



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

РТУ МИРЭА

Отчет по выполнению практического задания №7

Тема:

«Рекурсивные алгоритмы и их реализация»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент:

Подоплелов А.С.

Группа:

ИНБО-21-23

Москва - 2024

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ.....	3
1.1 УСЛОВИЕ ЗАДАНИЯ.....	3
1.2 ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ.....	3
РЕШЕНИЕ ЗАДАНИЯ 1.....	5
2.1 ИТЕРАЦИОННЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ.....	5
2.2 ТЕОРЕТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА.....	6
2.3 РЕАЛИЗАЦИЯ И ОТЛАДКА РЕКУРСИВНОЙ ФУНКЦИИ.....	6
2.4 ГЛУБИНА РЕКУРСИИ И СЛОЖНОСТЬ РЕКУРСИВНОГО АЛГОРИТМА.....	7
2.5 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ.....	8
2.6 ВЫВОД ПО ПЕРВОМУ ЗАДАНИЮ.....	9
РЕШЕНИЕ ЗАДАНИЯ 2.....	10
2.1 РЕАЛИЗАЦИЯ ОСНОВНЫХ ФУНКЦИЙ.....	10
2.2 ОСНОВНАЯ ПОДПРОГРАММА И ТЕСТИРОВАНИЕ ВСЕЙ ПРОГРАММЫ.....	12
ВЫВОД ПО ВТОРОМУ ЗАДАНИЮ.....	14
ОБЩИЙ ВЫВОД.....	15
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ.....	16

ПОСТАНОВКА ЗАДАЧИ

Цель работы: получить знания и практические навыки по разработке и реализации рекурсивных алгоритмов.

1.1 УСЛОВИЕ ЗАДАНИЯ

Разработать и протестировать рекурсивные функции в соответствии с задачами варианта.

Мой индивидуальный вариант 7:

1. Найти максимальный элемент в массиве из n элементов.
2. Создание очереди на однонаправленном списке.

1.2 ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ

1) Требования к выполнению первой задачи варианта:

- приведите итерационный алгоритм решения задачи;
- реализуйте алгоритм в виде функции и отладьте его;
- определите теоретическую сложность алгоритма;
- опишите рекуррентную зависимость в решении задачи;
- реализуйте и отладьте рекурсивную функцию решения задачи;
- определите глубину рекурсии, изменяя исходные данные;
- определите сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии;
- приведите для одного из значений схему рекурсивных вызовов;
- разработайте программу, демонстрирующую выполнение обеих функций и покажите результаты тестирования.

2) Требования к выполнению второй задачи варианта:

- рекурсивную функцию для обработки структуры списка согласно варианту. Информационная часть узла – простого типа – целого;
- для создания списка может быть разработана простая или рекурсивная функция по желанию (в тех вариантах, где не требуется рекурсивное создание списка);
- определите глубину рекурсии;

- определите теоретическую сложность алгоритма;
- разработайте программу, демонстрирующую работу функций и покажите результаты тестов.

3) Составить отчет по выполненному заданию.

РЕШЕНИЕ ЗАДАНИЯ 1

2.1 ИТЕРАЦИОННЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

Цель: разработать алгоритм поиска максимального элемента в массиве из n элементов с помощью языка C++.

Данный алгоритм будет состоять из:

- одного цикла «for», с помощью которого будет происходить проход по массиву;
- инициализации переменной «max», которой изначально будет присвоено значение элемента заданного массива с индексом «0».
- результатом работы будет возврат целочисленной переменной «max».

Представим блок-схема алгоритма «Поиск максимального элемента в массиве из n элементов» на рисунке 1.

