

**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

Проверил:

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

Нижний Новгород
2024

Содержание

1. Постановка задачи.....	3
2. Архитектура программной системы.....	4
3. Алгоритм обработки.....	6
Приложение 1.....	7
Версия 1.0.....	7
Версия 0.5.....	11
Версия 0.1.....	14

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 41.

В заданной последовательности целых чисел $\{a_1, a_2, \dots, a_n\}$ подсчитать количество отрицательных, положительных и нулевых элементов.

2. Архитектура программной системы

В качестве структуры был использован подход с применением парадигмы АТД, и структуры Application соответственно. Структура хранит в себе все переменные используемые данным приложением, необходимые для выполнения поставленной задачи. Разберём структуру Application, представленную в заголовочном файле application.h

```
typedef struct Application{  
  
    int n;  
    std::vector<int>array;  
    int odd = 0, even = 0, zero = 0;  
  
}Application;
```

Первый элемент — число n , по условию задачи. Представляет собой количество элементов последовательности для обработки, запрашивается у пользователя.

Второй элемент — вектор целочисленных. В него будет добавлено n элементов. Значение каждого элемента запрашивается у пользователя.

Третий элемент — три переменных, являющиеся счётчиками чётных, нечётных и нулей в введённой последовательности.

Теперь разберём объявления функций основного модуля application.

```
int app_launch(Application& app);  
bool app_input(Application& app);  
bool app_process(Application& app);  
bool app_output(Application& app);
```

Первая функция — основная функция запуска рассматриваемого приложения. Она вызывается в функции main, принимает на вход переменную по типу АТД. Далее в этой функции будут вызваны под-модули программы, а именно ввод информации, обработка и вывод.

Вторая функция — функция ввода данных. Сначала запрашивает у пользователя число *n*, далее исходя из этого числа, будет выполнен запрос значений последовательности в цикле, до числа *n*.

Третья функция — обработка последовательности. Пользователю будет показан получившийся вектор. Далее будут вызваны функции модуля `vector_handle`, для подсчёта количества чётных, нечётных и нулей.

Четвёртая функция — отвечает за вывод результата работы программы — количества чётных чисел, нечётных чисел и нулей.

Далее разберём модуль `vector_handle`. Глобальная задача модуля — предоставить удобный интерфейс для взаимодействия с последовательностью — вектором.

```
std::vector<int>vector_fill(int n);  
int vector_count_odd(std::vector<int>to_compute);  
int vector_count_even(std::vector<int>to_compute);  
int vector_count_zeroes(std::vector<int>to_compute);  
void vector_show(std::vector<int>to_compute);
```

Первая функция реализует запрос элементов последовательности у пользователя. Принимая на вход число *n*, в цикле, пользователю будет предложено ввести каждое значение последовательности. Возвращает новообразованную последовательность.

Вторая, третья, и четвёртая функции реализуют функционал подсчёта количества чётных, нечётных и нулей в последовательности. Каждая из этих функций возвращает получившийся результат.

Пятая функция позволяет избежать дублирования кода, при необходимости демонстрации последовательности пользователю. Функция принимает на вход последовательность, которую необходимо отобразить. И соответственно, отображает её.

3. Алгоритм обработки

Алгоритм обработки достаточно тривиален. После запуска приложения, пользователь введёт число n . Далее, исходя из этого числа, будут запрошены остальные значения. Далее будут вызваны функции подсчёта количества элементов, чётных, нечётных и нулей. После обработки последовательности, на стандартное устройство вывода будет представлена информация о количестве чётных, нечётных и нулей в последовательности.

Приложение 1.

Версия 1.0

Файл main.cpp

```
#include "application.h"

int main(){
    std::cout << "Welcome to Even/Odd/Zero evaluation program." << std::endl;
    Application app;
    int ret = app_launch(app);
    return ret;
}
```

Файл application.h

```
#ifndef HEADER_H
#define HEADER_H

#include <iostream>
#include "vector_handle.h"

typedef struct Application{
    int n;
    std::vector<int>array;
    int odd = 0, even = 0, zero = 0;
}Application;

int app_launch(Application& app);
bool app_input(Application& app);
bool app_process(Application& app);
bool app_output(Application& app);

#endif //HEADER_H
```

