

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 1

Тема: Стандартные типы данных языка для представления многоэлементных однородных структур данных задачи в программе

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Юдов С.А.

Группа ИНБО-02-21

**Москва 2022**

**Отчет**

Задание 1

1. Условие задачи и варианта
   1. Разработать программу для выполнения операций на статическом массиве
   2. Задачи варианта
      1. Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31
      2. Удалить минимальное число, которое является степенью степень числа 2
      3. Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна.
2. Разработка первой задачи варианта
   1. Постановка задачи

Дано: Дан массив из n элементов целого типа int

Результат: Определить сколько в массиве простых чисел Мерсенна по заданным правилам.

* 1. Описание модели решения

Исходный массив a статически максимального размера N = 1000.

Текущий размер n массива mas определяет пользователь n <= N.

Число Мерсенна – это числа вида , где n – натуральное число

* 1. Декомпозиция
     1. Список подзадач

1. Проход по числам массива
2. Определение верное число или нет
3. Подсчёт подходящих чисел
   * 1. Определение прототипов функций
4. Заполнение исходного массива значениями с клавиатуры

Предусловие: n – число заполняемых элементов, 0<n<=N, где N – максимальное число элементов в массиве.

Постусловие: Заполненный массив из n-элементов.

void inputarray(int\* x, int &n)

1. Вывод итогового массива

Предусловие: n > 0

Постусловие: Вывод итогового массива

void outputarray(int\* x, int n)

1. Функции декомпозиции

**// Определение является ли элемент числом Мерсенна**

Предусловие: целое X – натуральное число

Постусловие: результат целый, количество элементов

int checkMersenN(int \*x, int n)

* 1. Разработка алгоритмов

1. Алгоритмы ввода и вывода массива определим позже
2. Алгоритмы задачи декомпозиции

Алгоритм функции int checkMersenN(int \*x, int n)

int checkMersenN(int \*x, int n) {

int m = 0;

for (int i = 0; i < n; i++) {

if (checkMersen(x[i])) {

m += 1;

}

}

return m;

}

1. Разработка второй задачи варианта
   1. Постановка задачи: Удалить минимальное число, которое является степенью двойки
   2. Декомпозиция задачи
      1. Список подзадач
2. Поиск индекса минимального числа
3. Удаление элемента
4. Сдвиг массива
   * 1. Прототипы функций
5. Функции задачи декомпозиции:

**//функция для сдвига массива и вставки элемента**

Предусловие: массив из n элементов, индекс ячейки

Постусловие: получение изменённого массива

void findmindel(int \*x, int &n)

* 1. Разработка алгоритма

1. Алгоритм функции void findmindel(int \*x, int &n)

void findmindel(int \*x, int &n) {

bool flag = false;

long long int min = 1000000000000;

int min\_i = -1;

for (int i = 0; i < n; i++) {

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

}

}

if (min\_i > -1) {

for (int i = min; i < n; i++) {

x[i] = x[i + 1];

}

n--;

}

}int checkMersenN(int \*x, int n) {

int m = 0;

for (int i = 0; i < n; i++) {

if (checkMersen(x[i])) {

m += 1;

}

}

return m;

1. Разработка третьей задачи варианта
   1. Постановка задачи: Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна
   2. Описание модели решения: вставка нужного элемента и сдвиг массива
   3. Декомпозиция задачи
      1. Список подзадач
   4. Поиск индекса элемента с максимальным значением числа Мерсенна
   5. Удаление элемента
   6. Сдвиг массива
      1. Определение функций

Функции задачи декомпозиции

**//Поиск индекса элемента с максимальным значением числа Мерсенна в массиве**

Предусловие: массив из целых чисел, длина массива

Постусловие: получение изменённого массива

void findmaxins(int \*x, int &n, int nw)

bool degreeoftwo(int entered) // Определение является ли число степенью двойки

bool prost(int n) // Определение является ли число простым

bool checkMersen(int x) // Определение является ли число числом Мерсенна

int checkMersenN(int \*x, int n) // Определение количества чисел Мерсенна в массиве

void addelem(int \*&arr, int &size, const int value, int currid) // Добавление нового элемента в любое место массива