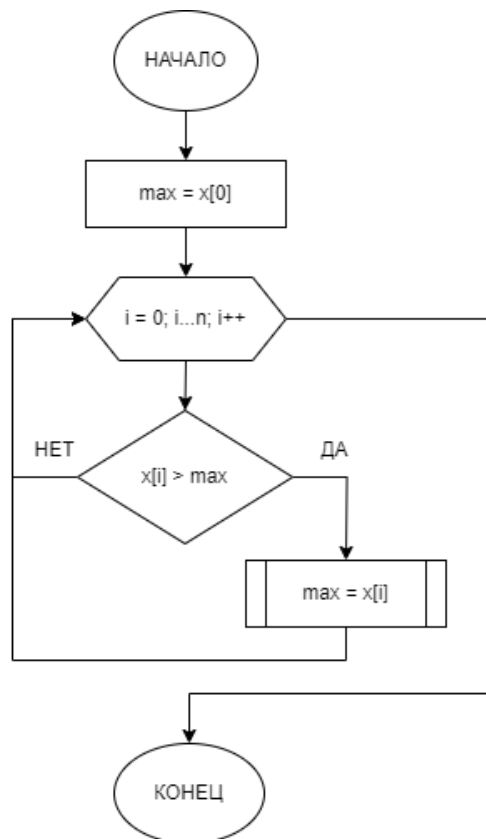


Рисунок 1 — Блок-схема алгоритма «Поиск максимального элемента в массиве из n элементов»

Реализуем данный алгоритм с помощью языка C++ (См. Рисунок 2).

```

int poisk_max(int* x, int n) {
    int max = x[0];
    for (int i = 0; i < n; ++i) {
        if (x[i] > max) {
            max = x[i];
        }
    }
    return max;
}

```

Рисунок 2 — Реализация алгоритма «Поиск максимального элемента в массиве из n элементов» с помощью языка C++

2.2 ТЕОРЕТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА

Единственный способ точно найти самое большое число в случайном массиве будет перебор каждого числа в поисках максимума. Поэтому сложность такого алгоритма – $O(n)$.

Таблица 1 — Теоретическая сложность алгоритма.

n	T(n), мс	$T_n(n) = C_\phi + M_\phi$
100	0	3500
1000	0	35000
10_000	0	350000
100_000	1	3500000
1_000_000	4	35000000

2.3 РЕАЛИЗАЦИЯ И ОТЛАДКА РЕКУРСИВНОЙ ФУНКЦИИ

Перейдем к рекурсивной части реализации данного алгоритма.

Рекурсивная зависимость будет выражается в том, что пока длина массива, который мы передаем в качестве параметра рекурсивной функции, не будет равна 1, то рекурсивная функция не прекратит свою работу.

Реализуем рекурсивную функцию алгоритма «Поиск максимального элемента в массиве из n элементов» с помощью языка C++ (См. Рисунок 3).

```

//Рекурсивная функция
int func(int* arr, int n) {
    int max = arr[0];
    if (n == 1)
        max = arr[0];
    else {
        int prevResult = func(arr, n - 1);
        if (prevResult > arr[n - 1]) {
            max = prevResult;
        }
        else {
            max = arr[n - 1];
        }
    }
    return max;
}

```

Рисунок 2 — Реализация рекурсивной функции алгоритма «Поиск максимального элемента в массиве из n элементов» с помощью языка C++

2.4 ГЛУБИНА РЕКУРСИИ И СЛОЖНОСТЬ РЕКУРСИВНОГО АЛГОРИТМА

Глубина рекурсии - это количество раз, которое функция вызывает саму себя в процессе выполнения.

Определим глубину рекурсии нашего алгоритма с помощью добавления счётчика в нашу рекурсивную функцию в то место, где происходит вход в функцию с измененным параметром (См. Таблица 2).

Таблица 2 — Глубина рекурсии при различных входных данных

n	Глубина рекурсии
100	110
1000	1070
10_000	13700
100_000	Невозможно измерить
1_000_000	Невозможно измерить

Сложность рекурсивного алгоритма будет вычисляться по формуле $O(n \cdot \log_n n)$.

Приведем схему рекурсивных вызовов, при $n = 10$ (См. Рисунок 3).

