МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № **3**

**Сортировка целочисленного вектора**

по курсу   
Основы программирования

Группа 6101

Студент А. И. Афанасенко

(*подпись*)

Преподаватель,

к.ф.-м.н. C. С. Стафеев

(*подпись*)

Самара 2023

Задание

Задание на лабораторную работу приведено в таблице 1.

**Таблица 1 –** Задание на ЛР № 3

|  |  |
| --- | --- |
| Основное задание на ЛР | Написать консольное приложение, предназначенное для сортировки целочисленного вектора собственной функцией, реализующей алгоритм сортировки согласно варианту. **Входные данные**: количество элементов. **Выходные данные**: исходный и отсортированный массивы. |

Вариант для задания приведен в таблице 2.

**Таблица 2 –** Вариант основного задания на ЛР № 3

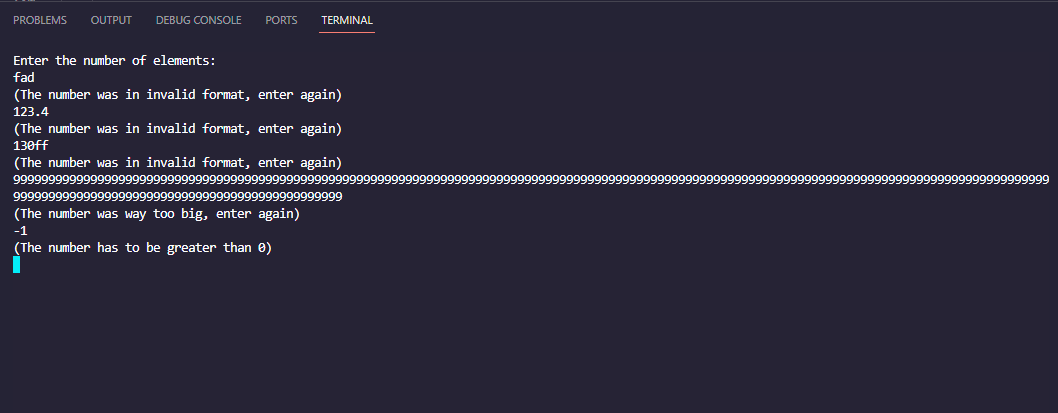
|  |  |
| --- | --- |
| Вариант 16 | Быстрая сортировка с разделением на 3 части |

Ход выполнения работы

В рамках выполнения задания я реализовал сортировку массива случайных чисел алгоритмом быстрой сортировки с разделением на три части. Выполнение функции main начинается с чтения размера данного массива от пользователя. Чтение размера производится функцией readIntLine. Она получает ввод пользователя с помощью getline(), затем использует функцию из стандартной библиотеки stoi чтобы получить int из строки. stoi может выкинуть ошибки конвертации, которые затем обрабатываются конструкцией try...catch. Таким образом, readIntLine обрабатывает случаи, когда вместо числа вводится обычная строка или число вводится слишком большое, как показано на Рисунке 1.

Интересной особенностью stoi является то, что он может без ошибок конвертировать такую строку как, например, 140afd – когда за числами следуют буквы. Для того чтобы выкидывать ошибку в таком случае, readIntLine использует регулярное выражение [+-]?[0-9]+([,.][0-9]+ чтобы проверить, что строка является числом и ничем больше.

Также после вызова функции readIntLine добавлена проверка на то, чтобы введенное число было больше 0. (Рисунок 1)

Рисунок 1 — проверка ввода

Затем я вызываю функцию generateRandomArray, которая создает вектор целых значений и с помощью функции rand() генерирует каждый ее элемент. Также результат rand() делиться по модулю на 10 — чтобы значения были не больше 10 и было легко проверять работу алгоритма.

После вызывается функция printArray — печатается получившийся массив случайных чисел.

После данных подготвительных шагов вызывается сам алгоритм — функция quickSort().

