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 ПРАКТИЧЕСКОЙ РАБОТЕ №3
по дисциплине «Структуры и алгоритмы обработки данных»

Тема «Абстрактный тип данных и его реализация на одномерном динамическом массиве и векторе»

Выполнил студент группы ИКБО-50-23 Русаков М.Ю.

Принял старший преподаватель Скворцова Л.А.

Москва 2024

Оглавление

1. Отчет по заданию 1

1.1. Условие задачи

Реализовать АТД задачи, разработанное в практической работе 1, используя для представления значений множества динамический массив. Выполнить реализацию АТД задачи на динамическом массиве. Для управления динамической памятью использовать функции файла заголовка stdlib.h: malloc, free, realloc.

1.2. АТД задачи

Сформировать новый массив из чисел исходного, которые делятся на каждую цифру числа.

1.3. Текст АТД с операциями варианта

АТД `myDynArray`
{
Данные (описание свойств структуры данных задачи):
n - количество элементов массива
m - количество элементов массива для дополнительного задания
arr - список элементов массива
Операции (объявления операций):

- Метод, осуществляющий вывод текущих значений множества

Предусловие: нет
Постусловие: выведенные через пробел элементы множества

Заголовок: `printElements()`

- Метод, осуществляющий заполнение массива вручную – с клавиатуры

Предусловие: нет
Постусловие: массив, заполненный значениями, введенными с клавиатуры
Заголовок: `fillManually()`

- Метод, осуществляющий заполнение массива случайными значениями

Предусловие: нет
Постусловие: массив, заполненный случайными значениями
Заголовок: `fillRandomly()`

- Метод, возвращающий индекс первого элемента, делящегося на каждую из своих цифр. В случае отсутствия такового элемента возвращается -1

Предусловие: нет
Постусловие: число – индекс первого элемента, нацело делящегося на каждую из своих цифр
Заголовок: `getIndex()`

- Метод осуществляющий вставку элемента newElem на позицию с индексом pos

Предусловие: pos – индекс элемента, на место которого требуется вставить новый элемент, newElem – значение нового элемента
Постусловие: массив arr длиной n+1 со вставленным элементом newElem на позиции pos
Заголовок: `insert(int pos, int newElem)`

- Метод, осуществляющий вставку нового элемента newElem после элемента с индексом getIndex(). Если getIndex() $= -1$, вставка производится в начало массива

Предусловие: newElem – значение нового элемента
Постусловие: массив arr длиной n+1 со вставленным элементом newElem на позиции getIndex() + 1
Заголовок: `getIndexInsert(int newElem)`

- Метод, осуществляющий удаление из массива всех элементов, нацело делящихся на 3

Предусловие: нет
Постусловие: измененный массив arr, содержащий элементы, не делящихся нацело на 3
Заголовок: `deleteMultiplesOfThree()`

Дополнительные операции:

- Метод, формирующий новый массив из чисел исходного, которые делятся на каждую цифру числа

Предусловие: нет
Постусловие: новый массив, содержащий только числа исходного массива, нацело делящиеся на каждую из своих цифр
Заголовок: `newArray()`
}

1.4. Разработка программы

1.4.1. Реализация данных АТД

1.4.1.1. Код файла myArray.h

```
#ifndef MYDYNARRAY__H
#define MYDYNARRAY__H

#include <iostream>

struct myDynArray {
    const static int N = 100;
    unsigned int n;
    int* arr;

    myDynArray(int len);

    ~myDynArray();

    void printElements();

    void fillRandomly();

    void fillManually();

    int getIndex();

    void insert(int pos, int newElem);

    void getIndexInsert(int newElem);

    void erase(int pos);

    void deleteMultiplesOfThree();

    void newArray();

    bool dividesByAllDigits(int num);
};

#endif
```