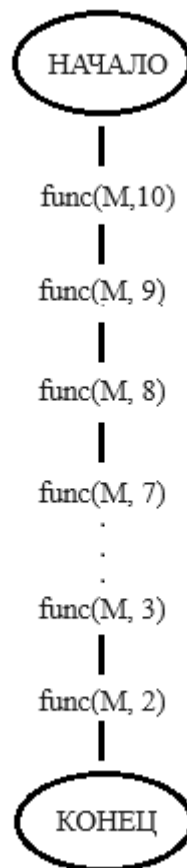


Рисунок 3— Реализация схемы рекурсивных вызовов, при $n = 10$

2.5 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Реализуем основную подпрограмму, где будет осуществляться выполнение двух функций для поиска максимального элемента массива и проведем тесты, при $n = 10$, $n = 100$, $n = 1000$.

```
int main() {
    srand(time(NULL));
    cout << "Start!" << "\n";
    int n;
    cin >> n;
    int* M = new int[n];
    for (int i = 0; i < n; i++) {
        M[i] = rand();
    }
    print(M, n);
    cout << "MAX1 = " << poisk_max(M, n) << "\n";
    cout << "MAX2 = " << func(M, n) << "\n";
}
```

Рисунок 4 – Реализация основной функции main


```

Start!
10
32278 42 21595 24177 10849 19818 16246 18636 31627 28532
MAX1 = 32278
MAX2 = 32278

```

Рисунок 5 – Результат работы программы, при n = 10

```

Start!
100
32434 20445 27107 32336 14428 11693 1803 9199 4984 28337 10813 28643 22082 23714 16061 5038 21060 4276 3965 10951 39 326
99 24979 15168 3530 8174 4843 10662 4753 7147 4409 9257 24064 14076 15666 18684 5256 18073 3580 3071 3744 25902 2275 809
2 11800 16926 24909 25838 7594 13063 21744 10025 11987 27208 6210 9782 18913 30400 26417 13469 9292 28889 3605 27816 126
17 30917 32483 16582 157 27144 18083 27437 32687 26679 30500 195 20706 10495 22766 11071 11553 8040 31096 20107 956 2773
9 27035 7533 19570 14435 16455 24935 16335 23216 13685 31196 27064 29025 26972 22131
MAX1 = 32699
MAX2 = 32699