Файл application.cpp

```
#include "application.h"

int app_launch(Application &app){
    app_input(app);
    app_process(app);
    app_output(app);
    return 0;
}

bool app_input(Application& app){
    std::cout << "Input n: " << std::endl;
    std::cin >> app.n;
    app.array = vector_fill(app.n);
    return true;
}

bool app_process(Application& app){
    vector_show(app.array);
    app.odd = vector_count_odd(app.array);
    app.even = vector_count_even(app.array);
    app.zero = vector_count_zeroes(app.array);
    return true;
}

bool app_output(Application& app){
    std::cout << "ODD: " << app.odd << std::endl;
    std::cout << "EVEN: " << app.even << std::endl;
    std::cout << "ZERO: " << app.zero << std::endl;
    return true;
}
```

Файл vector_handle.h

```
#include <vector>
```



```

std::vector<int>vector_fill(int n);
int vector_count_odd(std::vector<int>to_compute);
int vector_count_even(std::vector<int>to_compute);
int vector_count_zeroes(std::vector<int>to_compute);
void vector_show(std::vector<int>to_compute);

```

Файл vector_handle.cpp

```

#include "vector_handle.h"

```

```

#include <iostream>

```

```

std::vector<int> vector_fill(int n){
    std::vector<int> to_return;
    int temp_int;
    for(int i = 0; i < n; ++i){
        std::cout << "Input a_" << i+1 << " / " << "a_" << n << " character:" <<
std::endl;
        std::cin >> temp_int;
        to_return.push_back(temp_int);
    }
    return to_return;
}

```

```

int vector_count_odd(std::vector<int>to_compute){
    int counter = 0;
    for(int i : to_compute){
        if ( i % 2 && i != 0){
            ++counter;
        }
    }
    return counter;
}

```

```

int vector_count_even(std::vector<int>to_compute){
    int counter = 0;

```

```

for(int i : to_compute){
    if ( !(i % 2) && i != 0){
        ++counter;
    }
}
return counter;
}

```

```

int vector_count_zeroes(std::vector<int>to_compute){
    int counter = 0;
    for(int i : to_compute){
        if ( i == 0 ){
            ++counter;
        }
    }
    return counter;
}

```

```

void vector_show(std::vector<int>to_compute){
    //DEBUG
    std::cout << "Vector is: " << std::endl;
    for (int i : to_compute){
        std::cout << i << " ";
    }
    std::cout << std::endl;
}

```

Версия 0.5

Файл main.cpp

```
#include "application.h"

int main(){
    std::cout << "Welcome to Even/Odd/Zero evaluation program." << std::endl;
    Application app;
    int ret = app_launch(app);
    return ret;
}
```

Файл application.h

```
#ifndef HEADER_H
#define HEADER_H

#include <iostream>
#include "vector_handle.h"

typedef struct Application{
    int n;
    std::vector<int>array;
    int odd = 0, even = 0, zero = 0;
}Application;

int app_launch(Application& app);
bool app_input(Application& app);
bool app_process(Application& app);
bool app_output(Application& app);

#endif //HEADER_H
```

Файл application.cpp

```
#include "application.h"
```

```

int app_launch(Application &app){
    app_input(app);
    app_process(app);
    app_output(app);
    return 0;
}

bool app_input(Application& app){
    return true;
}

bool app_process(Application& app){
    return true;
}

bool app_output(Application& app){
    return true;
}

```

Файл vector_handle.h

```

#include <vector>

std::vector<int>vector_fill(int n);
int vector_count_odd(std::vector<int>to_compute);
int vector_count_even(std::vector<int>to_compute);
int vector_count_zeroes(std::vector<int>to_compute);
void vector_show(std::vector<int>to_compute);

```

Файл vector_handle.cpp

```

#include "vector_handle.h"

#include <iostream>

std::vector<int> vector_fill(int n){
    std::vector<int> to_return;
    return to_return;
}

```

```
}
```

```
int vector_count_odd(std::vector<int>to_compute){  
    int counter = 0;  
    return counter;  
}
```

```
int vector_count_even(std::vector<int>to_compute){  
    int counter = 0;  
    return counter;  
}
```

```
int vector_count_zeroes(std::vector<int>to_compute){  
    int counter = 0;  
    return counter;  
}
```

```
void vector_show(std::vector<int>to_compute){  
  
}
```

Версия 0.1

Файл main.cpp

```
#include <iostream>
```

```
int main () {
```

```
    std::cout << "Welcome to Even/Odd/Zero evaluation program." << std::endl;
```

```
    return 0;
```

```
}
```