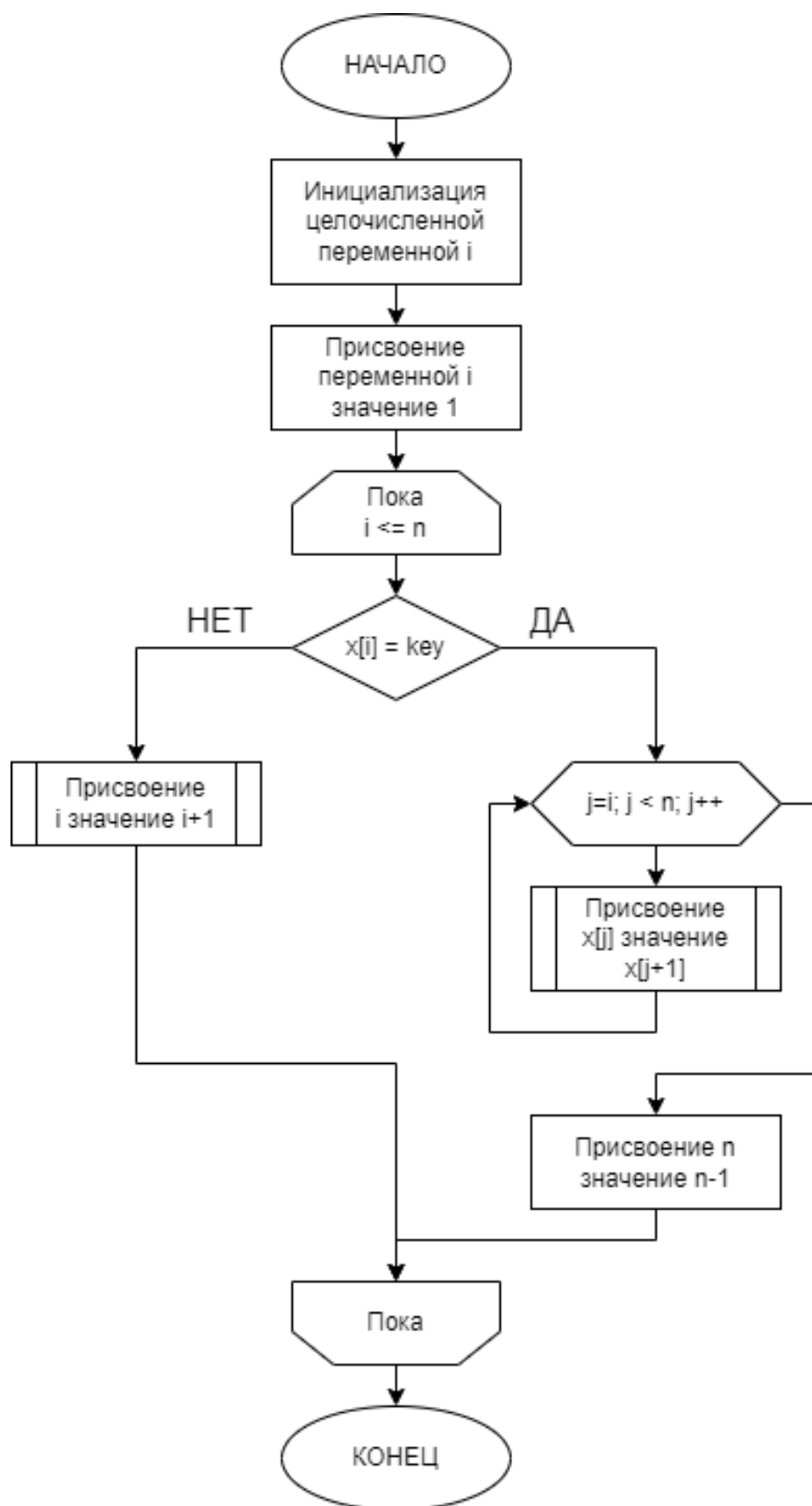


Рисунок 1 — Первый алгоритм

Описание работы второго алгоритма (См. Рисунок 2).

Идем с помощью цикла по элементам массива. Присваиваем «j» элементу значение «i» элемента, где «j» показывает количество элементов массива, не равных «key». Если «i» элемент не равен «key», то увеличиваем значение j на 1. По завершении работы алгоритма присваиваем «n», значению количества элементов в массиве, значение «j».