```

Рисунок 6 – Результат работы программы, при n = 100

```

Start!
1000
32523 19740 17920 26686 30778 19411 20383 28467 24813 26179 6819 29832 30088 11712 25425 14414 24250 10237 20626 23525 13346 24700 4396 7954 4594 11849 23118 29521 12419 488 19899 22893 6985 20364 2858 15693 2
7993 27818 28999 17425 14120 24332 20434 16478 29078 13066 5703 16629 8747 32630 14017 28043 5280 22711 19806 9685 19873 1369 18633 5858 6865 24067 6750 15021 25615 27049 12686 30195 0696 2562 18053 18498 7596
10517 14347 16079 24021 12704 168 10377 25339 1617 1804 22554 9277 5832 19888 15320 19164 19716 17842 28261 19332 16429 18704 6398 19426 25563 32755 24092 21277 5669 11663 27629 25691 20380 24343 30479 25598
19545 23742 2657 29556 30453 12860 26939 6953 3127 16739 29237 29484 26882 18131 140469 9893 7248 7706 18312 31852 27274 4384 17566 12755 6874 14019 25403 3501 11635 9406 24379 8729 18479 38927 29046 20860 2861
7 32025 29333 9037 24908 27591 8228 25623 8139 5618 14312 1083 2643 29124 23922 18132 3614 21470 32244 25375 18697 22520 9812 21050 12360 15559 32639 10796 3929 14116 28586 10790 29494 186 18897 23561 21967 92
48 24432 28455 7607 20696 13110 12366 17950 12374 13797 17155 17151 2805 24960 11 38106 14919 29023 7472 17701 12028 24706 577 12223 3589 8742 28507 26533 10489 18582 22631 20648 21943 3652 6628 29151 26957 29
976 27367 12185 23064 28680 12383 24081 19757 19177 5534 6518 4856 17540 3605 11633 22307 3821 9311 23369 1977 10768 14978 6013 14220 19597 8049 23744 32724 20166 7637 4009 9496 19599 26636 19116 850 28428 139
4 9081 1358 6615 9400 20607 11091 28581 30835 3732 28561 7015 32399 26591 17034 16871 23507 12582 16628 14473 11310 10493 17578 1822 26704 28094 7663 2871 20404 16191 21579 23040 27044 12673 25215 12788 8922 2
4234 24735 22463 25583 29174 23650 6809 29400 20329 28862 29395 22726 18184 11745 28335 28070 2120 7071 24949 7027 27079 16852 2203 15904 22842 2388 7205 17627 1958 9833 31461 30867 9684 9254 27685 3080 17663
5402 9514 22338 12239 22686 2555 26868 20363 15154 23595 18146 29066 30237 19443 3121 234 21076 8068 17182 2181 29341 25754 8257 4385 2292 24661 787 12263 31534 12613 24495 22807 15068 22467 30165 19193 21618
6280 31828 9495 31145 28024 3817 32681 2849 30911 17475 5969 25931 11807 25289 24840 29830 9864 17790 6744 21640 20970 4963 10701 32512 31559 830 25510 19833 21260 3226 1743 21122 29601 10767 1556 19980 5183 4
291 29071 16659 24066 305 5218 25729 5951 5417 6828 1871 6423 12856 21429 6125 17228 28298 25708 23780 25172 19949 32276 8378 15276 19864 27962 21781 13423 20492 5354 28964 17959 2937 10786 3899 13795 13444 17
287 3517 16778 17991 4439 14143 10190 2960 20895 13802 11300 19856 21678 22049 17044 19269 9673 23652 14924 24279 11296 29301 14362 29619 22529 28681 21961 12618 12364 224 32395 963 21543 29385 6056 22050 1765
0 0376 16566 15753 3198 25920 2070 21934 4085 16305 15406 15052 15405 31810 31635 23136 5403 21736 8137 20876 14785 10899 13020 11150 12019 15044 31265 18550 22953 31083 2114 6262 22086 951 14358 22406 24761 1
2496 22229 16219 23991 970 13130 4109 1704 20507 12330 20026 7937 15031 20644 23072 2195 22137 29497 4078 29953 24517 11982 4133 19319 23404 1536 22112 5785 7375 28783 12242 3770 30536 7166 2095 5199 7877 2049
8 29326 11939 31670 3256 15469 25292 30250 29652 12798 30429 31796 14904 30593 30045 18324 26298 7257 1272 9875 1929 19139 26741 19503 20929 14652 15080 27512 24380 16643 15401 8762 32428 15831 16772 30428 189
18 6914 7858 28048 30337 12694 6920 30861 5280 23543 18159 18433 12300 25241 17812 15530 1926 9739 24289 543 20059 18085 28075 5162 11320 26334 32301 13169 28408 369 12469 3413 9308 21504 13469 6939 2514 25993
28541 31817 4122 14200 9575 20563 23789 5182 12467 21367 1182 3137 11716 17126 28077 26948 4071 19492 4762 27336 10851 2993 5623 226 28092 24297 23745 17775 10868 3369 8944 3070 21155 3123 15049 29965 27989 2
5556 900 30859 19901 29082 378 5404 11520 9923 1760 4337 31080 16943 9737 12745 10223 30708 11000 12226 23071 24416 14270 10728 10703 26107 31091 8325 1761 274 13538 29029 9209 6602 32258 13556 2074 32343 1217
19060 22676 23068 13596 28612 23938 38995 32301 5354 5701 23152 10860 17774 29233 22145 32250 3522 389 23513 3890 17567 12961 1965 3456 1681 28531 7550 27254 2741 12083 8696 6428 13505 20560 7451 22408 2118 2
0138 18328 13999 24882 10753 10110 21627 5529 4269 24597 31774 25781 8848 14602 8357 22315 24650 8397 20604 29839 6827 11832 11041 29227 30318 25969 19763 16124 22564 17425 4429 4350 19566 22662 28471 22442 16
495 26806 11596 14268 7809 27620 721 29753 8999 23512 13445 26801 32324 8506 24441 30913 5258 8000 26146 6365 30004 8841 13229 21580 22520 14152 18834 29812 14135 10916 19308 11608 23160 15199 32008 29535 2587
6 15798 26300 24151 15666 13061 5114 15801 24310 12732 8228 16943 26662 22493 1442 24659 9966 1799 25395 20666 23476 29406 11136 20705 21358 4283 29431 2886 7426 20674 20438 17419 4558 27907 134 14614 20122 20
326 19409 20172 14038 20251 20857 29664 25560 21236 14031 20086 27394 5300 30316 1101 4028 1940 9701 31164 18009 12691 4479 3700 20726 23000 30175 22054 13047 26243 11408 3711 22853 16567 11841 18345 9767 1097
4 8479 9436 19225 8441 28350 23469 1242 8536 3073 13069 21124 6458 27894 29853 9666 2036 22180 18432 25405 27863 843 5787 18263 26893 19546 23704 796 14532 20323 2865 9589 18310 16575 24619 9375 25608 24746 22
176 7132 22203 18073 5378 4457 24639 30414 31837 22703 25254 30249 15580 25789 14872 20545 5007 7926 2234 17331 5403 5639 31188 2234 11155 6429 16436 26112 9430 4948 21453 1972 120 15218 12964 19077 22869 2760
5 10477 32494 32643 13994 25712 10138 26136 2656 22322 30953 7084 31324 11602 31628 7936 32751 23089 30574 29215 18309 213 19083 26083 24095 15248 8709 16060 21251 1756 18804 5805 8625 21386 1150 8343 30746 10
52 0897 12474 11533 22895 23916
MAX1 = 32755
MAX2 = 32755