Суть работы алгоритма заключается в выбирании какого то случайного элемента из массива — якоря, и последующем логическом разбиении массива на три части:

1. Левая часть — элементы меньше якоря

2. Средняя часть — элементы равные якорю

3. Правая часть — элементы больше якоря

Разбиение выполняется с помощью цикла, проходящего по элементам массива, и сравнивающего каждый элемент с якорем, после чего:

1. Если элемент меньше якоря, он передвигается в начало массива

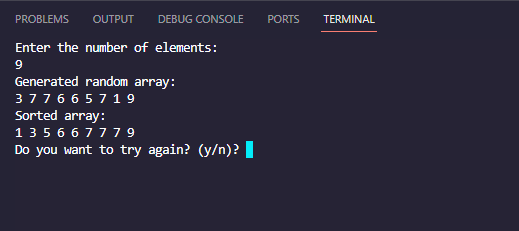
2. Если элемент равен якорю, он пропускается (остается на месте)

3. Если элемент больше якоря, он передвигается в конец массива

Таким образом, после разбиения, элементы равные якорю находятся в их соответствующих местах в конечном отсортированном массиве.

Все данные шаги обернуты в функцию threeWayPartition(). После первого разбиения, мы просто вызываем функцию quickSort рекурсивно для каждой левой и правой части массива, пока все элементы не окажутся на правильных местах, массив не будет отсортирован.

После выполнения quickSort(), функция printArray() вызывается снова, и возвращает пользователю отсортированный массив. (Рисунок 2)

Рисунок 2 — результат выполнения программы

Заключение

Цель лабораторной работы – написать программу, предназначенную для сортировки массива используя алгоритм быстрой сортировки с разделением на 3 части, достигнута. В ходе выполнения лабораторной работы я изучил алгоритм быстрой сортировки, понял как его модифицировать чтобы достигнуть разделения на 3 части, приобрел навыки по использованию функций, рекурсии. Самой интересной частью работы было использование целочисленных указателей для логического разделения массива на 3 части.

Литература

1. Безус, Е.А. Лабораторные работы по курсу «Методы программирования» (язык программирования C++): Методические указания [Текст] / Сост. Е.А. Безус, Е.С. Козлова, С.С. Стафеев Самара, 2019. – 56 с.
2. Страуструп, Б. Язык программирования С++ / Б. Страуструп. – М:Бином, 2011 –1136 с.
3. Шилдт. Г. С++. Полное руководство / Г. Шилдт. – М: "ООО И.Д. Вильямс", 2019 – 800 с.
4. Страуструп, Б. Программирование: принципы и практика использования С++, испр. изд.: Пер. с англ. / Б. Страуструп. – М: "ООО И.Д. Вильямс", 2011 –1244 с.
5. Прата, С. Язык программирования С++. Лекции и упражения , 6-е изд.: Пер. с англ. / С. Прата. – Киев: "Диалектика", 2018 –1248 с.
6. Библиографическое описание документа. Общие требования и правила составления [Электронный ресурс] / сост.: В.С. Крылова, С.М. Григорьевская, Е.Ю. Кичигина // Официальный интернет-сайт научной библиотеки Томского государственного университета. – Электрон. дан. – Томск, [2010]. – <http://www.lib.tsu.ru/win/produkzija/metodichka/metodich.html> (дата обращения: 10.09.2019).

Приложение

#include <iostream>

#include <string>

#include <regex>

#include <stdlib.h>

#include <time.h>

//Quicksort with 3-way partition

using namespace std;

const int ARR\_PRINT\_MAX\_SIZE = 50;

const bool DEBUG = false;

void printArray(vector<int> &arr)

{

for (int i = 0; i < arr.size() && i < ARR\_PRINT\_MAX\_SIZE; i++)

{

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

}

if (arr.size() > ARR\_PRINT\_MAX\_SIZE)

{

cout << "... (" << arr.size() - ARR\_PRINT\_MAX\_SIZE << " more)";

}

cout << endl;

}

/// @brief Reads int and performs input validation, asks for input again if needed

/// @return input int

int readIntLine()

{

regex reg\_matchInts = regex(R"(^[+-]?[0-9]+$)");

int result = 0;

bool done = false;

while (!done)

{

try

{

string input;

getline(cin, input);

result = stoi(input);

//regex to block the case when input is smth like "100ffas"

if(!regex\_match(input, reg\_matchInts))

throw invalid\_argument("");

done = true;

}

catch(const invalid\_argument& e)

{

cout << "(The number was in invalid format, enter again)\n";

}

catch(const out\_of\_range& e)

{

cout << "(The number was way too big, enter again)\n";

}

}

return result;

}

bool askWhetherToContinue()