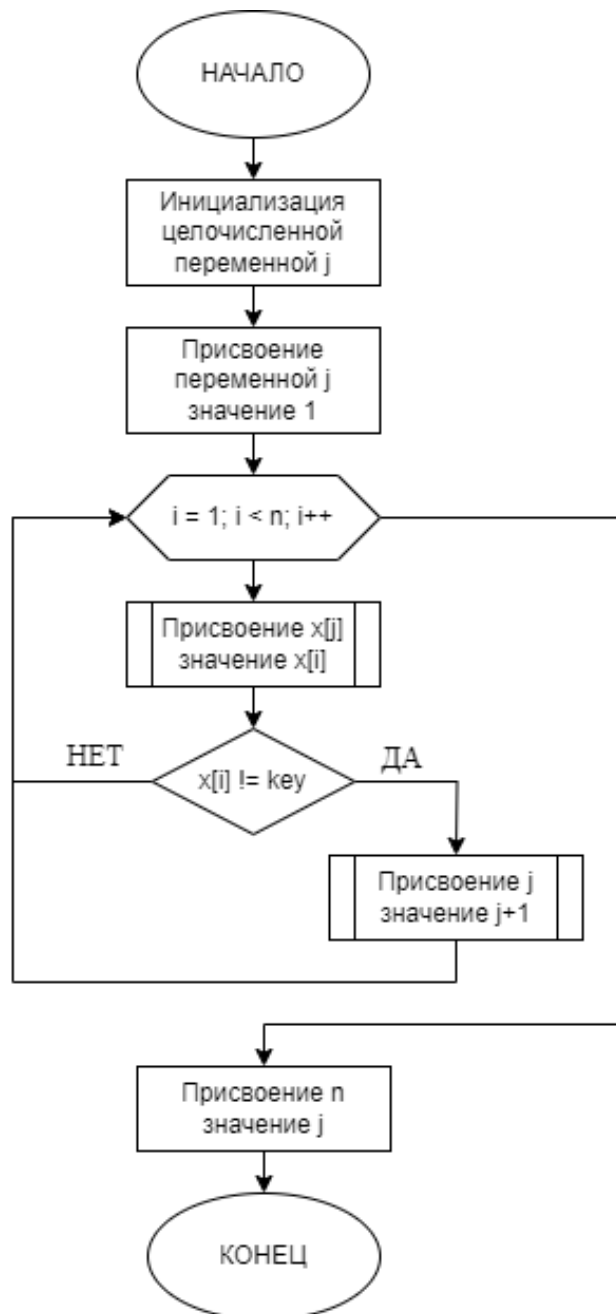


Рисунок 2 — Второй алгоритм

б) Определить для цикла инвариант цикла и доказать его конечность, т. е. доказать корректность цикла.

Для первого алгоритма:

Циклом первого алгоритма является цикл while, где $i \leq n$.

Инвариант: $x[0:i)$ не содержит элементов, равных key, $x[i:n]$ содержит ненулевое количество элементов. С каждой итерацией значение переменной i увеличивается на 1, либо значение переменной n уменьшается на 1, таким образом, с течением времени i будет больше n и цикл завершит свою работу.

Для второго алгоритма:

Циклом второго алгоритма является цикл for, где целочисленная переменная $i = 0; i < n; i++$.

Инвариант: $x[i:j)$ не содержит элементов, равных key. С каждой итерацией значение переменной i увеличивается на 1, в следствие чего с течением времени i будет больше n и цикл завершит свою работу.

с) Вычислительная сложность алгоритма.

Вычислительная сложность первого алгоритма (См. Таблица 1).

Таблица 1 — Вычислительная сложность первого алгоритма.

Оператор	Количество действий оператора	
	В лучшем случае	В худшем случае
$i = 1$	1	1
while $i \leq n$	$n+1$	$n+1$
if $x[i] = \text{key}$	n	n
for $j = 1$ to $n-1$	0	$\sum j+1 = n * (n + 1) / 2$
$x[j] = x[j+1]$	0	$\sum j = n * (n - 1) / 2$
$n--;$	0	n
$i--;$	n	0

Для лучшего случая сложность $T(n) = 3n + 1$, где $n = 10$:

$T(n) = 31$, где $n = 100$: $T(n) = 301$.

