

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 №2**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Абстрактный тип данных задачи и его реализация на одномерном динамическом массиве и векторе

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА 3](#_Toc129122088)

[2 АБСТРАКТНЫЙ ТИП ДАННЫХ ДЛЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 4](#_Toc129122089)

[3 РАЗРАБОТКА ПРОГРАММЫ НА ДИНАМИЧЕСКОМ МАССИВЕ 7](#_Toc129122090)

[3.1 РЕАЛИЗАЦИЯ ДАННЫХ АТД 7](#_Toc129122091)

[3.2 АЛГОРИТМЫ РЕАЛИЗОВАННЫХ ОПЕРАЦИЙ 7](#_Toc129122092)

[3.2.1 Проверка элемента на чётность длины 7](#_Toc129122093)

[3.2.2 Проверка элемента на палиндром 7](#_Toc129122094)

[3.2.3 Создать новый массив из чисел исходного, которые являются палиндромами чётной длины 8](#_Toc129122095)

[3.3 ТАБЛИЦЫ ТЕСТОВ ДЛЯ ОПЕРАЦИЙ ВАРИАНТА 8](#_Toc129122096)

[3.3.1 Тест создания нового массива из чисел исходного, которые являются палиндромами чётной длины 8](#_Toc129122097)

[3.4 КОД ПРОЕКТА 9](#_Toc129122098)

[3.4.1 Код файла Dynamic\_array.h 9](#_Toc129122099)

[3.4.2 Код файла Dynamic\_array.cpp 9](#_Toc129122100)

[3.4.3 Код файла siaod\_programme2.cpp 13](#_Toc129122101)

[3.5 СКРИНЫ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ 18](#_Toc129122102)

[4 РАЗРАБОТКА ПРОГРАММЫ НА ВЕКТОРЕ 20](#_Toc129122103)

[4.1 РЕАЛИЗАЦИЯ ДАННЫХ АТД 20](#_Toc129122104)

[4.2 ТАБЛИЦЫ ТЕСТОВ ДЛЯ ОПЕРАЦИЙ ВАРИАНТА 20](#_Toc129122105)

[4.2.1 Тест создания нового массива из чисел исходного, которые являются палиндромами чётной длины 20](#_Toc129122106)

[4.3 КОД ПРОГРАММЫ 20](#_Toc129122107)

[4.3.1 Код файла Vector\_array.h 20](#_Toc129122108)

[4.3.2 Код файла Vector\_array.cpp 21](#_Toc129122109)

[4.3.3 Код файла siaod2\_vector.cpp 24](#_Toc129122110)

[4.4 СКРИНЫ РЕЗУЛЬТАТА ТЕСТИРОВАНИЯ 29](#_Toc129122111)

[5 ВЫВОДЫ 31](#_Toc129122112)

[6 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 32](#_Toc129122113)

# 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.

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 в раздел операций АТД.

Операции над множеством, которые требуется реализовать:

1. Заполнение структуры данных значениями (с клавиатуры, применение датчика случайных чисел);
2. Вывод структуры в консоль;
3. Удалить элемент в заданной позиции;
4. Вставить элемент в заданную позицию;
5. Найти позицию элемента массива (первое вхождение), которое является палиндромом;
6. Удалить элементы массива, расположенные непосредственно перед элементом, содержащим число палиндром;
7. Вставить новый элемент в массив после элемента массива, который является палиндромом;
8. Сформировать новый массив из чисел исходного, которые являются

палиндромами четной длины;

1. Добавить новый элемент в массив.

# 2 АБСТРАКТНЫЙ ТИП ДАННЫХ ДЛЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

АТД TypeX

{

Данные

n – количество элементов множества

arr – список значений элементов множества, реализованный на динамическом массиве

Pos – позиция элемента, который является палиндромом

TypeX – тип элементов множества

typeitem – тип элементов множества

Операции

1. Заполнение структуры данных значениями с клавиатуры

Предусловие. Ссылка на множество типа TypeX

Постусловие. Заполненный массив data из n элементов. Нет возвращаемого значения.

**input\_data(TypeX& data);**

1. Заполнение структуры данных значениями – применение датчика случайных чисел

Предусловие. Ссылка на множество типа TypeX

Постусловие. Заполненный массив data из n случайных элементов. Нет возвращаемого значения.