void delelem(int \*&arr, int &size, int id) // Удаление любого элемента из списка

void findmindel(int \*&x, int &n) // Поиск и удаление минимального элемента степени двойки

void findmaxins(int \*&x, int &n, int nw) // Поиск максимального числа Мерсенна и вставка любого числа перед ним

void inputarray(int \*x, int n) // Ввод массива с клавиатуры

void outarray(int \*x, int n) // Вывод массива в консоль

* 1. Разработка алгоритма
     1. Алгоритм void findmaxins(int \*x, int &n, int nw)

void findmaxins(int \*x, int &n, int nw) {

int max = -1;

for (int i = 0; i < n; i++) {

if (x[i] > max && checkMersen(x[i])) {

max = i;

}

}

if (max > -1) {

for (int i = n; i > max; i--) {

x[i + 1] = x[i];

}

x[max + 1] = nw;

}

n++;

}

1. Код программы

#include "iostream"

using namespace std;

const int N = 1000;

bool degreeoftwo(int entered) {

int temp = 1;

while (temp < entered) {

temp \*= 2;

}

if (temp == entered) {

return true;

} else {

return false;

}

}

bool prost(int n) {

for (int i = 2; i < n; i++) {

if (n % i == 0) {

return false;

}

return true;

}

}

bool checkMersen(int x) {

if (prost(x) && degreeoftwo(x + 1)) {

return true;

}

return false;

}

int checkMersenN(int \*x, int n) {

int m = 0;

for (int i = 0; i < n; i++) {

if (checkMersen(x[i])) {

m += 1;

}

}

return m;

}

void findmindel(int \*x, int &n) {

bool flag = false;

int min\_i = -1

long long int min = 1000000000000;

for (int i = 0; i < n; i++) {

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

}

}

if (min\_i > -1) {

for (int i = min\_i; i < n; i++) {

x[i] = x[i + 1];

}

n--;

}

}

void findmaxins(int \*x, int &n, int nw) {

int max\_i = -1;

int max = 0;

for (int i = 0; i < n; i++) {

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i];

}

}

if (max > -1) {

for (int i = n; i > max\_i; i--) {

x[i + 1] = x[i];

}

x[max\_i + 1] = nw;

}

n++;

}

void inputarray(int \*x, int n) {

cout << "Wedite " << n << " chisel\n";

for (int i = 0; i < n; i++) {

cin >> x[i];

}

}

void outarray(int \*x, int n) {

cout << "Massiv " << n << " chisel\n";

for (int i = 0; i < n; i++) {

cout << x[i] << ' ';

}

cout << endl;

}

void inputRandarray(int \*x, int n) {

srand(time(0));

for (int i = 0; i < n; i++) {

x[i] = rand() % 100;

}

}

int main() {

int a[N];

inputarray(a, size);

cout << "Zadanie 1.1 " << checkMersenN(a, size) << endl;

cout << "Zadanie 1.2 " << endl;

findmindel(a, size);

outarray(a, size);

cout << "Zadanie 1.3 " << endl;

findmaxins(a, size, 5555);

1. Набор тестов для тестирования программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  checkMersenN(a, size) | 2 | 2 | Тест пройден |
| 2 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmindel(a, n); | 0 3 5 7 10 12 16 | 0 3 5 7 10 12 16 | Тест пройден |
| 3 | size = 8  nw = 5555  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmaxins(a, size, nw); | 0 3 5 7 5555 10 12 16 | 0 3 5 7 5555 10 12 16 | Тест пройден |

Задание 2

1. Условие задачи варианта
   1. Разработать программу для выполнения операций на динамическом массиве
   2. Задачи варианта
      1. Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31
      2. Удалить минимальное число, которое является степенью степень числа 2
      3. Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна.
2. Декомпозиция
   1. Список подзадач
      1. Поиск индекса элемента с максимальным значением числа Мерсенна
      2. Удаление элемента
      3. Сдвиг массива
3. Определение функций

bool degreeoftwo(int entered) // Определение является ли число степенью двойки

bool prost(int n) // Определение является ли число простым

bool checkMersen(int x) // Определение является ли число числом Мерсенна

int checkMersenN(int \*x, int n) // Определение количества чисел Мерсенна в массиве

