|  |  |
| --- | --- |
| C:\Documents and Settings\admin\Рабочий стол\МИРЭА_ЭМБЛЕМА_приказ.JPG | |
| МИНОБРАНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «**МОСКОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**  **МИРЭА** | |
| Институт информационных технологий  Кафедра математического обеспечения и стандартизации информационных технологий | |
| **Отчет по лабораторной работе №1**  **Тема работы «ОДНОМЕРНЫЙ МАССИВ»** | |
| Выполнил: |  |
| Студент группы ИКБО-07-17  Отчет принял:  Преподаватель кафедры МОСИТ | Акжигитов Р. Р.  Миронов Антон Николаевич |
| Москва  2018 | |

Оглавление

[Задание 1 3](#_Toc507631498)

[Условия задания 3](#_Toc507631499)

[Декомпозиция 3](#_Toc507631500)

[Определение функций 3](#_Toc507631501)

[Реализация функций 3](#_Toc507631502)

[Задание 2 3](#_Toc507631503)

[Условия задания 3](#_Toc507631504)

[Декомпозиция 3](#_Toc507631505)

[Определение функций 4](#_Toc507631506)

[Реализация функций 4](#_Toc507631507)

[Функции ввода\вывода массива\значений 6](#_Toc507631508)

[Кодирование алгоритма программы 7](#_Toc507631509)

[Вывод 9](#_Toc507631510)

# Задание 1

# Условия задания

Дан массив Х из М целых чисел( 0<M<97;1000<X[i]<100). Найти максимальное из чисел, встречающееся в данном массиве более 1 раза.

# Декомпозиция

* Отметить частоту встречи всех элементов массива (используя массив-словарь, где индекс – число, а элемент под индексом – частота)
* Пройтись по массиву-словарю в обратном направлении (от большего числа до меньшего) до первого встречного элемента, который встречается более 1 раза.

# Определение функций

int findMax(int \*arr, int n)

На вход подается указатель и длина массива, на выходе найденное максимальное число.

# Реализация функций

int findMax(int \*arr, int n) {

//int \*tempArr = (int\*)malloc(highLimit - lowLimit + 1);

int tempArr[highLimit - lowLimit + 1];

int max = lowLimit - 1;

for (int i = 0; i < highLimit - lowLimit + 1; tempArr[i++] = 0);

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

tempArr[arr[i] - lowLimit]++;

if (tempArr[arr[i] - lowLimit] > 1) max = max < arr[i] ? arr[i] : max;

}

cout << endl;

printArray(tempArr, highLimit - lowLimit + 1);

//free(tempArr);

return max;

}

# Задание 2

# Условия задания

Дан одномерный целочисленный массив А, состоящий из N элементов. N заданное натуральное число.

1. Cформировать массив В из элементов массива А, являющихся простыми числами.
2. Выполнить сортировку части массива В между первым минимальным и последним минимальным значениями методом прямого обмена.
3. Удалить все элементы массива, сумма цифр которых кратна 7.
4. Найти самую длинную подпоследовательность массива А, состоящую из элементов, которые являются четными числами.

# Декомпозиция

1. Массив простых чисел. Для его получения необходимо пройтись по данному массиву и определить является ли каждое число простым, если да, то добавить его в итоговый массив. Две подпрограммы: проверка на простоту и итерирование входного массива, выделение динамической памяти под новый массив.
2. Поиск минимального с начала массива и с конца, возврат индекса. Сортировка Пузырьком части массива между 2-мя переданными индексами.
3. Функция подсчета суммы цифр, процедура удаления элемента заданного индекса, объединяющая процедура итерирования массива и проверки на сумму.
4. Процедура, которая вернет по ссылке массив скопированный из главного, нужно найти индекс начала самой длинной последовательности и ее длину, для дальнейшего копирования.
   1. Функция копирования массива.
   2. Общая идея поиска: идем по элементам массива, если очередной элемент четен, то инкрементируем временную переменную. Если это последний элемент или следующий элемент нечетен, то сравниваем временные переменные с результативными, обнуляем временную. Если это первый элемент или предыдущий элемент нечетен, то временный индекс старта меняем на очередной индекс. Выделяем память для нового массива, копируем.