**random\_input\_data(TypeX& data);**

1. Вывод структуры в консоль

Предусловие. Ссылка на множество типа TypeX

Постусловие. Вывод значений из массива. Нет возвращаемого значения.

**output\_data(TypeX& data);**

1. Удалить элемент в заданной позиции

Предусловие. Ссылка на множество типа TypeX. Передача значения переменной pos в функцию.

Постусловие. Массив data с (n-1) элементами и удалённым элементов в позиции pos.Возвращает 0, если элемент удалён, и 1, если удаление невозможно.

**deleteElem(TypeX& data, int pos);**

1. Вставить элемент в заданную позицию

Предусловие. Ссылка на множество типа TypeX. Передача вставляемого элемента типа typeitem по значению. Передача позиции вставляемого элемента по значению.

Постусловие. Массив data с (n + 1) элементом и вставленным элементом n в pos позицию. Код возвращаемого значения: 0, если удалось вставить новый элемент, 1 – если вставка не прошла.

**insertElem(TypeX& data, typeitem n, int pos);**

1. Найти позицию элемента массива (первое вхождение), которое является палиндромом

Предусловие. Ссылка на множество типа TypeX.

Постусловие. Вывод на экран позиции элемента – палиндрома. Код возвращаемого значения: 0 – если позиция найдена успешна, 1 – такого элемента не найдено.

**special\_findElem(TypeX& data);**

1. Проверка, является ли данный элемент палиндромом

Предусловие. n – элемент типа typeitem.

Постусловие. Возвращает логическую единицу или ноль, в зависимости от того, является ли элемент палиндромом.

**check\_palindrom(typeitem n);**

1. Удалить элементы массива, расположенные непосредственно перед элементом, содержащим число палиндром

Предусловие. Ссылка на множество типа TypeX.

Постусловие. Массив data с некоторым количеством элементов и удалёнными элементами, стоящими до Pos позиции. Возвращает 0, если получилось удалить элемент, и единицу, если удаление не получилось.

**special\_deleteElem(TypeX& data);**

1. Вставить новый элемент в массив после элемента массива, который является палиндромом

Предусловие. Ссылка на множество типа TypeX, элемент n типа typeitem.

Постусловие. Массив data с (n+1) элементов и вставленным элементом n типа typeitem, если найдена позиция элемента – палиндрома, в этом случае код работы программы - 0, в противном случае – возвращает 1.

**special\_insertElem(TypeX& data, typeitem n);**

1. Добавление нового элемента в массив

Предусловие. Ссылка на множество типа TypeX, новый элемент n типа typeitem.

Постусловие. Массив data с (n+1) элементами и новым добавленным элементом x в конец массива. Возвращаемое значение: 0 – при успешном добавлении элемента, 1 – добавить новый элемент не удалось.

**append\_element(TypeX& data, typeitem n);**

1. Проверка, является ли данный элемент чётной длины

Предусловие. Элемент x типа typeitem.

Постусловие. Возвращает логическую единицу или ноль, в зависимости от того, является ли элемент элементом чётной длины.

**check\_element\_len(typeitem x);**

1. Генерация нового массива из чисел исходного, которые являются палиндромами чётной длины

Предусловие. Ссылка на множество типа TypeX, ссылка на новое множество типа TypeX.

Постусловие. Возвращает 0, если удалось сгенерировать новый массив согласно логике, установленной в задаче, в противном случае возвращает 1.

**generate\_new\_array(TypeX data, TypeX& new\_data);**

}

# 3 РАЗРАБОТКА ПРОГРАММЫ НА ДИНАМИЧЕСКОМ МАССИВЕ

## 3.1 РЕАЛИЗАЦИЯ ДАННЫХ АТД

typedef int typeitem; //тип элементов множества

struct TypeX

{

unsigned int n = 0; //n - количество элементов множества

int Pos = -1;

typeitem \*X = {};

TypeX(int n1);

TypeX();

};

## 3.2 АЛГОРИТМЫ РЕАЛИЗОВАННЫХ ОПЕРАЦИЙ

## 3.2.1 Проверка элемента на чётность длины

check\_element\_len(x)