```

Рисунок 7 – Результат работы программы, при n = 1000

В результате тестирования никаких ошибок не обнаружено, программа работает успешно.

2.6 ВЫВОД ПО ПЕРВОМУ ЗАДАНИЮ

В процессе выполнения индивидуального задания было изучено много новой информации о рекурсивных функциях.

РЕШЕНИЕ ЗАДАНИЯ 2

Цель: разработать алгоритм, который будет создавать очередь на однонаправленном списке.

Информационная часть узла — простого типа — целого.

2.1 РЕАЛИЗАЦИЯ ОСНОВНЫХ ФУНКЦИЙ

Реализуем узел и инициализируем три переменных: указатель на голову очереди, указатель на хвост очереди и количество элементов с помощью языка C++ (См. Рисунок 8).

```
struct main_{  
    int data;  
    struct main_* next;  
};  
main_* head, * tail;  
int cnt = 0; //Количество элементов
```

Рисунок 8 – Реализация схемы узла и необходимых переменных с помощью языка C++

Далее реализуем все необходимые функции для реализации очереди однонаправленного списка (См. Рисунок 8-11).

```
//Создание очереди  
void Create() {  
    head = (main_*)malloc(sizeof(main_));  
    head->next = NULL;  
    tail = head;  
    cnt = 0;  
}  
  
//Проверка на пустой список  
void is_empty() {  
    if (cnt == 0) {  
        cout << "Список пуст!" << endl;  
        system("pause");  
        menu();  
    }  
}  
  
/*Функция, которая добавляет элемент в конец*/  
void Add_last(main_* temp) {  
    tail->data = temp->data;  
    tail->next = (main_*)malloc(sizeof(main_));  
    tail = tail->next;  
    tail->next = NULL; //Обнуление указателя на следующий элемент  
    cnt++;  
}
```