Для худшего случая сложность $T(n) = 3n + 2$, где $n = 10$:

$T(n) = 132$, где $n = 100$: $T(n) = 10302$.

Таким образом, для среднего случая, где $n = 10$, временная сложность будет составлять:

$T(n) = 96$, где $n = 100$: $T(n) = 530$.

Вычислительная сложность второго алгоритма (См. Таблица 2).

Таблица 2 – Теоретическая сложность второго алгоритма

Оператор	Количество действий оператора	
	В лучшем случае	В худшем случае
$j = 1$	1	1
for $i = 1$ to n	$n + 1$	$n + 1$
$x[j] = x[i]$	n	n
if $x[i] \neq \text{key}$	n	n
$j++$;	0	n
$n = j$	1	1

Для лучшего случая сложность $T(n) = 3n + 3$, где $n = 10$:

$T(n) = 33$, где $n = 100$: $T(n) = 303$.

Для худшего случая сложность $T(n) = 4n + 3$, где $n = 10$:

$T(n) = 43$, где $n = 100$: $T(n) = 403$.

Таким образом, для среднего случая временная сложность будет составлять:

$T(n) = 38$, где $n = 100$: $T(n) = 353$.

1.3 РЕАЛИЗАЦИЯ АЛГОРИТМА В ВИДЕ ФУНКЦИИ И ОТЛАДКА НА МАССИВЕ ПРИ $n = 10$, $n = 100$. ВКЛЮЧЕНИЕ В ФУНКЦИЮ ПОДСЧЕТА СУММАРНОГО КОЛИЧЕСТВА ВЫПОЛНЕННЫХ СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ ЭЛЕМЕНТОВ ПРИ РЕШЕНИИ ЗАДАЧИ УДАЛЕНИЯ КЛЮЧЕВЫХ ЭЛЕМЕНТОВ.

Реализации первого и второго алгоритмов в виде функций (См. Рисунок 3-4).


```

void dell(int* x, int n, int key) {
    int sravn = 0;
    int* y = new int[n];
    for (int i = 0; i < n; i++) {
        y[i] = x[i];
    }
    int i = 0;
    while (i < n) {
        sravn++;
        if (y[i] == key) {
            sravn++;
            for (int j = i; j < n - 1; j++) {
                y[j] = y[j + 1];
                sravn+=2;
            }
            n--;
        }
        else {
            sravn++;
            i++;
        }
    }
    for (int g = 0; g < n; g++) {
        cout << y[g] << " ";
    }
    cout << "\n";
    cout << "КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 1 --> " << sravn << "\n";
    delete[] y;
}

```

Рисунок 3 – Реализация первого алгоритма в виде функции

```

void del2(int* x, int n, int key) {
    int sravn = 0;
    int* y = new int[n];
    for (int i = 0; i < n; i++) {
        y[i] = x[i];
    }
    int j = 0;
    for (int i = 0; i < n; ++i) {
        y[j] = y[i];
        sravn+=2;
        if (y[i] != key) {
            sravn++;
            j++;
        }
        else {
            sravn++;
        }
    }
    n = j;
    for (int i = 0; i < n; ++i) {
        cout << y[i] << " ";
    }
    cout << "\n";
    cout << "КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 2 --> " << sravn << "\n";
    delete[] y;
}

```

Рисунок 4 – Реализация первого алгоритма в виде функции

1.4 РЕАЛИЗАЦИЯ ФУНКЦИИ: ЗАПОЛНЕНИЕ МАССИВА ДАТЧИКОМ СЛУЧАЙНЫХ ЧИСЕЛ, ВЫВОД МАССИВА НА ЭКРАН МОНИТОРА.

В оба алгоритма включена функция вывода на экран (См. Рисунок 3 — 4).