void addelem(int \*&arr, int &size, const int value, int currid) // Добавление нового элемента в любое место массива

void delelem(int \*&arr, int &size, int id) // Удаление любого элемента из списка

void findmindel(int \*&x, int &n) // Поиск и удаление минимального элемента степени двойки

void findmaxins(int \*&x, int &n, int nw) // Поиск максимального числа Мерсенна и вставка любого числа перед ним

void inputarray(int \*x, int n) // Ввод массива с клавиатуры

void outarray(int \*x, int n) // Вывод массива в консоль

1. Кодирование алгоритма программы

#include <iostream>

#include <cstdlib>

using namespace std;

bool degreeoftwo(int entered) {

int temp = 1;

while (temp < entered) {

temp \*= 2;

}

if (temp == entered) {

return true;

} else {

return false;

}

}

bool prost(int n) {

for (int i = 2; i < n; i++) {

if (n % i == 0) {

return false;

}

return true;

}

}

bool checkMersen(int x) {

if (prost(x) && degreeoftwo(x + 1)) {

return true;

}

return false;

}

int checkMersenN(int \*x, int n) {

int m = 0;

for (int i = 0; i < n; i++) {

if (checkMersen(x[i])) {

m += 1;

}

}

return m;

}

void addelem(int \*&arr, int &size, const int value, int currid) {

size++;

arr = (int \*) realloc(arr, sizeof(int) \* (size));

if (!arr) exit(1);

for (int i = size - 1; i > currid; i--) {

arr[i] = arr[i - 1];

}

arr[currid] = value;

}

void delelem(int \*&arr, int &size, int id) {

for (int i = id; i < size; i++) {

arr[i] = arr[i + 1];

}

size--;

arr = (int \*) realloc(arr, sizeof(int) \* size);

}

void findmindel(int \*&x, int &n) {

bool flag = false;

min\_i = -1;

long long int min = 1000000000000;

for (int i = 0; i < n; i++) {

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

flag = true;

}

}

if (min\_i > -1) {

delelem(x,n,min);

}

}

void findmaxins(int \*&x, int &n, int nw) {

int max\_i = -1;

int max = 0;

for (int i = 0; i < n; i++) {

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i]

}

}

if (max\_i > -1) {

addelem(x,n,nw,max + 1);

}

}

void inputarray(int \*x, int n) {

cout << "Wedite " << n << " chisel\n";

for (int i = 0; i < n; i++) {

cin >> x[i];

}

}

void outarray(int \*x, int n) {

cout << "Massiv " << n << " chisel\n";

for (int i = 0; i < n; i++) {

cout << x[i] << ' ';

}

cout << endl;

}

int main() {

int size = 8;

int \*dinarr = (int \*) malloc(sizeof(int) \* size);

inputarray(dinarr, size);

outarray(dinarr, size);

cout << "Zadanie 1.1 " << checkMersenN(dinarr, size) << endl;

cout << "Zadanie 1.2 " << endl;

findmindel(dinarr, size);

outarray(dinarr, size);

cout << "Zadanie 1.3 " << endl;

findmaxins(dinarr, size, 5555);

outarray(dinarr,size);

free(dinarr);

dinarr = nullptr;

return 0;

}

1. Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  checkMersenN(a, size) | 2 | 2 | Тест пройден |
| 2 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmindel(a, n); | 0 3 5 7 10 12 16 | 0 3 5 7 10 12 16 | Тест пройден |
| 3 | size = 8  nw = 5555  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmaxins(a, size, nw); | 0 3 5 7 5555 10 12 16 | 0 3 5 7 5555 10 12 16 | Тест пройден |