{

count ← 0

While (x>0) do

x←x/10

count← count+1

od

return count % 2 == 0

}

## 3.2.2 Проверка элемента на палиндром

check\_palindrom(n)

{

copy\_n ← n

last\_digit ← 0

res ← 0

While (n > 0) do

last\_digit ← n%10

res ← (res \* 10) + last\_digit

n ← n/10

od

return res == copy\_n

}

## 3.2.3 Создать новый массив из чисел исходного, которые являются палиндромами чётной длины

generate\_new\_array(data, new\_data)

{

new\_data.n ← 0

For i ← 0 to data.n do

If check\_element\_len (data.X[i]) and check\_palindrom(data.X[i]) then

new\_data.X[new\_data.n] ← data.X[i]

new\_data.n ← new\_data.n + 1

endIf

od

}

## 3.3 ТАБЛИЦЫ ТЕСТОВ ДЛЯ ОПЕРАЦИЙ ВАРИАНТА

## 3.3.1 Тест создания нового массива из чисел исходного, которые являются палиндромами чётной длины

|  |  |  |
| --- | --- | --- |
| Сформировать новый массив из чисел исходного, которые являются  палиндромами четной длины | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | data.n = 5  data.X = {1, 44, 766, 45, 444} | new\_data.n = 1  new\_data = {44} |
| 2 | data.n = 3  data.X = {22, 2332, 17771} | new\_data.n = 2  new\_data = {22, 2332} |
| 3 | data.n = 4  data.X = {1,2,3,4} | new\_data.n = 0  Массив не заполнен |
| 4 | data.n = 4  data.X = {55, 5555, 7667, 1001} | N = 100, n =4  new\_data.n = 4  new\_data = {55, 5555, 7667, 1001} |

## 3.4 КОД ПРОЕКТА

## 3.4.1 Код файла Dynamic\_array.h

#pragma once

#include "malloc.h"

using namespace std;

typedef int typeitem; //тип элементов множества

struct TypeX

{

unsigned int n = 0; //n - количество элементов множества

int Pos = -1;

typeitem \*X = {};

TypeX(int n1);

TypeX();

};

void input\_array(TypeX& data);

void random\_input\_array(TypeX& data);

void output\_array(TypeX& data);

int delete\_element(TypeX& data, int pos);

int insert\_elemet(TypeX& data, typeitem x, int pos);

bool check\_palindrom(typeitem x);

int find\_palindrom(TypeX& data);

int delete\_before\_palindrom(TypeX& data);

int insert\_nextto\_palindrom(TypeX& data, typeitem x);

int append\_element(TypeX& data, typeitem x);

bool check\_element\_len(typeitem x);

int generate\_new\_array(TypeX data, TypeX& new\_data);

## 3.4.2 Код файла Dynamic\_array.cpp

#include "Dynamic\_array.h"

#include <iostream>

#include <ctime>

TypeX::TypeX(int n1)

{

n = n1;

X = (typeitem\*)malloc(n \* sizeof(int));

}

TypeX::TypeX()

{

n = 0;

}

void input\_array(TypeX& data)

{

for (int i = 0; i < data.n; i++)

{

cin >> data.X[i];

}

}

void random\_input\_array(TypeX& data)

{

srand(time(NULL));

for (int i = 0; i < data.n; i++)

{

data.X[i] = rand() % 100;

}

}

void output\_array(TypeX& data)

{

for (int i = 0; i < data.n; i++)

{

cout << data.X[i] << " ";

}

cout << endl;

}

int delete\_element(TypeX& data, int pos)

{

if (pos >= 0 and pos < data.n)

{

for (int i = pos+1; i < data.n ; i++)

{

data.X[i - 1] = data.X[i];

}

data.n--;

return ((data.X = (typeitem\*)realloc(data.X, (data.n ) \* sizeof(int))) == NULL);

}

else

{

return 1;

}

}

int insert\_elemet(TypeX& data, typeitem x, int pos)

{

if (pos >= 0 and pos <= data.n and (data.X = (typeitem\*)realloc(data.X, (data.n + 1) \* sizeof(int))) != NULL)

{

for (int i = data.n; i > pos; i--)

{

data.X[i] = data.X[i - 1];

}

data.X[pos] = x;

data.n++;

return 0;

}

else

{

return 1;

}

}

bool check\_palindrom(typeitem n)

