МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"
РТУ МИРЭА

Отчет по выполнению практического задания №2

Тема: Технология реализации алгоритмов с использованием функций (процедурное программирование) в заданиях дисциплины. Статические и динамические массивы. Разработка операций.

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент Антонов А.Д.

группа ИКБО-01-20

Москва 2021

СОДЕРЖАНИЕ

1. Задание 1

Разработать программу выполнения операций над статическим массивом целых чисел. Размер массива 1000 элементов. Предусмотреть ввод значения n – текущий размер массива. Разработать операции для управления массивом и реализовать их функциями. Функции должны принимать входные данные через параметры и возвращать результат, если этого требует алгоритм операции

1.1 Разработать функцию ввода массива из n элементов с клавиатуры.

Предусловие: в алгоритм поступает массив mass и размер массива n.

Постусловие: массив mass будет заполнен с клавиатуры.

```
3  void get(int* mass, int n)
4  {
5      for (int i = 0; i < n; i++)
6          cin >> mass[i];
7  }
```

1.2 Разработать функцию вывода массива из n на монитор

Предусловие: в алгоритм поступает массив mass и размер массива n.

Постусловие: массив mass будет выведен на экран.

```
9  void out(int* mass, int n)
10 {
11     for (int i = 0; i < n; i++)
12         cout << mass[i] << " ";
13 }
```

1.3 Разработать функцию заполнения массива из n элементов, используя датчик случайных чисел.

Предусловие: в алгоритм поступает массив mass и размер массива n.

Постусловие: массив mass будет заполнен случайными числами.

```
15 void Random(int* mass, int n)
16 {
17     for (int i = 0; i < n; i++)
18         mass[i] = rand() % 10;
19 }
```

1.4 Разработать программу и протестировать работу функций

```cpp
#include <iostream>
#include <fstream>
using namespace std;

void get(int* mass, int n) {
    for (int i = 0; i < n; i++)
        cin >> mass[i];
}
void out(int* mass, int n) {
    for (int i = 0; i < n; i++)
        cout << mass[i] << " ";
}
void Random(int* mass, int n) {
    for (int i = 0; i < n; i++)
        mass[i] = rand() % 10;
}
int main() {
    int n;
    cout << "Enter n:";
    cin >> n;
    int* mass = new int[n];
    cout << "Enter array:" << endl;
    get(mass, n);
    cout << "Print array:" << endl;
    out(mass, n);
    Random(mass, n);
    cout <<endl<< "Random array:" << endl;
    out(mass, n);
    return 0;
}
```

Тестирование работы программы:

```
Enter n: 5
Enter array:
1 2 5 6 8
Print array:
1 2 5 6 8
Random array:
1 7 4 0 9
```

Программа работает корректно.

1.5 Разработать функцию поиска первого вхождения значения в массив.

Предусловие: в алгоритм поступает массив mass, размер массива n и искомое значение k.

Постусловие: возвращает -1, если значение в массиве не встретилось. Возвращает число от 0 до n-1 (индекс первого вхождения), если число в массиве встретилось.

```
22  int FirstEntry(int* mass, int n, int k) {
23      int f = -1;
24      for (int i = 0; i < n; i++)
25          if (mass[i] == k)
26          {
27              f = i;
28              break;
29          }
30      return f;
31  }
```

1.5.1 Тестирование функции:

Таблица 1. Нахождение первого вхождения значения

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 5 Массив: 1 1 1 1 1 Искомое значение: 1	Индекс первого вхождения значения: 0	Enter n: 5 Enter array: 1 1 1 1 1 Enter key: 1 Index of first entry: 0
2	Кол-во элементов: 3 Массив: 1 2 3 Искомое значение: 2	Индекс первого вхождения значения: 1	Enter n: 3 Enter array: 1 2 3 Enter key: 2 Index of first entry: 1

3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Искомое значение: 0	-1 (значение в массиве не встречается)	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter key: 0 Index of first entry: -1

1.6 Разработать функцию нахождения индекса первого отрицательного числа в массиве

Предусловие: в алгоритм поступает массив mass и размер массива n.