Задание 3

1. Условие задачи варианта
   1. Разработать программу для выполнения всех операций варианта над динамическим массивом, используя контейнер <vector> для его представляя в программе.
   2. Задачи варианта
      1. Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31
      2. Удалить минимальное число, которое является степенью степень числа 2
      3. Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна.
2. Декомпозиция
   1. Список подзадач
      1. Поиск индекса элемента с максимальным значением числа Мерсенна
      2. Удаление элемента
      3. Сдвиг массива
3. Определение функций

bool degreeoftwo(int entered) // Определение является ли число степенью двойки

bool prost(int n) // Определение является ли число простым

bool checkMersen(int x) // Определение является ли число числом Мерсенна

int checkMersenN(vector<int> &x, int n) // Определение количества чисел Мерсенна в векторе

void findmindel(vector<int> &x) // Поиск и удаление минимального элемента степени двойки

void findmaxins(vector<int> &x, int nw) // Поиск максимального числа Мерсенна и вставка любого числа перед ним

void inputvector(vector<int> &x, int size) // Ввод вектора с клавиатуры

void outputvector(vector<int> &x) // Вывод вектора в консоль

1. Кодирование программы

#include <iostream>

#include <vector>

using namespace std;

bool degreeoftwo(int entered) {

int temp = 1;

while (temp < entered) {

temp \*= 2;

}

if (temp == entered) {

return true;

} else {

return false;

}

}

bool prost(int n) {

for (int i = 2; i < n; i++) {

if (n % i == 0) {

return false;

}

return true;

}

}

bool checkMersen(int x) {

if (prost(x) && degreeoftwo(x + 1)) {

return true;

}

return false;

}

int checkMersenN(vector<int> &x) {

int m = 0;

for (vector<int>::iterator i = x.begin(); i != x.end(); i++) {

if (checkMersen(\*i)) {

m += 1;

}

}

return m;

}

void findmindel(vector<int> &x) {

bool flag = false;

long long int min = 1000000000000;

int min\_i = -1;

for (int i = 0; i < x.size(); i++) {

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

}

}

if (min\_i > -1) {

x.erase(x.begin() + min\_i);

}

}

void findmaxins(vector<int> &x, const int nw) {

int max\_i = -1;

int max = -1;

for (int i = 0; i < x.size(); i++) {

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i];

}

}

if (max\_i > -1) {

x.insert(x.begin() + max\_i + 1, nw);

}

}

void inputvector(vector<int> &x, int size) {

cout << "Wedite " << size << " chisel\n";

for (int i = 0; i < size; i++) {

int temp;

cin >> temp;

x[i] = temp;

}

}

void outputvector(vector<int> &x) {

cout << "Massiv " << x.size() << " chisel\n";

for (auto const &element: x) {

cout << element << ' ';

}

cout << endl;

}

int main() {

int size = 8;

vector<int> myVector(size);

inputvector(myVector, size);

outputvector(myVector);

cout << "Zadanie 1.1 " << checkMersenN(myVector) << endl;

cout << "Zadanie 1.2 " << endl;

findmindel(myVector);

outputvector(myVector);

cout << "Zadanie 1.3 " << endl;

findmaxins(myVector, 5555);

outputvector(myVector);

return 0;

}

1. Тестирование программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | size = 8  vector<int> myVector = {0, 3, 5, 8, 7, 10, 12, 16}  checkMersenN(a) | 2 | 2 | Тест пройден |
| 2 | vector<int> myVector = {0, 3, 5, 8, 7, 10, 12, 16}  findmindel(a); | 0 3 5 7 10 12 16 | 0 3 5 7 10 12 16 | Тест пройден |
| 3 | size = 8  nw = 5555  vector<int> myVector = {0, 3, 5, 8, 7, 10, 12, 16}  findmaxins(a, nw); | 0 3 5 7 5555 10 12 16 | 0 3 5 7 5555 10 12 16 | Тест пройден |

**Вывод**

В течение выполнения данной работы, мной были получены знания по работе со стандартными типами данных языка для представления многоэлементных однородных структур данных задачи в программе. Также были приобретены навыки создания алгоритмов операций над одномерными массивами и по их реализации.

**Список информационных источников**

* Учебник по C++ [http://www.lmpt.univ-tours.fr/~volkov/C++.pdf](https://d.docs.live.net/793bf2b94bcfd004/Рабочий%20стол/http:/www.lmpt.univ-tours.fr/~volkov/C++.pdf)
* Документация по языку C++ [https://docs.microsoft.com/ruru/cpp/?view=msvc-160](https://d.docs.live.net/793bf2b94bcfd004/Рабочий%20стол/https:/docs.microsoft.com/ruru/cpp/?view=msvc-160)