ЫЫ**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

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 $\{a_1, a_2, \ldots, a_n\}$ 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 двух полей, а именно:

int n;

std::vector<int> array;

По условию, n – количество элементов исходного массива. Для хранения элементов массива, была применяется вектор из <vector>.

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_erase(std::vector<int>& vector, int index);
bool vector_display(std::vector<int>& vector);
int vector_size(std::vector<int>& vector);
```

vector_push – отвечает за добавление элементов в конец вектора.

vector_erase – выполняет функцию «сжатия», принимая на вход ссылку на вектор и итератор цикла i из функции app_process.

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

Выполнение кода начинается в функции main, естественно. Сначала мы объявляем переменную app по типу АТД Application, его мы используем для хранения массива и числа n. После этого, мы вызываем основную исполняемую функцию приложение — app_run, передавая ей нашу переменную АТД. Возвращаемое значение app_run напрямую возвращается в main, но это не обязательно, и зависит от программного комплекса, в который будет встроено данное приложение.

Далее, по очереди, вызываются функции запроса данных у пользователя, обработки массива, и выдачи полученного результата на стандартное устройство вывода.

Приложение 1.

Версия 1.0

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{
    int n;
    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);
    app_process(app);
    app_end(app);
```

```
    return 0;
}

bool app_begin(Application& app){
    std::cout << "What's n?" << std::endl;
    std::cin >> app.n;
    int temp_read_value;
    for (int i = 0; i < app.n; ++i){
        std::cout << "<" << i+1 << " of " << app.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);
    return true;
}

bool app_process(Application& app){
    for (int i = 1; i < vector_size(app.array); ++i){
        vector_erase(app.array, i);
    }
    return true;
}

bool app_end(Application& app){
    std::cout << "Processed vector is: ";
    vector_display(app.array);
    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_erase(std::vector<int>& vector, int index);
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){
    return vector.size();
}

bool vector_push(std::vector<int>& vector, int value){
    vector.push_back(value);
    return true;
}

bool vector_erase(std::vector<int>& vector, int index){
    vector.erase(vector.begin()+index);
    return true;
}

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{
    int n;
    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);
    app_process(app);
    app_end(app);
    return 0;
}
```

```
bool app_begin(Application& app){

   return true;
}

bool app_process(Application& app){

   return true;
}

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_erase(std::vector<int>& vector, int index);
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_erase(std::vector<int>& vector, int index){

   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;
}
```