Постусловие: возвращает -1, если отрицательное число в массиве не встретилось. Возвращает число от 0 до n-1(индекс первого вхождения отрицательного числа), если оно в массиве встретилось.

```
33  int Negative(int* mass, int n) {
34      int f = -1;
35      for (int i = 0; i < n; i++)
36          if (mass[i] < 0)
37          {
38              f = i;
39              break;
40          }
41      return f;
42  }
```

1.6.1 Тестирование функции:

Таблица 2. Нахождение индекса первого отрицательного числа в массиве

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 5 Массив: 1 2 3 -4 5	Индекс первого вхождения отрицательного числа: 3	Enter n: 5 Enter array: 1 2 3 -4 5 Index of first negative: 3
2	Кол-во элементов: 3 Массив: -1 -2 -3	Индекс первого вхождения отрицательного числа: 0	Enter n: 3 Enter array: -1 -2 -3 Index of first negative: 0

3	Кол-во элементов: 4 Массив: 1 2 3 4	-1 (в массиве нет отрицательных чисел)	Enter n: 4 Enter array: 1 2 3 4 Index of first negative: -1

1.7 Разработать функцию поиска всех вхождений в массив

Предусловие: в алгоритм поступает массив mass, размер массива n и искомое значение k.

Постусловие: выводит «No entries», если число не встретилось. Выводит "Index of value:" и числа от 0 до n-1, если число встретилось.

```
44  void AllEntries(int* mass, int n, int k) {
45      bool f = false;
46      for (int i = 0; i < n; i++)
47          if (mass[i] == k)
48          {
49              f = true;
50              cout << "Index of value: "<< i << endl;
51          }
52      if (!f)
53          cout << "No entries";
54  }
```

1.7.1 Тестирование функции

Таблица 3. Поиск всех вхождений

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 3 Массив: 1 1 1 Искомое значение: 1	Index of value: 0 Index of value: 1 Index of value: 2	Enter n: 3 Enter array: 1 1 1 Enter key: 1 Index of value: 0 Index of value: 1 Index of value: 2
2	Кол-во элементов: 5 Массив: 1 2 3 4 5 Искомое значение: 2	Index of value: 1	Enter n: 5 Enter array: 1 2 3 4 5 Enter key: 2 Index of value: 1

3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Искомое значение: 0	No entries	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter key: 0 No entries

1.8 Разработать функцию вставки нового значения в заданную позицию массива.

Предусловие: в алгоритм поступает массив mass, индекс k (k>0 и k<n) и новое значение p.

Постусловие: вставляет новое значение в данную позицию массива

```
56  void NewValue(int* mass, int k, int p) {
57      mass[k] = p;
58  }
```

1.8.1 Тестирование функции

Таблица 4. Вставка нового значения

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 3 Массив: 1 1 1 Индекс: 1 Замена: 8	1 8 1	Enter n: 3 Enter array: 1 1 1 Enter index: 1 Enter new value: 8 Print array: 1 8 1
2	Кол-во элементов: 5 Массив: 1 2 3 4 5 Индекс: 4 Замена: 1	1 2 3 4 1	Enter n: 5 Enter array: 1 2 3 4 5 Enter index: 4 Enter new value: 1 Print array: 1 2 3 4 1
3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Индекс: 6	Out of bounds	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter index: 6 Out of bounds

1.9 Разработать функцию алгоритма удаления со сжатием из массива значения в заданной позиции, сохраняя порядок следования остальных элементов.

Предусловие: в алгоритм поступает массив mass, его размер n, индекс p (p>0 и p<n).

Постусловие: удаляет значение со сжатием

```
60  void Delete(int* mass, int n, int p) {
61      for (int i = p; i < n - 1; i++)
62          mass[i] = mass[i + 1];
63  }
```

1.9.1 Тестирование функции

Таблица 5. Алгоритм удаления со сжатием из массива значения в заданной позиции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Индекс: 0	2 3 4 5 6	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter index: 0 Print array: 2 3 4 5 6
2	Кол-во элементов: 5 Массив: 1 2 3 4 5 Индекс: 3	1 2 3 5	Enter n: 5 Enter array: 1 2 3 4 5 Enter index: 3 Print array: 1 2 3 5
3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Индекс: 6	Такого индекса нет	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter index: 6 Out of bounds

1.10 Разработать функцию алгоритма удаления со сжатием всех вхождений заданного значения из массива. Сложность алгоритма O(n^2).

Предусловие: в алгоритм поступает массив mass, размер массива n и искомое значение k.