# Определение функций

1. int isPrime(int n) – проверка на простоту
   1. void createPrimeArr(int \*arrIN, int lengthIN, int \*&arrOUT, int &lengthOUT) – возвращение нового массива из простых чисел
2. int\* bubbleSort(int \*arr, int length, int startInx = -1, int endInx = -1) – алгоритм сортировки прямыми обменами
   1. int getMinimumIndex(int \*arr, int n) – получение индекса первого минимума
   2. int getLastMinimumIndex(int \*arr, int n) – получение индекса последнего минимума
3. void deleteDigitsSum7(int \*&arr, int &length) – удаление из массива определенных элементов
   1. int getSumOfDigits(int n) – сумма цифр числа
   2. void deleteItem(int \*&arr, int &length, int n) – удаление n-го элемента из массива
4. void evenSequence(int \*arr, int lengthIn, int \*&arrOut, int &lengthOut) – получение четного массива максимальной длины
   1. void copyArray(int \*arr, int start, int lengthOfCopy, int \*&arrOut) – скопировать часть массива со start-позиции lenghtOfCopy элементов.

# Реализация функций

// проверка на простоту чилса

int isPrime(int n) {

if (n < 2) return 0;

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

if (!(n % i)) return 0;

}

return 1;

}

// формирование массива простых чисел из другого массива

void createPrimeArr(int \*arrIN, int lengthIN, int \*&arrOUT, int &lengthOUT) {

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

if (isPrime(arrIN[i])) {

arrOUT = (int\*)realloc(arrOUT, sizeof(int) \* ++lengthOUT);

arrOUT[lengthOUT - 1] = arrIN[i];

}

}

}

// алгоритм сортировки прямого обмена, с начальным и конечным индексами сортировки

int\* bubbleSort(int \*arr, int length, int startInx = -1, int endInx = -1) {

if (startInx < 0 && endInx < 0) {

startInx = 0;

endInx = length;

}

for (int i = startInx; i < endInx; i++)

for (int j = startInx; j < endInx - 1; j++)

if (arr[j] > arr[j + 1]) swap(arr[j], arr[j + 1]);

return arr;

}

// получение индекса первого минимального

int getMinimumIndex(int \*arr, int n) {

int min = arr[0];

int minIndex = 0;

for (int i = 1; i < n; i++) {

if (min > arr[i]) {

min = arr[i];

minIndex = i;

}

}

return minIndex;

}

// получение индекса последнего минимального

int getLastMinimumIndex(int \*arr, int n) {

int min = arr[n - 1];

int minIndex = n - 1;

for (int i = n - 1; i > 0; i--) {

if (min > arr[i]) {

min = arr[i];

minIndex = i;

}

}

return minIndex;

}

// удаление из массива n-ого элемента

void deleteItem(int \*&arr, int &length, int n) {

for (int i = n; i < length - 1; i++) {

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

}

arr = (int\*)realloc(arr, sizeof(int) \* (length - 1));

length--;

}

// сумма цифр числа

int getSumOfDigits(int n) {

int res = 0;

do {

res += n % 10;

} while ((n /= 10) > 0);

return res;

}

// удаление из массива всех чисел, сумма цифр которых равна 7

void deleteDigitsSum7(int \*&arr, int &length) {

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

if (getSumOfDigits(arr[i]) == 7) deleteItem(arr, length, i);

}

}

// подпрограмма копирования массива, с start индекса lengthOfCopy элементов

void copyArray(int \*arr, int start, int lengthOfCopy, int \*&arrOut) {

for (int i = start; i < start + lengthOfCopy; i++) {

arrOut[i - start] = arr[i];

}

}

// получение максимальной длины четной подпоследовательности

void evenSequence(int \*arr, int lengthIn, int \*&arrOut, int &lengthOut) {

int subLength = 0, temp = 0, start = 0, tempStart = 0;

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

if (i == 0 || arr[i - 1] % 2) {

tempStart = i;

}

if (!(arr[i] % 2)) temp++;

if (i == lengthIn - 1 || arr[i + 1] % 2) {

if (temp > subLength) {

start = tempStart;

subLength = temp;

}

temp = 0;

}

}

lengthOut = subLength;

arrOut = (int\*)realloc(arrOut, sizeof(int) \* lengthOut);

copyArray(arr, start, subLength, arrOut);

}

# Функции ввода\вывода массива\значений

// контроль вводимых значений

void input(int &number, int min, int max) {

while (!(cin >> number) || (number < min || number > max)) {

cout << "\nPlease input integer number [" << min << ".." << max << "]... Your choice's: ";

}

}

// подпрограмма печати массива

void printArray(int \*arr, int n) {

cout << "[";

for (int i = 0; i < n - 1; i++) {

cout << arr[i] << ", ";

}

cout << arr[n - 1] << "]" << endl;

}

// клавиатурный ввод массива

void manualEnter(int \*arr, int n, int min = lowLimit, int max = highLimit) {

cout << "Enter whole array divided by spacebar: ";