{

int copy\_n = n, last\_digit, res = 0;

while (n > 0)

{

last\_digit = n % 10;

res = (res \* 10) + last\_digit;

n /= 10;

}

return res == copy\_n;

}

int find\_palindrom(TypeX& data)

{

for (int i = 0; i < data.n; i++)

{

if (check\_palindrom(data.X[i]))

{

data.Pos = i;

return 0;

}

}

return 1;

}

int delete\_before\_palindrom(TypeX& data)

{

while (data.Pos != 0) { delete\_element(data, --data.Pos); }

return 0;

}

int insert\_nextto\_palindrom(TypeX& data, typeitem n)

{

return insert\_elemet(data, n, data.Pos + 1);

}

int append\_element(TypeX& data, typeitem x)

{

if ((data.X = (typeitem\*)realloc(data.X, (data.n + 1) \* sizeof(int))) != NULL)

{

data.X[data.n] = x;

data.n++;

return 0;

}

return 1;

}

bool check\_element\_len(typeitem x)

{

int count = 0;

while (x > 0)

{

x=x/10;

count++;

}

return count % 2 == 0;

}

int generate\_new\_array(TypeX data, TypeX& new\_data) //Генерация нового массива не подразумевает выполнения действий над ним, следующее действие будет выполнено для исходного массива!

{

new\_data.n = 0;

new\_data.X = (typeitem\*)malloc(0);

for (int i = 0; i < data.n; i++)

{

if ((check\_element\_len(data.X[i]) == 1) and (check\_palindrom(data.X[i]) == 1))

{

//расширение исходного массива и добавление нового элемента

if ((new\_data.X = (typeitem\*)realloc(new\_data.X, (new\_data.n + 1) \* sizeof(typeitem))) != NULL)

{

new\_data.X[new\_data.n] = data.X[i];

new\_data.n++;

}

else

{

return 1;

}

}

}

return 0;

}

## 3.4.3 Код файла siaod\_programme2.cpp

#include "Dynamic\_array.h"

#include <ctime>

#include <stdlib.h>

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251");

int n, pos, num;

typeitem x;

TypeX arr, new\_arr;

while (true)

{

cout << "-----------------------------------------------------------------------------------------------------------" << endl;

cout << " Меню\n";

cout << " 1. Заполнить множество с клавиатуры\n";

cout << " 2. Заполнить множество случайными числами\n";

cout << " 3. Вывести множество\n";

cout << " 4. Удалить элемент в позиции\n";

cout << " 5. Вставить элемент в позицию\n";

cout << " 6. Найти позицию элемента, который является палиндромом\n";

cout << " 7. Удалить элементы массива, расположенные непосредственно перед элементом, содержащим число палиндром\n";

cout << " 8. Вставить новый элемент в массив после элемента массива, который является палиндромом\n";

cout << " 9. Добавить новый элемент в массив\n";

cout << " 10. Сгенерировать новый массив из чисел старого, которые являются палиндромами чётной длины\n";

cout << " 11. Завершить работу\n";

cin >> num;

switch (num)

{

case 1:

cout << " Введите размер множества: ";

cin >> n;

arr = TypeX(n);

if (arr.n == 0 or arr.X==NULL)

{

cout << " Неккоректный размер множества" << endl;

}

else

{

cout << " Введите " << arr.n << " чисел: ";

input\_array(arr);

}

break;

case 2:

cout << " Введите размер множества: ";

cin >> n;

arr = TypeX(n);

if (arr.n == 0 or arr.X==NULL)

{

cout << " Неккоректный размер множества" << endl;

}

else

{

random\_input\_array(arr);

}

break;

case 3:

if (arr.n == 0 or arr.X == NULL)

{

cout << " пустое множество" << endl;

}

else

{

cout << "Data is " << arr.n << " numbers: ";

output\_array(arr);

}

break;

case 4:

if (arr.n == 0 or arr.X == NULL)

{

cout << " пустое множество" << endl;

}

else

{

cout << " Введите позицию удаляемого элемента: " << endl;

cin >> pos;

if (delete\_element(arr, pos) == 1)

{

cout << "Невозможно удалить элемент" << endl;

}

else

{

output\_array(arr);

}

}

break;

case 5:

if (arr.n == 0 or arr.X == NULL)