Постусловие: возвращает кол-во элементов в обновленном массиве

```
65  int DeleteAll(int* mass,int n, int k) {
66      int counter=0;
67      for (int i = 0; i < n; i++)
68          if (mass[i] == k)
69          {
70              counter++;
71              for (int j = i; j < n - 1; j++)
72                  mass[j] = mass[j + 1];
73              mass[n - 1] = 0;
74              i--;
75          }
76      return n - counter;
77  }
```

1.10.1 Тестирование функции

Таблица 6. Удаление со сжатием всех значений

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 3 Массив: 1 1 1 Искомое значение: 1		Enter n: 3 Enter array: 1 1 1 Enter key: 1 Print array:
2	Кол-во элементов: 5 Массив: 1 2 3 2 5 Искомое значение: 2	1 3 5	Enter n: 5 Enter array: 1 2 3 2 5 Enter key: 2 Print array: 1 3 5

3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Искомое значение: 0	1 2 3 4 5 6	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter key: 0 Print array: 1 2 3 4 5 6

1.10.2 Оценка сложности алгоритма

Введем новую переменную iter, которая будет считать количество операций, совершенных внутренними циклами для оценки сложности.

```
65  int DeleteAll(int* mass, int n, int k) {
66      int counter = 0, iter = 0;
67      for (int i = 0; i < n; i++)
68          if (mass[i] == k)
69          {
70              iter++;
71              counter++;
72              for (int j = i; j < n - 1; j++) {
73                  mass[j] = mass[j + 1];
74                  iter++;
75              }
76              mass[n - 1] = 0;
77              i--;
78          }
79      cout << endl << "Iterations: " << iter << endl;
80      return n - counter;
81  }
```

При n=1

```
Enter n: 1
Enter array:
1
Enter key: 1

Iterations: 1
Print array:
```

При n=10

```
Enter n: 10
Enter array:
1 1 1 1 1 1 1 1 1 1
Enter key: 1

Iterations: 100
Print array:
```

При n=100

```
Enter n: 100
Enter array:
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Enter key: 1

Iterations: 10000
Print array:
```

Можно заметить, что при увеличении значения n на порядок, количество операций увеличивается на 2 порядка, что подтверждается тем, что порядок роста для алгоритма T(n)=Θ(n^2).

1.11 Разработать функцию алгоритма удаления со сжатием всех вхождений заданного значения из массива. Сложность алгоритма O(n).

Предусловие: в алгоритм поступает массив mass, размер массива n и искомое значение k.

Постусловие: возвращает кол-во элементов в обновленном массиве

```
85  int DeleteAllEff(int* mass, int n, int k) {
86      vector<int> n_mass;
87      for (int i = 0; i < n; i++)
88          if (mass[i] != k)
89              n_mass.push_back(mass[i]);
90      for (int i = 0; i < n_mass.size(); i++)
91          mass[i] = n_mass[i];
92      return n_mass.size();
93  }
```

1.11.1 Тестирование функции

Таблица 7. Удаление со сжатием всех значений

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 3 Массив: 1 1 1 Искомое значение: 1		Enter n: 3 Enter array: 1 1 1 Enter key: 1 Print array:
2	Кол-во элементов: 5 Массив: 1 2 3 2 5 Искомое значение: 2	1 3 5	Enter n: 5 Enter array: 1 2 3 2 5 Enter key: 2 Print array: 1 3 5
3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Искомое значение: 0	1 2 3 4 5 6	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter key: 0 Print array: 1 2 3 4 5 6

1.11.2 Оценка сложности алгоритма

Введем новую переменную iter, которая будет считать количество операций, совершенных внутренними циклами для оценки сложности.

```
85  int DeleteAllEff(int* mass, int n, int k) {
86      int iter = 0;
87      vector<int> n_mass;
88      for (int i = 0; i < n; i++)
89      {
90          if (mass[i] != k)
91              n_mass.push_back(mass[i]);
92          iter++;
93      }
94      for (int i = 0; i < n_mass.size(); i++)
95      {
96          mass[i] = n_mass[i];
97          iter++;
98      }
99      cout << endl << "Iterations: " << iter << endl;
100     return n_mass.size();
101 }
```

При n=1

```
Enter n: 1
Random array:
1
Enter key: 1

Iterations: 1
Print array:
```

При n=10

```
Enter n: 10
Random array:
1 7 4 0 9 4 8 8 2 4
Enter key: 3

Iterations: 20
Print array:
1 7 4 0 9 4 8 8 2 4
```