1.4.1.2. Код файла myArray.cpp

```
#include <iostream>
#include <cstdlib>
#include <ctime>
```

```cpp
#include <random>
#include "../headers/dynamic_array.h"

// Конструктор. Создание динамического массива длиной len
myDynArray::myDynArray(int len) {
    n = len;
    arr = (int*)malloc(len * sizeof(int));
}

myDynArray::~myDynArray() {
    free(arr);
}

// Вывод текущих элементов массив в консоль
void myDynArray::printElements() {
    for (size_t i{}; i < n; i++) {
        std::cout << arr[i] << ' ';
    }
    std::cout << '\n';
}

// Метод, осуществляющий заполнение массива случайными значениями в
диапазоне [0; 100]
void myDynArray::fillRandomly() {
    srand(time(NULL));

    for (size_t i{}; i < n; i++) {
        int num = rand() % 101;
        arr[i] = num;
    }
}

// Метод, осуществляющий заполнение массива вручную
void myDynArray::fillManually() {
    for (size_t i{}; i < n; i++) {
        std::cin >> arr[i];
    }
}

/* Метод, возвращающий индекс первого элемента, делящегося на каждую из
своих цифр.
В случае отсутствия такового элемента возвращается -1. */
int myDynArray::getIndex() {
    for (size_t i{}; i < n; i++) {
        int number = arr[i];
```

```
        while (number > 0) {
            if (number % 10 != 0) {
                if (arr[i] % (number % 10) != 0) {
                    break;
                }
            } else {
                break;
            }

            number /= 10;
        }

        if (number == 0) {
            return i;
        }
    }

    return -1;
}

/* Метод осуществляющий вставку элемента newElem на позицию с индексом
pos */
void myDynArray::insert(int pos, int newElem) {
    if (!(pos >= 0 && pos <= n)) {
        std::cout << "Невозможно вставить в массив: неверный индекс!
\n";
        return;
    }

    int* newArr = (int*)malloc((n + 1) * sizeof(int));

    for (int i = 0; i < pos; ++i) {
        newArr[i] = arr[i];
    }

    newArr[pos] = newElem;

    for (int i = pos; i < n; ++i) {
        newArr[i + 1] = arr[i];
    }

    free(arr);
    arr = newArr;

    ++n;
}
```

```
/* Метод, осуществляющий вставку нового элемента newElem после элемента
с индексом getIndex().
Если getIndex() = -1, вставка производится в начало массива */
void myDynArray::getIndexInsert(int newElem) {
    insert(getIndex() + 1, newElem);
}

// Метод, осуществляющий удаление элемента с индексом pos
void myDynArray::erase(int pos) {
    if (!(pos > -1 && pos < this->n)) {
        std::cout << "Невозможно удалить элемент: неверный индекс!\n";
        return;
    }

    for (size_t i = pos; i < this->n - 1; i++) {
        arr[i] = arr[i+1];
    }

    arr = (int*)realloc(arr, (n - 1) * sizeof(int));
    n--;
}

/* Метод, осуществляющий удаление из массива всех элементов, нацело
делящихся на 3
Error-handling функций malloc и realloc отсутствует! */
void myDynArray::deleteMultiplesOfThree() {
    int* buffArr = (int*)malloc(n * sizeof(int));
    int index{};

    for (size_t i{}; i < n; i++) {
        if (arr[i] % 3 != 0) {
            if (++index > n) {
                buffArr = (int*)realloc(buffArr, index * sizeof(int));
            }

            buffArr[index - 1] = arr[i];
        }
    }

    n = index;

    if (arr != nullptr) {
        free(arr);
    }
```

```
    arr = buffArr;
}

/* Метод, проверяющий, делится ли число на каждую из своих цифр.
Если делится, то возвращается true, иначе else */
bool myDynArray::dividesByAllDigits(int num) {
    if (num == 0) {
        return false;
    }

    int temp = num;

    while (temp > 0) {
        int digit = temp % 10;

        if (digit == 0 || num % digit != 0) {
            return false;
        }

        temp /= 10;
    }

    return true;
}

// Метод, формирующий новый динамический массив из элементов исходного,
нацело делящихся на каждую из своих цифр
void myDynArray::newArray() {
    int* newArray = nullptr;
    int size{};

    for (size_t i{}; i < n; i++) {
        if (dividesByAllDigits(arr[i])) {
            newArray = (int*)realloc(newArray, ++size * sizeof(int));
            newArray[size - 1] = arr[i];
        }
    }
    n = size;

    arr = newArray;
}
```

1.4.1.3. Код файла `main.cpp`

Комментарий: в данном файле показан пример работы программы для теста №1

```
#include "myDynArray.h"
#include <iostream>

int main() {
    myDynArray test(6);

    // {12, 43, 11, 99, 0, 7}
    test.fillManually();

    test.newArray();

    std::cout << "n = " << test.n << '\n';
    std::cout << "Новый массив, сформированный из чисел исходного,
нацело делящихся на все свои цифры:\n";
    if (test.arr != nullptr) {
        test.printElements();
    } else {
        std::cout << "Массив пуст!\n";
    }
}
```

1.4.2. Алгоритм дополнительной операции варианта

1. Вспомогательная функция, возвращающая `true` в случае, если переданное в нее в качестве параметра число `num` нацело делится на каждую из своих цифр, иначе `false`.