{

cout << " пустое множество" << endl;

}

else

{

cout << " введите элемент для вставки: " << endl;

cin >> x;

cout << " Введите позицию вставляемого элемента: " << endl;

cin >> pos;

if (insert\_elemet(arr, x, pos) == 1)

{

cout << "Невозможно вставить элемент" << endl;

}

else

{

output\_array(arr);

}

}

break;

case 6:

if (arr.n == 0 or arr.X == NULL)

{

cout << " пустое множество" << endl;

}

else

{

if (find\_palindrom(arr) != 1)

{

cout << " Позиция элемента палиндрома: " << arr.Pos << endl;

}

else

{

cout << " Элемента палиндрома не найдено!" << endl;

}

}

break;

case 7:

if (arr.n == 0 or arr.X == NULL)

{

cout << " пустое множество" << endl;

}

else if ((find\_palindrom(arr)) == 1)

{

cout << " элемента - палиндрома не найдено! " << endl;

}

else if ((delete\_before\_palindrom(arr))==1)

{

cout << "Удалить элементы невозможно!" << endl;

}

else

{

output\_array(arr);

}

break;

case 8:

if (arr.n == 0 or arr.X == NULL)

{

cout << " пустое множество" << endl;

}

else if (find\_palindrom(arr) == 1)

{

cout << " элемента - палиндрома не найдено! " << endl;

}

else

{

cout << " Введите элемент для вставки: " << endl;

cin >> x;

if (insert\_nextto\_palindrom(arr, x) == 1)

{

cout << "Невозможно вставить элемент " << endl;

}

else

{

output\_array(arr);

}

}

break;

case 9:

if (arr.n == 0 or arr.X == NULL)

{

cout << " Пустое множество" << endl;

}

else

{

cout << "Введите вставляемый элемент: ";

cin >> x;

if (append\_element(arr, x) == 1)

{

cout << "Невозможно вставить элемент" << endl;

}

else output\_array(arr);

}

break;

case 10:

new\_arr = TypeX(0);

if (generate\_new\_array(arr, new\_arr) == 1)

{

cout << "Невозможно создать новый массив" << endl;

}

else if (new\_arr.n == 0) cout << "Массив не заполнен" << endl;

else output\_array(new\_arr);

free(new\_arr.X);

break;

case 11:

free(arr.X);

return 0;

default: cout << " Нет такого пункта" << endl;

}

}

return 0;

}

## 3.5 СКРИНЫ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Результат теста 1.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат теста 1.2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 3 – Результат теста 1.3

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 4 – Результат теста 1.4

# 4 РАЗРАБОТКА ПРОГРАММЫ НА ВЕКТОРЕ

## 4.1 РЕАЛИЗАЦИЯ ДАННЫХ АТД

typedef int typeitem;

struct TypeX

{

unsigned int n = 0;

int Pos = -1;

vector<typeitem> X;

TypeX();

TypeX(int n1);

};

## 4.2 ТАБЛИЦЫ ТЕСТОВ ДЛЯ ОПЕРАЦИЙ ВАРИАНТА

## 4.2.1 Тест создания нового массива из чисел исходного, которые являются палиндромами чётной длины

|  |  |  |
| --- | --- | --- |
| Сформировать новый массив из чисел исходного, которые являются  палиндромами четной длины | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | data.X.size() = 6  data.X = {3, 78, 232, 2442, 1, 7} | new\_data.X.size() = 1  new\_data.X = {2442} |
| 2 | data.X.size() = 3  data.X = {33, 2222, 6116} | new\_data.X.size() = 3  new\_data.X = {33, 2222, 6116} |
| 3 | data.X.size() = 3  data.X = {373, 21321, 2323} | new\_data.X.size() = 0  Массив не заполнен |
| 4 | data.X.size() = 4  data.X = {12, 22, 45, 9009} | new\_data.X.size() = 2  new\_data.X = {22, 9009} |

## 4.3 КОД ПРОГРАММЫ

## 4.3.1 Код файла Vector\_array.h

#pragma once

#include <ctime>

#include <vector>

#include <string.h>

#include <iostream>

using namespace std;

typedef int typeitem;

struct TypeX

{

unsigned int n = 0;

int Pos = -1;

vector<typeitem> X;