Рисунок 9 – Реализация функций Create(), is_empty() и Add_last()

```

/*Считывание элемента из "головы" и перенос в "хвост"*/
void Head_to_tail() {
    main_* buff = head;
    tail->data = buff->data;
    tail->next = (main_*)malloc(sizeof(main_));
    tail = tail->next;
    tail->next = NULL;
    buff = head->next;
    free(head);
    head = buff;
}

/*Вывод списка*/
void Tablitsa() {
    for (int i = 0; i < cnt; i++) {
        printf("%d) %d\n", i + 1, head->data);
        Head_to_tail();
    }
}

/*Вывод информации для главного меню*/
void show_menu() {
    system("cls");
    cout << "1 - Добавить элемент" << endl;
    cout << "2 - Просмотр одного элемента" << endl;
    cout << "3 - Просмотр всех элементов" << endl;
    cout << "4 - Выход" << endl;
}

/*Загрузка данных из очереди в массив*/
void Loading(struct main_* array) {
    for (int i = 0; i < cnt; i++) {
        array[i].data = head->data;
        Head_to_tail();
    }
}

```

Рисунок 10 – Реализация функций Head_to_tail(), Tablitsa(), show_menu() и Loading()

```

void input() {
    system("cls");
    main_ queue;
    int num = 0;
    cout << "Введите число: ";
    cin >> num;
    queue.data = num;
    Add_last(&queue);
}

/*Информация об элементе*/
void inf() {
    int Num = 0;
    system("cls");
    cout << "Введите номер элемента." << "\n";
    cin >> Num;
    Num = Num - 1;
    main_* array = new main_[cnt];
    Loading(array);
    cout << "Число-> " << array[Num].data << endl;
}

```

Рисунок 11 – Реализация функций input() и inf()

После реализации необходимых функций определим теоретическую сложность алгоритма, которая будет равной количеству действий, выполненных пользователем в ходе работы программы.

2.2 ОСНОВНАЯ ПОДПРОГРАММА И ТЕСТИРОВАНИЕ ВСЕЙ ПРОГРАММЫ

Реализуем основную подпрограмму, где будут выполняться остальные подпрограммы (См. Рисунок 12).

```
void menu() {  
    char ch;  
    show_menu();  
    while (1) {  
        ch = _getch();  
        if (ch == 49) {  
            system("cls");  
            input();  
            system("pause");  
            menu();  
        }  
        if (ch == 50) {  
            system("cls");  
            //is_empty();  
            inf();  
            system("pause");  
            menu();  
        }  
        if (ch == 51) {  
            system("cls");  
            //is_empty();  
            Tablitsa();  
            system("pause");  
            menu();  
        }  
        if (ch == 52) {  
            exit(0);  
        }  
    }  
}
```

Рисунок 12 – Реализация основной подпрограммы

Проведем пару тестов данной программы (См. Рисунок 13-14).

```
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
Введите число: 100
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
Введите номер элемента.

1
Число-> 100
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
Введите число: 645
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
Введите номер элемента.

1
Число-> 100
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
Введите номер элемента.

2
Число-> 645
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
1) 100
2) 645
1 - Добавить элемент
2 - Просмотр одного элемента
3 - Просмотр всех элементов
4 - Выход
```

Рисунок 13 – Результаты теста №1

```
1 – Добавить элемент
2 – Просмотр одного элемента
3 – Просмотр всех элементов
4 – Выход
Введите число: 4234
1 – Добавить элемент
2 – Просмотр одного элемента
3 – Просмотр всех элементов
4 – Выход
Введите номер элемента.
1
Число-> 4234
1 – Добавить элемент
2 – Просмотр одного элемента
3 – Просмотр всех элементов
4 – Выход
1) 4234
1 – Добавить элемент
2 – Просмотр одного элемента
3 – Просмотр всех элементов
4 – Выход
Введите число: 5345345
1 – Добавить элемент
2 – Просмотр одного элемента
3 – Просмотр всех элементов
4 – Выход
Введите номер элемента.
2
Число-> 5345345
1 – Добавить элемент
2 – Просмотр одного элемента
3 – Просмотр всех элементов
4 – Выход
1) 4234
2) 5345345
```

Рисунок 14 – Результаты теста №2

Результаты тестирования показали, что программа функционирует успешно.

ВЫВОД ПО ВТОРОМУ ЗАДАНИЮ

Был разработан алгоритм, который будет создавать очередь на однонаправленном списке.

ОБЩИЙ ВЫВОД

Улучшены навыки реализации рекурсивных алгоритмов и
однонаправленных списков.

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