{

string str\_doContinue;

while (str\_doContinue != "y" && str\_doContinue != "Y" && str\_doContinue != "n" && str\_doContinue != "N")

{

//asks whether to execute algo again or not

cout << "Do you want to try again? (y/n)? ";

getline(cin, str\_doContinue);

if (str\_doContinue == "y" || str\_doContinue == "Y")

return true;

}

return false;

}

vector<int> generateRandomArray(int size)

{

vector<int> arr (size);

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

{

// Get a random number

int random = rand() % 10;

arr[i] = random;

}

return arr;

}

/// @brief сортирует подмассив заданный аргументами в виде [elements less than pivot, pivots, elements greater than pivot]

/// @param arr массив

/// @param lowerBound нижний индекс массива

/// @param upperBound верхний индекс массива

/// @return пару {указатель на элемент один правее от {elements less than pivot}, указатель на элемент один левее от {elements greater than pivot}}

pair<int, int> threeWayPartition(vector<int> &arr, int lowerBound, int upperBound)

{

//выбираем первый элемент как pivot

int pivot = arr[lowerBound];

// lessValsPointer - указывает на последний малый элемент

// greaterValsPointer - указывает на последний больший элемент

int lessValsPointer = lowerBound, greaterValsPointer = upperBound;

//i - указывает на текущий элемент который мы рассматриваем

int i = lessValsPointer;

if (DEBUG)

{

cout << "-------------partition run--------------------" << endl;

cout << "pivot: " << pivot << endl;

cout << "(lower pointer: " << lowerBound << ", upper pointer: " << upperBound << ")" << endl;

}

//Пока итерируемся, arr[i] может быть одним из трех:

//1. равно pivot - пропускаем

//2. меньше pivot - переносим за черту lessValsPointer, за пропущенные pivot

//3. больше pivot - переносим за черту greaterValsPointer, перед пропущенными pivot и еще не пройденными arr[i]

//

// что то такое посреди работы цикла:

// lessValsPointer i greaterValsPointer

//[-1, -5, -9, | 4, 4, 4, 4, 4, | 1, 4, 9, 0, 8, 3 |, 12, 41, 49, 50 ]

//

// что то такое после работы цикла:

// lessValsPointer greaterValsPointer

//[-1, -5, -9, | 4, 4, 4, 4, 4 |, 12, 41, 49, 50 ]

// если i > greaterValsPointer - непройденные arr[i] схлопнулись

while (i <= greaterValsPointer) {

if (DEBUG) printArray(arr);

if (arr[i] < pivot)

swap(arr[i++], arr[lessValsPointer++]);

else if (arr[i] > pivot)

swap(arr[i], arr[greaterValsPointer--]);

else

i++;

}

if (DEBUG)

{

printArray(arr);

cout << "-----------------------------------------------" << endl;

}

return {lessValsPointer, greaterValsPointer};

}

/// @brief performs three-way quick sort in-place on arr

/// @return input int

void quickSort(vector<int> &arr, int lowerBound, int upperBound)

{

//если верхний указатель достиг нижнего, ничего сортировать не осталось

//(размер массива для сортировки равен нулю)

if (lowerBound >= upperBound)

return;

pair<int, int> bounds = threeWayPartition(arr, lowerBound, upperBound);

int lowerPointer = bounds.first;

int upperPointer = bounds.second;

//сортируем значения, меньшие прошлого pivot

quickSort(arr, lowerBound, lowerPointer - 1);

//сортируем значения, большие прошлого pivot

quickSort(arr, upperPointer + 1, upperBound);

}

/// @brief Count elements of negative and positive integers in a sequence

int main()

{

//loop for repeating algo if user needs it

bool doContinue = false;

do

{

doContinue = false;

cout << "Enter the number of elements:\n";

int size = -1;

while (size < 1)

{

size = readIntLine();

if (size < 1)

cout << "(The number has to be greater than 0)" << endl;

}

// Providing a seed value for random numbers

srand((unsigned) time(NULL));

vector<int> arr = generateRandomArray(size);

cout << "Generated random array: " << endl;

printArray(arr);

cout << "Sorted array: " << endl;

quickSort(arr, 0, size-1);

printArray(arr);

doContinue = askWhetherToContinue();

} while (doContinue);

return 0;

}