При n=100

```
Enter n: 100
Random array:
1 7 4 0 9 4 8 8 2 4 5 5 1 7 1 1 5 2 7 6 1 4 2 3 2 2 1 6 8 5 7 6 1 8 9 2 7 9 5 4
3 1 2 3 3 4 1 1 3 8 7 4 2 7 7 9 3 1 9 8 6 5 0 2 8 6 0 2 4 8 6 5 0 9 0 0 6 1 3 8
9 3 4 4 6 0 6 6 1 8 4 9 6 3 7 8 8 2 9 1
Enter key: 1

Iterations: 186
Print array:
7 4 0 9 4 8 8 2 4 5 5 7 5 2 7 6 4 2 3 2 2 6 8 5 7 6 8 9 2 7 9 5 4 3 2 3 3 4 3 8
7 4 2 7 7 9 3 9 8 6 5 0 2 8 6 0 2 4 8 6 5 0 9 0 0 6 3 8 9 3 4 4 6 0 6 6 8 4 9 6
3 7 8 8 2 9
```

Можно заметить, что при увеличении значения n на порядок, количество операций также увеличивается на порядок, что подтверждается тем, что порядок роста для алгоритма $T(n)=\Theta(n)$.

2. Задание 2

2.1 Функция вставки нового значения в заданную позицию массива с помощью функции realloc модуля malloc

Предусловие: в алгоритм поступает массив mass, размер массива, индекс k и новое значение p.

Постусловие: в массив на k индекс будет вставлено значение p

```
103 void addForRealloc(int*& mass, int& n, int k, int p) {
104     n++;
105     mass = (int*)realloc(mass, (n) * sizeof(int));
106     for (int i = n; i > k; i--)
107         mass[i] = mass[i - 1];
108     mass[k] = p;
109 }
```

2.1.1 Тестирование функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 3 Массив: 1 1 1 Индекс: 1 Новое значение: 8	1 8 1 1	Enter n: 3 Enter array: 1 1 1 Enter index: 1 Enter new value: 8 Print array: 1 8 1 1
2	Кол-во элементов: 5 Массив: 1 2 3 4 5 Индекс: 4 Новое значение: 8	1 2 3 4 8 5	Enter n: 5 Enter array: 1 2 3 4 5 Enter index: 4 Enter new value: 8 Print array: 1 2 3 4 8 5
3	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Индекс: 6 Новое значение: 7	1 2 3 4 5 6 7	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter index: 6 Enter new value: 7 Print array: 1 2 3 4 5 6 7

2.2 Функция удаления значения в заданной позиции со сжатием массива с помощью функции realloc модуля malloc

Предусловие: в алгоритм поступает массив mass, размер массива n, индекс k (k>0, k<n).

Постусловие: из массива удаляется элемент с индексом k

```
111 void deleteForRealloc(int*& mass, int& n, int k) {
112     int last = mass[n - 1];
113     n--;
114     mass = (int*)realloc(mass, (n) * sizeof(int));
115     for (int i = k; i < n; i++)
116         mass[i] = mass[i + 1];
117     mass[n - 1] = last;
118 }
```

2.2.1 Тестирование функции

Номер теста	Входные данные	Ожидаемый результат	Результат выполнения программы
1	Кол-во элементов: 6 Массив: 1 2 3 4 5 6 Индекс: 0	2 3 4 5 6	Enter n: 6 Enter array: 1 2 3 4 5 6 Enter index: 0 Print array: 2 3 4 5 6
2	Кол-во элементов: 5 Массив: 1 2 3 4 5 Индекс: 3	1 2 3 5	Enter n: 5 Enter array: 1 2 3 4 5 Enter index: 3 Print array: 1 2 3 5
3	Кол-во элементов: 3 Массив: 1 2 3 Индекс: 3	Out of bounds	Enter n: 3 Enter array: 1 2 3 Enter index: 3 Out of bounds

К динамическому массиву были применены функции поиска. Результат применения аналогичен результатам применения функций для статического массива.

ВЫВОД

В процессе выполнения практической работы №2 мы научились работать со статическими и динамическими массивами. Научились увеличивать и уменьшать объём динамического массива с помощью функции realloc модуля malloc.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

- Лекционный материал по структуре и алгоритмам обработки данных Гданского Н.И.
- Методический материал «Примеры определения функций роста».
- Функция malloc // cppstudio [Электронный ресурс]. - http://cppstudio.com/post/856/– (дата обращения: 25.04.2021)