TypeX();

TypeX(int n1);

};

void input\_array(TypeX& data);

void random\_input\_array(TypeX& data);

void output\_array(TypeX& data);

int delete\_element(TypeX& data, int pos);

int insert\_elemet(TypeX& data, typeitem x, int pos);

bool check\_palindrom(typeitem x);

int find\_palindrom(TypeX& data);

int delete\_before\_palindrom(TypeX& data);

int insert\_nextto\_palindrom(TypeX& data, typeitem x);

int append\_element(TypeX& data, typeitem x);

bool check\_element\_len(typeitem x);

int generate\_new\_array(TypeX data, TypeX& new\_data);

## 4.3.2 Код файла Vector\_array.cpp

#include "Vector\_array.h"

TypeX :: TypeX()

{

n = 0;

}

TypeX::TypeX(int n1)

{

n = n1;

X.resize(n);

}

void input\_array(TypeX& data)

{

for (int i = 0; i < data.X.size(); i++)

{

cin >> data.X[i];

}

}

void random\_input\_array(TypeX& data)

{

srand(time(NULL));

for (int i = 0; i < data.X.size(); i++)

{

data.X[i] = rand() % 100;

}

}

void output\_array(TypeX& data)

{

for (int i = 0; i < data.X.size(); i++)

{

cout << data.X[i] << " ";

}

cout << endl;

}

int delete\_element(TypeX& data, int pos)

{

vector <typeitem> new\_data = {};

if (pos >= 0 and pos < data.X.size())

{

data.X.erase(data.X.begin() + pos);

return 0;

}

return 1;

}

int insert\_elemet(TypeX& data, typeitem x, int pos)

{

if (pos >= 0 and pos <= data.X.size())

{

data.X.insert(data.X.begin() + pos, x);

return 0;

}

return 1;

}

bool check\_palindrom(typeitem x)

{

int copy\_n = x, last\_digit, res = 0;

while (x > 0)

{

last\_digit = x % 10;

res = (res \* 10) + last\_digit;

x /= 10;

}

return res == copy\_n;

}

int find\_palindrom(TypeX& data)

{

for (int i = 0; i < data.X.size(); i++)

{

if (check\_palindrom(data.X[i]))

{

data.Pos = i;

return 0;

}

}

return 1;

}

int delete\_before\_palindrom(TypeX& data)

{

if (data.Pos > 0 and data.X.size() > 0)

{

while (data.Pos != 0)

{

delete\_element(data, --data.Pos);

}

return 0;

}

return 1;

}

int insert\_nextto\_palindrom(TypeX& data, typeitem x)

{

return insert\_elemet(data, x, data.Pos+1);

}

int append\_element(TypeX& data, typeitem x)

{

data.X.push\_back(x);

return 0;

}

bool check\_element\_len(typeitem x)

{

int count = 0;

while (x > 0)

{

x = x / 10;

count++;

}

return count % 2 == 0;

}

int generate\_new\_array(TypeX data,TypeX &new\_data)

{

for (int i = 0; i < data.X.size(); i++)

{

if (check\_element\_len(data.X[i]) == 1 and check\_palindrom(data.X[i]) == 1)

{

new\_data.X.push\_back(data.X[i]);

}

}

return 0;

}

## 4.3.3 Код файла siaod2\_vector.cpp

#include "Vector\_array.h"

int main()

{

system("chcp 1251");

int n, pos, num;

typeitem x;

TypeX arr, new\_arr;

while (true)

{

cout << "-----------------------------------------------------------------------------------------------------------" << endl;

cout << " Меню\n";

cout << " 1. Заполнить множество с клавиатуры\n";

cout << " 2. Заполнить множество случайными числами\n";

cout << " 3. Вывести множество\n";

cout << " 4. Удалить элемент в позиции\n";

cout << " 5. Вставить элемент в позицию\n";

cout << " 6. Найти позицию элемента, который является палиндромом\n";

cout << " 7. Удалить элементы массива, расположенные непосредственно перед элементом, содержащим число палиндром\n";

cout << " 8. Вставить новый элемент в массив после элемента массива, который является палиндромом\n";

cout << " 9. Добавить новый элемент в массив\n";

cout << " 10. Сгенерировать новый массив из чисел старого, которые являются палиндромами чётной длины\n";

cout << " 11. Завершить работу\n";

cin >> num;

switch (num)