В качестве датчика случайных чисел используется функция «rand» из стандартной библиотеки C++ (См. Рисунок 5).

```

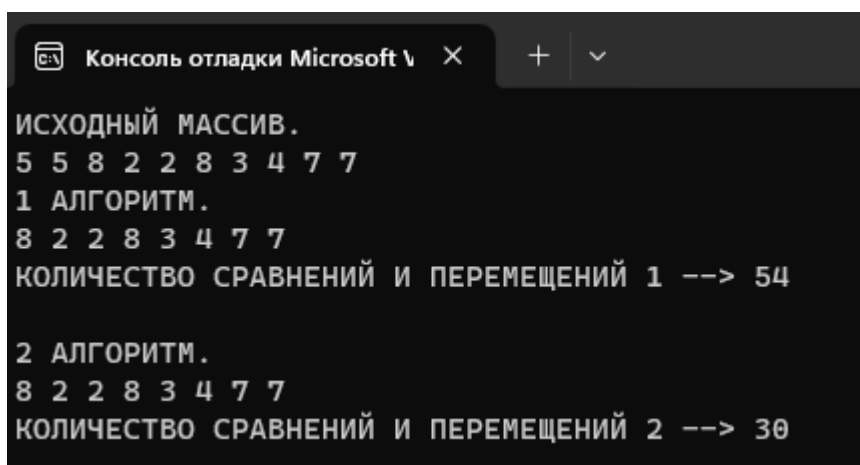
for (int i = 0; i < n; i++) {
    x[i] = rand() % random;
}

```

Рисунок 5 — функция rand

1.5 ТЕСТИРОВАНИЕ АЛГОРИТМА В СИТУАЦИЯХ: А) СЛУЧАЙНОЕ ЗАПОЛНЕНИЕ МАССИВА; Б) ВСЕ ЭЛЕМЕНТЫ МАССИВА ДОЛЖНЫ БЫТЬ УДАЛЕНЫ; В) НИ ОДИН ЭЛЕМЕНТ НЕ УДАЛЯЕТСЯ. СРАВНЕНИЕ РЕЗУЛЬТАТОВ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ЭТИХ СЛУЧАЕВ.

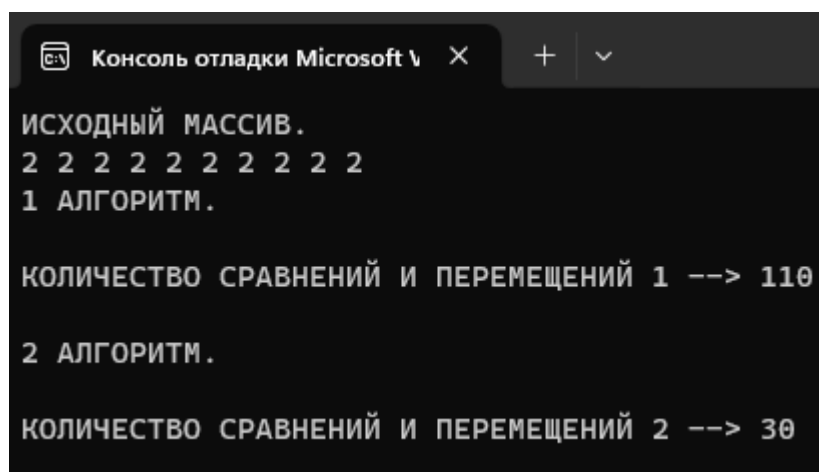
Результаты работы первого алгоритма представлены на рисунках 6 — 8.



```
Консоль отладки Microsoft v X + v
ИСХОДНЫЙ МАССИВ.
5 5 8 2 2 8 3 4 7 7
1 АЛГОРИТМ.
8 2 2 8 3 4 7 7
КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 1 --> 54

2 АЛГОРИТМ.
8 2 2 8 3 4 7 7
КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 2 --> 30
```

Рисунок 6 – Результаты работы первого и второго алгоритмов при случайном заполнении массива



```
Консоль отладки Microsoft v X + v
ИСХОДНЫЙ МАССИВ.
2 2 2 2 2 2 2 2 2 2
1 АЛГОРИТМ.

КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 1 --> 110

2 АЛГОРИТМ.

КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 2 --> 30
```

Рисунок 8 – Результаты работы первого и второго алгоритмов, когда все значения должны быть удалены

```

Консоль отладки Microsoft V X + v
ИСХОДНЫЙ МАССИВ.
6 5 3 6 7 8 6 1 5 6
1 АЛГОРИТМ.
6 5 3 6 7 8 6 1 5 6
КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 1 --> 20

2 АЛГОРИТМ.
6 5 3 6 7 8 6 1 5 6
КОЛИЧЕСТВО СРАВНЕНИЙ И ПЕРЕМЕЩЕНИЙ 2 --> 30

```

Рисунок 8 – Результаты работы первого и второго алгоритмов, когда ничего не удаляется

1.6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Результаты сравнения теоретической сложности двух алгоритмов при $n = 10$ и практической (См. Таблица 3).

