МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

# РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

## ОТЧЕТ ПО САМОСТОЯТЕЛЬНОЙ РАБОТЕ №2

**по дисциплине**
**«Структуры и алгоритмы обработки данных»**

Тема. Абстрактный тип данных и его реализация на одномерном статическом массиве

Выполнил студент группы ИКБО-15-23 Гольд.Д.В.

Принял старший преподаватель Скворцова Л.А.

# Оглавление

# 1. УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА.

Дано множество из n целых чисел. Дан набор задач (операций), которые требуется выполнить над исходным множеством. Набор задач определен в варианте задания табл. 6.

Разработать и реализовать АТД задачи, по управлению множеством посредством операций, указанных в варианте задания. В АТД включить операции по заполнению исходного множества и отображения множества. При разработке алгоритмов операций варианта могут быть выявлены дополнительные алгоритмы, например такие: определить является ли число простым, или определить сумму цифр числа, эти алгоритмы надо включить враздел операций АТД

1. Найти максимальное значение массива.
2. Вставить максимальное значение массива после элемента, у которого первая и последняя цифры равны.
3. Удалить элементы массива, цифы которых образуют последовательность чисел Фибоначчи, в которой первое и второе число равно 1.

## 2. АТД ЗАДАЧИ.

АТД Set
{
Данные.
n – количество элементов множества.
A – список значений элементов множества.
pos – позиция элемента.
Операции (объявления операций).
*Общие для всех вариантов.*
1) заполнение структуры данных значениями (с клавиатуры, применение датчика случайных чисел);
//Предусловие. Множество s размером n, где 0<n<N.
//Постусловие. Заполненное множество A. void fillRandom(DataStructure& data, int size)

2) вывод структуры в консоль;
//Предусловие. Множество s размером N.
//Постусловие. Выведенное в консоль множество A. void printData(const DataStructure& data)

3) вставить элемент в заданную позицию. //Предусловие. Множество s размером N. //Постусловие. Добавление в массив элемента.
void insertElement(DataStructure& data, int position, int value)

*Дополнительные операции.*
1) Найти максимальное значение массива
//Предусловие. Массив array размером N.
//Постусловие. Нахождение максимального значения в массиве array.

int findMax(const int array[], int size)
2) Вставить максимальное значение массива после элемента, у которого первая и последняя цифры равны
//Предусловие. Массив array размером N, содержащий целые числа
//Постусловие. Вставка максимального значения массива после элемента, у которого первая и последняя цифры равны.
void insertMaxAfterEqualFirstAndLast(int array[], int& size, int max) {

3) Удалить элементы массива, цифры которых образуют последовательность чисел Фибоначчи, в которой первое и второе число равно 1

```
//Предусловие. Массив array размером N, содержащий целые числа
//Постусловие.  Удаление элементов массива, цифры которых образуют последовательность чисел Фибоначчи, где первое и второе число равны 1.
void deleteElementsWithFibonacciSequence(int array[], int& size) {
```

# 3. РАЗРАБОТКА ПРОГРАММЫ.

## 3.1. Включить реализацию данных АТД (представить код структуры).

```
struct DataStructure {
    int elements[MAX_SIZE];
    int size;
};

void fillRandom(DataStructure& data, int size) {
    srand(time(nullptr));
    data.size = size;
    for (int i = 0; i < size; ++i) {
        data.elements[i] = rand() % 100;
    }
}

void printData(const DataStructure& data) {
    cout << "elements in the structure: ";
    for (int i = 0; i < data.size; ++i) {
        cout << data.elements[i] << " ";
    }
    cout << endl;
}

int findMax(const int array[], int size) {
    if (size <= 0) {
        return -1;
    }
    int max = array[0];
    for (int i = 1; i < size; ++i) {
        if (array[i] > max) {
            max = array[i];
        }
    }
    return max;
}

void insertMaxAfterEqualFirstAndLast(int array[], int& size, int max) {
    for (int i = 0; i < size - 1; ++i) {
        if (array[i] % 10 == array[i] / 10 % 10) {
            for (int j = size; j > i + 1; --j) {
                array[j] = array[j - 1];
            }
            array[i + 1] = max;
            ++size;
            break;
        }
    }
}

bool isFibonacci(int num) {
    int a = 1, b = 1;
    while (true) {
        int c = a + b;
        if (c == num) {
            return true;
        }
        else if (c > num) {
            return false;
        }
        a = b;
        b = c;
    }
}

void deleteElementsWithFibonacciSequence(int array[], int& size) {
```

```
    int i = 0;
    while (i < size) {
        if (isFibonacci(array[i])) {
            for (int j = i; j < size - 1; ++j) {
                array[j] = array[j + 1];
            }
            --size;
        }
        else {
            ++i;
        }
    }
}
```