{

case 1:

cout << " Введите размер множества: ";

cin >> n;

arr = TypeX(n);

if (arr.X.size() == 0)

{

cout << " Неккоректный размер множества" << endl;

}

else

{

cout << " Введите " << arr.X.size() << " чисел: ";

input\_array(arr);

}

break;

case 2:

cout << " Введите размер множества: ";

cin >> n;

arr = TypeX(n);

if (arr.X.size() == 0)

{

cout << " Неккоректный размер множества" << endl;

}

else

{

random\_input\_array(arr);

}

break;

case 3:

if (arr.X.size() == 0)

{

cout << " пустое множество" << endl;

}

else

{

cout << "Data is " << arr.n << " numbers: ";

output\_array(arr);

}

break;

case 4:

if (arr.X.size() == 0)

{

cout << " пустое множество" << endl;

}

else

{

cout << " Введите позицию удаляемого элемента: " << endl;

cin >> pos;

if (delete\_element(arr, pos) == 1)

{

cout << "Невозможно удалить элемент" << endl;

}

else

{

output\_array(arr);

}

}

break;

case 5:

if (arr.X.size() == 0)

{

cout << " пустое множество" << endl;

}

else

{

cout << " введите элемент для вставки: " << endl;

cin >> x;

cout << " Введите позицию вставляемого элемента: " << endl;

cin >> pos;

if (insert\_elemet(arr, x, pos) == 1)

{

cout << "Невозможно вставить элемент" << endl;

}

else

{

output\_array(arr);

}

}

break;

case 6:

if (arr.X.size() == 0)

{

cout << " пустое множество" << endl;

}

else

{

if (find\_palindrom(arr) != 1)

{

cout << " Позиция элемента палиндрома: " << arr.Pos << endl;

}

else

{

cout << " Элемента палиндрома не найдено!" << endl;

}

}

break;

case 7:

if (arr.X.size() == 0)

{

cout << " пустое множество" << endl;

}

else if ((find\_palindrom(arr)) == 1)

{

cout << " элемента - палиндрома не найдено! " << endl;

}

else if ((delete\_before\_palindrom(arr)) == 1)

{

cout << "Удалить элементы невозможно!" << endl;

}

else

{

output\_array(arr);

}

break;

case 8:

if (arr.X.size() == 0)

{

cout << " пустое множество" << endl;

}

else if (find\_palindrom(arr) == 1)

{

cout << " элемента - палиндрома не найдено! " << endl;

}

else

{

cout << " Введите элемент для вставки: " << endl;

cin >> x;

if (insert\_nextto\_palindrom(arr, x) == 1)

{

cout << "Невозможно вставить элемент " << endl;

}

else

{

output\_array(arr);

}

}

break;

case 9:

if (arr.X.size() == 0)

{

cout << " Пустое множество" << endl;

}

else

{

cout << "Введите вставляемый элемент: ";

cin >> x;

if (append\_element(arr, x) == 1)

{

cout << "Невозможно вставить элемент" << endl;

}

else output\_array(arr);

}

break;

case 10:

new\_arr = TypeX(0);

if (generate\_new\_array(arr, new\_arr) == 1)

{

cout << "Невозможно создать новый массив" << endl;

}

else if (new\_arr.X.size() == 0) cout << "Массив не заполнен" << endl;

else output\_array(new\_arr);

new\_arr.X.clear();

break;

case 11:

arr.X.clear();

return 0;

default: cout << " Нет такого пункта" << endl;

}

}

return 0;

}

## 4.4 СКРИНЫ РЕЗУЛЬТАТА ТЕСТИРОВАНИЯ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Результат теста 2.1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Результат теста 2.2

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 7 – Результат теста 2.3

Изображение выглядит как текст

Автоматически созданное описаниеРисунок 8 – Результат теста 2.4

# 5 ВЫВОДЫ

В ходе выполнения практической работы были получены навыки создания АТД для поставленной задачи и работы с динамическими массивами и векторами. Была изучена структура C++ для реализации абстрактного типа данных.

# 6 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)
2. Приложение к практическим работам – СДО (online-edu.mirea.ru)