Таблица 3 – Таблица сравнения теоретической и практической сложностей алгоритмов.

№ АЛГОРИТМ А	Количество действий, где $n = 10$			
	Теоретически		Практически	
	В лучшем случае	В худшем случае	В лучшем случае	В худшем случае
1	31	132	35	145
2	33	43	34	44

1.7 ЁМКОСТНАЯ СЛОЖНОСТЬ

В первом, и во втором алгоритмах используются две переменные (i, j) целого типа, поэтому их ёмкостная сложность равна двум единицам (двум ячейкам памяти).

1.8 ВЫВОД ПО ПЕРВОМУ ЗАДАНИЮ

На основании проведенного анализа можно сделать вывод, что второй алгоритм гораздо эффективнее первого, поскольку его временная сложность в худшем случае линейно зависит от « n », в то время как временная сложность первого алгоритма зависит от « n » уже квадратично.

РЕШЕНИЕ ЗАДАНИЯ 2

Выполнить задание и создать алгоритм в зависимости от варианта.

2.1 ФОРМУЛИРОВКА ЗАДАЧИ

Сложение двух матриц.

2.2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ ЗАДАЧИ

а) Устное описание и описание блок-схемой.

Матрицы можно складывать только в том случае, если они одинаковы по размеру. СТРОГО одинаковы. Пользователь вручную задает количество строк и столбцов для обеих матриц. Следовательно, нам нужно сделать проверку ввода пользователем для второй матрицы, чтобы соблюдалось правило. Если все введено верно, то для пользователя доступны 2 метода ввода элементов матрицы, а именно вручную и случайным образом. В случайном методе диапазон чисел составляет от 0 до 9. Далее создается третья матрица «SUM» по параметрам первой и второй матрицы. Запускается цикл «for», с помощью него проходим по новой матрице «SUM» и присваиваем элементу с индексами «i» и «j» значение «matrix1[i][j] + matrix2[i][j]». Выводим на экран три матрицы: #1 - «matrix1», #2 - «matrix2», а также результирующую матрицу - «SUM».

Блок-схема алгоритма заполнения результирующей матрицы (См. Рис. 9)

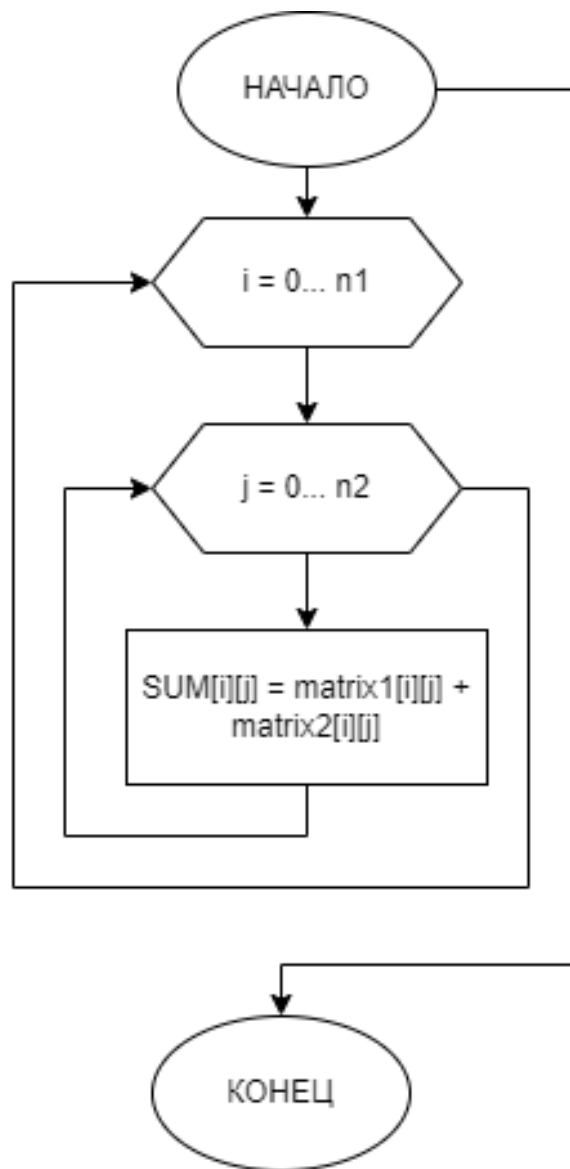


Рисунок 9 – Алгоритм формирования элементов результирующей матрицы

б) Циклами алгоритма являются 2 цикла «for», первый от i до $n1$, второй от i до $n2$.