3.2. Таблица тестов тестирования операций варианта.

| Название алгоритма операции | | |
|---|---|---|
| Номер теста | Входные данные | Эталон результата |
| 1 | 25 21 78 65 79 5 10 41 27 47 | 25 78 65 79 10 41 27 47 |
| 2 | 23 93 15 68 37 35 79 26 54 45 | 23 93 15 68 37 35 79 26 54 45 |

## 3.3. Код проекта:

```
#include <iostream>
#include <ctime>
using namespace std;

const int MAX_SIZE = 100;

struct DataStructure {
    int elements[MAX_SIZE];
    int size;
};

void fillRandom(DataStructure& data, int size) {
    srand(time(nullptr));
    data.size = size;
    for (int i = 0; i < size; ++i) {
        data.elements[i] = rand() % 100;
    }
}

void printData(const DataStructure& data) {
    cout << "elements in the structure: ";
    for (int i = 0; i < data.size; ++i) {
        cout << data.elements[i] << " ";
    }
    cout << endl;
}

int findMax(const int array[], int size) {
    if (size <= 0) {
        return -1;
    }
    int max = array[0];
    for (int i = 1; i < size; ++i) {
        if (array[i] > max) {
            max = array[i];
        }
```

```
    }
    return max;
}

void insertMaxAfterEqualFirstAndLast(int array[], int& size, int max) {
    for (int i = 0; i < size - 1; ++i) {
        if (array[i] % 10 == array[i] / 10 % 10) {
            for (int j = size; j > i + 1; --j) {
                array[j] = array[j - 1];
            }
            array[i + 1] = max;
            ++size;
            break;
        }
    }
}

bool isFibonacci(int num) {
    int a = 1, b = 1;
    while (true) {
        int c = a + b;
        if (c == num) {
            return true;
        }
        else if (c > num) {
            return false;
        }
        a = b;
        b = c;
    }
}

void deleteElementsWithFibonacciSequence(int array[], int& size) {
    int i = 0;
    while (i < size) {
        if (isFibonacci(array[i])) {
            for (int j = i; j < size - 1; ++j) {
```

```
            array[j] = array[j + 1];
        }
        --size;
      }
      else {
        ++i;
      }
    }
}

int main() {
    DataStructure data;
    fillRandom(data, 10);
    cout << "Original ";
    printData(data);
    int max = findMax(data.elements, data.size);
    cout << "maximum value: " << max << endl;
    insertMaxAfterEqualFirstAndLast(data.elements, data.size, max);
    cout << "array with inserted maximum value after element with equal first and last digits:" << endl;
    printData(data);
    deleteElementsWithFibonacciSequence(data.elements, data.size);
    cout << "array after deleting elements with Fibonacci sequence values:" << endl;
    printData(data);
    return 0;
}
```

## 4. СКРИНЫ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ.

```
elements in the structure: 23 93 15 68 37 35 79 26 54 45
maximum value: 93
array with inserted maximum value after element with equal first and last digits:
elements in the structure: 23 93 15 68 37 35 79 26 54 45
array after deleting elements with Fibonacci sequence values:
elements in the structure: 23 93 15 68 37 35 79 26 54 45
```

```
elements in the structure: 58 46 84 23 90 16 0 71 73 36
maximum value: 90
array with inserted maximum value after element with equal first and last digits:
elements in the structure: 58 46 84 23 90 16 0 90 71 73 36
array after deleting elements with Fibonacci sequence values:
elements in the structure: 58 46 84 23 90 16 0 90 71 73 36
```

## 5. ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ.

1.Скворцова Л.А. Структуры и алгоритмы обработки данных. Часть 1: линейные структуры данных в алгоритмах [Электронный ресурс]: Практикум / Скворцова Л.А., Гусев К.В., Филатов А.С. — М.: МИРЭА – Российский технологический университет, 2023. — 1 электрон. опт. диск (CD-ROM).

# 6. ВЫВОДЫ.

В ходе самостоятельной работы:

− освоены технологии разработки программ на основе абстрактного типа; − приобретены умения и навыки определения и реализации абстрактного типа данных задачи на наиболее эффективной структуре представления данных в программе;

− приобретены навыки по реализации алгоритмов операций над массивом через аппарат функций языка C++;

− приобретены навыки реализации операций модификации (вставка и удаление элементов) одномерного массива и других операций обработки массива как структуры данных.