Предусловие: целочисленная переменная `num`
Постусловие: `true/false`
Заголовок: `dividesByAllDigits(int num)`

Таблица 1 – Алгоритм вспомогательной функции `dividesByAllDigits(int num)`

Номер	Инструкция
1	if num = 0 then
2	return false
3	endIf
4	temp ← num
5	while temp > 0 do
6	digit ← temp mod 10
7	if digit = 0 ∨ num mod digit ≠ 0
8	return false
9	endIf

Номер	Инструкция
10	temp ← temp mod 10
11	od
12	return true

2. Основной метод

Таблица 2 – Алгоритм метода `newArray()`

Номер	Инструкция
1	newArray ← []
2	size ← 0
3	for i ← 0 to $n-1$ do
4	if dividesByAllDigits(arr[i])
5	// TODO
6	newArray[size − 1] ← arr[i]
7	endIf
8	od
9	n ← size
10	arr ← newArray

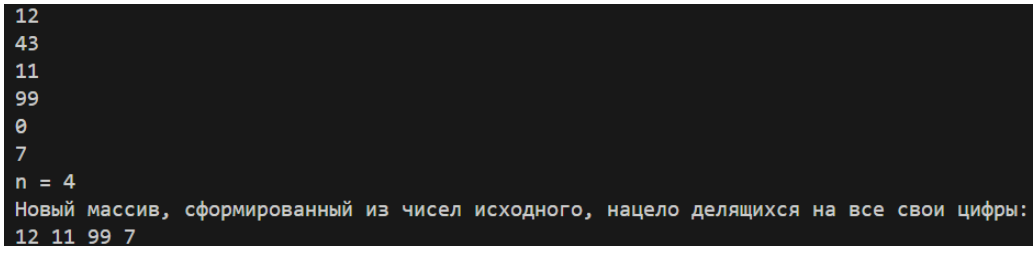
1.4.3. Таблицы тестов тестирования дополнительной операции варианта

Таблица 3 – Таблица тестов для тестирования дополнительной операции варианта

Номер	Входные данные	Результат работы
1	n = 6 arr = {12, 43, 11, 99, 0, 7}	n = 4 arr = {12, 11, 99, 7}
2	n = 4 arr = {13, 17, 23, 29}	n = 0 Ошибка: массив пуст!

1.4.4. Скрины результатов тестирования

Тест 1.



Тест 2.

```
13
17
23
29
n = 0
Новый массив, сформированный из чисел исходного, нацело делящихся на все свои цифры:
Массив пуст!
```

2. Отчет по заданию 2

2.1. Коды функций операций вставки, удаления, формирования нового множества заданий 1 и 2, представленные в таблице

Операция	Коды функций задания 1	Коды функций задания 2

Вставить элемент	`if (!(pos >= 0 && pos <= n)) {`
	`  std::cout << "Невозможно вставить в массив: неверный индекс!\n";`
	`  return;`
	`}`
	
	`int* newArr = (int*)malloc((n + 1) * sizeof(int));`
	
	`for (int i = 0; i < pos; ++i) {`
	`  newArr[i] = arr[i];`
	`}`
	
	`newArr[pos] = newElem;`
	
	`for (int i = pos; i < n; ++i) {`
	`  newArr[i + 1] = arr[i];`
	`}`
	
	`free(arr);`
	`arr = newArr;`
	
	`++n;`
		`if (!(pos > -1 && pos < this->n)) {`
		`  std::cout << "Невозможно вставить в массив: неверный индекс!\n";`
		`  return;`
		`}`
		
		`arr.insert(arr.begin() + pos, newElem);`

Удалить элемент	`if (!(pos > -1 && pos < this->n)) {`
	`  std::cout << "Невозможно удалить элемент: неверный индекс!\n";`
	`  return;`
	`}`
	
	`for (size_t i = pos; i < this->n - 1; i++) {`
	`  arr[i] = arr[i+1];`
	`}`
	
	`arr = (int*)realloc(arr, (n - 1) * sizeof(int));`
	`n--;`
		`if (!(pos > -1 && pos < this->n)) {`
		`  std::cout << "Невозможно удалить элемент: неверный индекс!\n";`
		`  return;`
		`}`
		
		`arr.erase(arr.begin() + pos);`

Операция	Коды функций задания 1	Коды функций задания 2

Формирование нового множества	`int* newArray = nullptr;`
	`int size{};`
	
	`for (size_t i{}; i < n; i++) {`
	`    if (dividesByAllDigits(arr[i])) {`
	`        newArray = (int*)realloc(newArray, ++size * sizeof(int));`
	`        newArray[size - 1] = arr[i];`
	`    }`
	`}`
	`n = size;`
	
	`arr = newArray;`
		`std::vector<int> newVector;`
		`for (int& num: arr) {`
		`    if (dividesByAllDigits(num)) {`
		