Инвариант: элемент $SUM[i][j]$ равен сумме элементов $matrix1[j][i]$ и $matrix2[i][j]$ перед каждой следующей итерацией цикла.

С каждой итерацией значение переменной i увеличивается на 1 и значение j также увеличивается на 1, таким образом, с течением времени j будет больше $n1$ и внутренний цикл закончит свою работу, а также j будет больше $n2$ и внешний цикл в свою очередь тоже завершит свою работу.

с) Определим сложность алгоритма. (См. Таблица 4).

Таблица 4 — Теоретическая сложность алгоритма.

Оператор	Количество выполненных операций	
	Лучший случай	Худший случай
for i to n1	1	n1 + 2
for j to n2	2	(n2+1)*(n2+1)
SUM[i][j] = matrix1[i][j] + matrix2[i][j]	2	n1*n2

Для лучшего случая сложность $T(n1, n2) = 3$

$n1 = n2 = 10 : T(n1, n2) = 5;$

$n1 = n2 = 100 : T(n1, n2) = 5.$

Для худшего случая сложность $T(n1, n2) = 2 + 3n1 + 2$

$n1 = n2 = 10 : T(n1, n2) = 232$

$n1 = n2 = 100 : T(n1, n2) = 20302$

Таким образом, для среднего случая временная сложность будет составлять:

$n1 = n2 = 10 : T(n1, n2) = 117$

$n1 = n2 = 100 : T(n1, n2) = 10152$

2.3 РЕАЛИЗАЦИЯ ФУНКЦИИ

Реализация алгоритма решения задачи в виде одной функции. (См. Рисунок 10).

```
for (int i = 0; i < n1; i++) {
    for (int j = 0; j < n2; j++) {
        SUM[i][j] = matrix1[i][j] + matrix2[i][j];
    }
}
```

Рисунок 10 – Реализация алгоритма

2.4 ТЕСТИРОВАНИЕ АЛГОРИТМА

Результаты работы алгоритма представлены на рисунках 11-13.

МАТРИЦА #1.		
9	5	5
0	2	1
6	5	4
МАТРИЦА #2.		
0	0	1
8	9	6
2	5	7
РЕЗУЛТАТ.		
9	5	6
8	11	7
8	10	11

Рисунок 11 – Результат #1

МАТРИЦА #1.			
7	7	9	0
2	1	8	2
4	5	0	6
6	5	3	1
МАТРИЦА #2.			
6	1	3	3
8	2	7	2
7	0	3	4
6	3	3	9
РЕЗУЛТАТ.			
13	8	12	3
10	3	15	4
11	5	3	10
12	8	6	10

Рисунок 12 – Результат #2

МАТРИЦА #1.					
2	8	7	2	8	
4	3	3	7	5	
7	6	2	9	7	
4	6	4	0	9	
6	2	2	3	0	
МАТРИЦА #2.					
3	9	8	1	5	
7	6	5	9	6	
0	2	8	0	3	
5	5	2	6	7	
7	8	1	2	3	
РЕЗУЛЬТАТ.					
5	17	15	3	13	
11	9	8	16	11	
7	8	10	9	10	
9	11	6	6	16	
13	10	3	5	3	

Рисунок 13 – Результат #3

2.5 ЁМКОСТНАЯ СЛОЖНОСТЬ

В алгоритме создаются две переменные целого типа («i» и «j»), поэтому ёмкостная сложность алгоритма будет составлять две единицы (две ячейки памяти).

2.6 ВЫВОД ПО ВТОРОМУ ЗАДАНИЮ

В результате выполнения второго задания был создан и применен алгоритм для того, чтобы складывать две матрицы.

ОБЩИЙ ВЫВОД

В ходе выполнения работы я научился теоретически и практически определять сложность алгоритма и на основе результатов выбирать из нескольких алгоритмов более эффективный. Также я научился самостоятельно составлять алгоритм по заданным условиям.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.
2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796