**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»**

**(НГТУ)**

Институт радиоэлектроники и информационных технологий

Кафедра «Информационные радиосистемы»

**Контрольная работа по дисциплине**

**«Системное программирование»**

Выполнил:

Студент гр. 23-РЗ Шувалов Даниил

Проверил:

к.т.н., доцент кафедры ИРС Сидоров С.Б.

Нижний Новгород

2024

Содержание

[1. Постановка задачи 3](#__RefHeading___Toc4034_1917984385)

[2. Архитектура программной системы 4](#__RefHeading___Toc4036_1917984385)

[3. Алгоритм обработки 6](#__RefHeading___Toc4038_1917984385)

[Приложение 1. 7](#__RefHeading___Toc4366_1917984385)

[Версия 1.0 7](#__RefHeading___Toc4368_1917984385)

[Версия 0.5 10](#__RefHeading___Toc4370_1917984385)

[Версия 0.1 13](#__RefHeading___Toc4433_1917984385)

# 1. Постановка задачи

Применяя парадигму абстрактных типов данных и инкрементную модель разработки, создать программную систему для решения поставленной  
задачи. Все исходные данные должны вводиться со стандартного  
устройства ввода (с клавиатуры), то есть запрашиваться у пользователя.  
Результаты обработки должны быть выданы на стандартное устройство  
вывода (дисплей). Кроме окончательного варианта программной системы  
должны быть предоставлены и её промежуточные версии.  
  
Вариант 6

Дан целочисленный массив из n элементов. «Сожмите» массив, выбросив из него каждый второй элемент. Для решения задачи нельзя использовать дополнительный массив.

# 2. Архитектура программной системы

Разберём, использованный для решения поставленной задачи, абстрактный тип данных.

Application состоит из одного поля, а именно:

std::vector<int> array;

Поле array является последовательностью значений

В качестве основных функций в приложении, можно выделить:

int app\_run(Application& app);

bool app\_begin(Application& app);

bool app\_process(Application& app);

bool app\_end(Application& app);

app\_run – отвечает за запуск приложения в функции main, принимая на вход ссылку на переменную app по типу АТД. В main возвращается целочисленное значение, в качестве индикации успешности исполнения приложения.

app\_begin — отвечает за запрос данных у пользователя — т. е. За заполнение вектора.

app\_process – отвечает за «сжатие» массива данных.

app\_end – отвечает за отображение пользователю получившегося массива данных.

Так же, для взаимодействия с вектором, был введён модуль vector, в котором присутствуют следующие функции:

bool vector\_push(std::vector<int>& vector, int value);

bool vector\_display(std::vector<int>& vector);

int vector\_size(std::vector<int>& vector);

vector\_push – отвечает за добавление элементов в конец вектора.

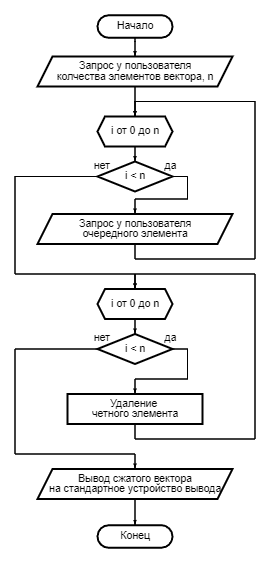
vector\_display – позволяет просмотреть все элементы вектора на стандартном устройстве ввода.

vector\_size – возвращает количество элементов в переданном в качестве аргумента, векторе.

# 3. Алгоритм обработки

Берется начало вектора - значение с индексом 0, к нему добавляется 1, получается второй элемент, он стирается. Так как один элемент стерся дальше добавляется +1 в следующем цикле, тем самым перескакивая нечетный элемент и что приводит к тому, что мы снова попадаем на четный, он опять удалится, и так до конца вектора.  
 Из-за того, что цикл ходит по стандартному, +1, стирание одного числа из вектора дает разницу в 2 числа. Так стираются только четные, по индексу, значения вектора.