		`newVector.push_back(num);`
		`    }`
		`}`
		
		`arr = newVector;`

2.2. Код проекта

2.2.1. Код файла `myArray.h`

```
#ifndef MYARRAY__H
#define MYARRAY__H

#include <iostream>
#include <vector>

struct myVector {
    const static int N = 100;
    unsigned int n;
    std::vector <int> arr;

    myVector(int len);

    void printElements();

    void fillRandomly();

    void fillManually();

    int getIndex();

    void insert(int pos, int newElem);

    void getIndexInsert(int newElem);

    void erase(int pos);
```

```
    void deleteMultiplesOfThree();

    bool dividesByAllDigits(int num);

    void newVector();
};

#endif
```

2.2.2. Код файла myArray.cpp

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <random>
#include "myVector.h"

// Конструктор. Инициализация переменной n (длина массива)
myVector::myVector(int len) {
    n = len;
    arr.assign(n, 0);
}

// Вывод текущих элементов массив в консоль
void myVector::printElements() {
    for (int& num: arr) {
        std::cout << num << ' ';
    }

    std::cout << '\n';
}

// Метод, осуществляющий заполнение массива случайными значениями из
диапазона [0; 100]
void myVector::fillRandomly() {
    srand(time(NULL));

    for (int& num: arr) {
        num = rand() % 101;
    }
}

// Метод, осуществляющий заполнение массива вручную
void myVector::fillManually() {
    for (int& num: arr) {
        std::cin >> num;
    }
```

```
}

/* Метод, возвращающий индекс первого элемента, делящегося на каждую из
своих цифр.
В случае отсутствия такового элемента возвращается -1. */
int myVector::getIndex() {
    for (size_t i{}; i < n; i++) {
        int number = arr[i];

        while (number > 0) {
            if (number % 10 != 0) {
                if (arr[i] % (number % 10) != 0) {
                    break;
                }
            } else {
                break;
            }

            number /= 10;
        }

        if (number == 0) {
            return i;
        }
    }

    return -1;
}

/* Метод осуществляющий вставку элемента newElem на позицию с индексом
pos */
void myVector::insert(int pos, int newElem) {
    if (!(pos > -1 && pos < this->n)) {
        std::cout << "Невозможно вставить в массив: неверный индекс!
\n";
        return;
    }

    arr.insert(arr.begin() + pos, newElem);
}

/* Метод, осуществляющий вставку нового элемента newElem после элемента
с индексом getIndex().
Если getIndex() = -1, вставка производится в начало массива */
void myVector::getIndexInsert(int newElem) {
    arr.insert(arr.begin() + getIndex() + 1, newElem);
```

```
}

/* Метод, осуществляющий удаление элемента с индексом pos */
void myVector::erase(int pos) {
    if (!(pos > -1 && pos < this->n)) {
        std::cout << "Невозможно удалить элемент: неверный индекс!\n";
        return;
    }

    arr.erase(arr.begin() + pos);
}

/* Метод, осуществляющий удаление из массива всех элементов, нацело
делящихся на 3*/
void myVector::deleteMultiplesOfThree() {
    for (size_t i{}; i < arr.size(); i++) {
        if (arr[i] % 3 != 0) {
            arr.erase(arr.begin() + i);
        }
    }
}

bool myVector::dividesByAllDigits(int num) {
    if (num == 0) {
        return false;
    }

    int temp = num;

    while (temp > 0) {
        int digit = temp % 10;

        if (digit == 0 || num % digit != 0) {
            return false;
        }

        temp /= 10;
    }

    return true;
}

void myVector::newVector() {
    std::vector<int> newVector;
    for (int& num: arr) {
        if (dividesByAllDigits(num)) {
```

```
            newVector.push_back(num);
        }
    }

    arr = newVector;
}
```

2.2.3. Код файла `main.cpp`

Комментарий: в данном файле показан пример работы программы для теста №1

```
#include "myVector.h"
#include <vector>

int main() {
    myVector arr(6);

    // {12, 43, 11, 99, 0, 7}
    arr.fillManually();

    arr.newVector();

    std::cout << "n = " << arr.n << '\n';
    std::cout << "Новый массив, сформированный из чисел исходного,
нацело делящихся на все свои цифры: \n";
    arr.printElements();
}
```

2.3. Скриншоты результатов тестирования

Тест 1.

```
12
43
11
99
0
7
n = 6
Новый массив, сформированный из чисел исходного, нацело делящихся на все свои цифры:
12 11 99 7
```

Тест 2.

```
13
17
23
29
n = 4
Новый массив, сформированный из чисел исходного, нацело делящихся на все свои цифры:
```