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

/\*if (!(cin >> arr[i])) {

cout << "\nError, please input integer:";

i--;

}\*/

input(arr[i], min, max);

}

cout << "Array is entered successfully..." << endl;

}

// заполнение массива случайными числами

int\* randomEnter(int \*arr, int n, int low = lowLimit, int high = highLimit) {

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

arr[i] = low + rand() % (high - low + 1); // (-999 + 99 + 1); // -999 99

}

return arr;

}

# Кодирование алгоритма программы

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int arr[maxLength];

int finishFlag = 0;

//int \*A = NULL, \*B = NULL, \*seqArr = NULL;

while (finishFlag != 1) {

system("cls");

int choice = -1;

cout << "IKBO-07-17 Akzhigitov Radmir (I variant) 28.02.2018\nMain menu:\n1) Static array\n2) Dynamic array\n3) Exit\n";

cout << "Your choice's: ";

input(choice, 1, 3);

switch (choice) {

case 1: {

int arrayLength = -1;

cout << "Input length of static array: ";

input(arrayLength, 1, maxLength);

int subChoice = 0;

cout << "Choose keyboard input (1) or random input (2): ";

input(subChoice, 1, 2);

if (subChoice == 1) manualEnter(arr, arrayLength);

else randomEnter(arr, arrayLength);

cout << "Inputted array: ";

printArray(arr, arrayLength);

int founded;

if ((founded = findMax(arr, arrayLength)) == lowLimit - 1) {

cout << "The number matching all conditions wasn't found..." << endl;

}

else {

cout << "The max of array that occurs more once is " << founded << endl;

}

system("pause");

break;

}

case 2: {

int exitFlag = 0;

int lengthA = -1;

cout << "Input length of dynamic array: ";

input(lengthA, 1, 1000);

int \*A = NULL;

A = (int\*)realloc(A, sizeof(int) \* lengthA);

int \*B = NULL;

int lengthB = 0;

int \*seqArr = NULL;

int lengthSeq = 0;

int subChoice = 0;

cout << "Choose keyboard input (1) or random input (2): ";

input(subChoice, 1, 2);

int min, max;

cout << "Input min number: ";

input(min, -10000, 10000);

cout << "Input max number: ";

input(max, -10000, 10000);

if (max < min) swap(min, max);

if (subChoice == 1) manualEnter(A, lengthA, min, max);

else randomEnter(A, lengthA, min, max);

while (exitFlag != 1) {

cout << "Inputted array: ";

printArray(A, lengthA);

system("pause");

subChoice = 0;

system("cls");

cout << "1) Create prime numbers' array\n2) Bubble sort between first and last min\n3) Delete all items that digits sum is 7\n4) The longest even subsequence\n5) Exit\n";

input(subChoice, 1, 5);

switch (subChoice) {

case 1: {

createPrimeArr(A, lengthA, B, lengthB);

cout << "It's created prime array: ";

printArray(B, lengthB);

system("pause");

break;

}

case 2: {

if (lengthB == 0) { cout << "Prime array wasn't created or his length is 0...\n"; break; }

cout << "Index of first min is: " << getMinimumIndex(B, lengthB) << "\nIndex of last min is: " << getLastMinimumIndex(B, lengthB) << "\n";

cout << "Array before sorting: ";

printArray(B, lengthB);

B = bubbleSort(B, lengthB, getMinimumIndex(B, lengthB), getLastMinimumIndex(B, lengthB));

cout << "Sorted array betwen first and last mins: ";

printArray(B, lengthB);

system("pause");

break;

}

case 3: {

cout << "Start array: ";

printArray(A, lengthA);

deleteDigitsSum7(A, lengthA);

cout << "Edited array: ";

printArray(A, lengthA);

system("pause");

break;

}

case 4: {

evenSequence(A, lengthA, seqArr, lengthSeq);

cout << "Longest even subsequence: ";

printArray(seqArr, lengthSeq);

system("pause");

break;

}

case 5: {

exitFlag = 1;

break;

}

}

}

free(seqArr);

free(B);

free(A);

break;

}

case 3: { finishFlag = 1; break; }

}

}

system("pause");

return 0;

}

# Вывод

Выполнив работу, я многое узнал про структуру данных массив, различия статистического и динамического массивов, выделение памяти, передачу в функции (через указатель или ссылку на указатель), возвращение из функций. Есть два варианта выделения и очищения памяти для массивов: старый стиль – malloc/realloc/free, новый стиль – new[], delete[]. Также научился составлять простые текстовые меню.