Блок-схема программы



# Приложение 1.

## Версия 1.0

Файл main.cpp

#include "application.h"

int main(){

std::cout << "Array shortener program!" << std::endl;

Application app; // Ввод переменной по типу АТД

int ret = app\_run(app); // Запуск приложения

return ret;

}

Файл application.h

#ifndef NNTU\_APPLICATION\_H

#define NNTU\_APPLICATION\_H

#include "vector.h"

// Структура АТД

typedef struct Application{

std::vector<int> array;

}Application;

// Прототипы функций

int app\_run(Application& app);

bool app\_begin(Application& app);

bool app\_process(Application& app);

bool app\_end(Application& app);

#endif //NNTU\_APPLICATION\_H

Файл application.cpp

#include "application.h"

int app\_run(Application& app){

app\_begin(app); // Функция получения данных от пользотвателя, и сборки вектора

app\_process(app); // Функция сжатия вектора

app\_end(app); // Функция вывода сжатого вектора

return 0;

}

Файл app\_begin.cpp

#include "application.h"

bool app\_begin(Application& app){

int n, temp\_read\_value;

std::cout << "What's n?" << std::endl; //Запрос числа n

std::cin >> n;

for (int i = 0; i < n; ++i){ // Сборка вектора, путам запроса значений у пользователя

std::cout << "<" << i+1 << " of " << n << ">" << " item assigning:" << std::endl;

std::cout << "Input a value:" << std::endl

std::cin >> temp\_read\_value;

vector\_push(app.array, temp\_read\_value);

}

std::cout << "Initial vector is: ";

vector\_display(app.array); // Вывод получившегося вектора на стандартное устройство вывода

return true;

}

Файл app\_process.cpp

#include "application.h"

bool app\_process(Application& app){

for (int i = 1; i < vector\_size(app.array); ++i){ // Цикл поиска четных элементов вектора

app.array.erase(app.array.begin()+i); // Удаление четного элемента последовательности

}

return true;

}

Файл app\_process.cpp

#include "application.h"

bool app\_end(Application& app){

std::cout << "Processed vector is: ";

vector\_display(app.array); // Вывод сжатого вектора

return true;

}

Файл vector.h

#ifndef NNTU\_VECTOR\_H

#define NNTU\_VECTOR\_H

#include <iostream>

#include <vector>

// Прототипы функций

bool vector\_push(std::vector<int>& vector, int value);

bool vector\_display(std::vector<int>& vector);

int vector\_size(std::vector<int>& vector);

#endif //NNTU\_VECTOR\_H

Файл vector.cpp

#include "vector.h"

// Функция получения количества элементов вектора

int vector\_size(std::vector<int>& vector){

return vector.size();

}

// Функция добавления элемента в конец вектора

bool vector\_push(std::vector<int>& vector, int value){

vector.push\_back(value);

return true;

}

// Функция отображения вектора

bool vector\_display(std::vector<int>& vector){

for (int i : vector) {

std::cout << i << " ";

}

std::cout << std::endl;

return true;

}

## Версия 0.5

Файл main.cpp

#include "application.h"

int main(){

    std::cout << "Array shortener program!" << std::endl;

    Application app;

    int ret = app\_run(app);

    return ret;

}

Файл application.h

#ifndef NNTU\_APPLICATION\_H

#define NNTU\_APPLICATION\_H

#include "vector.h"

typedef struct Application{

    std::vector<int> array;

}Application;

int app\_run(Application& app);

bool app\_begin(Application& app);

bool app\_process(Application& app);

bool app\_end(Application& app);

#endif //NNTU\_APPLICATION\_H

Файл application.cpp

#include "application.h"

int app\_run(Application& app){

app\_begin(app); // Функция получения данных от пользотвателя, и сборки вектора

app\_process(app); // Функция сжатия вектора

app\_end(app); // Функция вывода сжатого вектора

return 0;

}

Файл app\_begin.cpp

#include "application.h"

bool app\_begin(Application& app){

return true;

}

Файл app\_process.cpp

#include "application.h"

bool app\_process(Application& app){

return true;

}

Файл app\_process.cpp

#include "application.h"

bool app\_end(Application& app){

return true;

}

Файл vector.h

#ifndef NNTU\_VECTOR\_H

#define NNTU\_VECTOR\_H

#include <vector>

#include <iostream>

bool vector\_push(std::vector<int>& vector, int value);

bool vector\_display(std::vector<int>& vector);

int vector\_size(std::vector<int>& vector);

#endif //NNTU\_VECTOR\_H

Файл vector.cpp

#include "vector.h"

int vector\_size(std::vector<int>& vector){

}

bool vector\_push(std::vector<int>& vector, int value){

    return true;

}

bool vector\_display(std::vector<int>& vector){

    return true;

}

## Версия 0.1

Файл main.cpp

#include <iostream>

int main(){

    std::cout << "Array shortener program!" << std::endl;

    return 0;

}