**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Сортировки

|  |  |  |
| --- | --- | --- |
| Студент гр. 9382 |  | Докукин В.М. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Изучить различные виды сортировок, принципы их работы, а также оценки сложности алгоритмов сортировки; научиться реализовывать сортировки на языке C++. Изучить шаблоны, их использование и применение для решения задач на языке С++.

**Основные теоретические положения.**

Сортировкой называется упорядочение элементов множества согласно некоторому правилу. Одной из самых эффективных сортировок на сегодняшний день является так называемая **быстрая сортировка**, или **сортировка Хоара**. Принцип работы сортировки заключается в определении места в массиве, которое должен занимать опорный элемент по окончании сортировки, и рекурсивном вызове алгоритма для обработки частей массива слева и справа от опорного элемента.

Оценка сложности алгоритма быстрой сортировки в худшем случае составляет *О*(n2), но в среднем случае, который на практике встречается гораздо чаще, сложность составляет *O(*nlogn), что и делает сортировку одной из наиболее быстрых.

**Задание.**

9. Быстрая сортировка, рекурсивная реализация. Во время сортировки массив должен быть в состоянии: элементы <x, элементы >=x, неотсортированные элементы.

**Ход работы.**

1. Написана шаблонная функция template <class T> void swap(T& a, T& b), меняющая местами два элемента типа T.

2. Написана шаблонная функция template <class T> int partition(T\* A, int left, int right), принимающая указатель на массив элементов типа T и индексы левой и правой границ подмассива, который нужно обработать. Функция итеративно разбивает элементы на 3 множества согласно условию задания и возвращает индекс опорного элемента после обработки массива.

3. Написана шаблонная функция template <class T> void quickSort(T& A, int left, int right) – рекурсивная функция, использующая функцию partition() для приведения массива к требуемому в задаче виду, а затем вызывающая саму себя для сортировки подмассивов слева и справа от опорного элемента, индекс которого возвращается функцией partition().

4. Написана функция int func(), считывающая выбор режима ввода, выбор типа вводимых данных, размер массива вводимых данных и сами данные, а также обрабатывающая массив введённых данных функцией quickSort() и выводящая на экран отсортированный массив.

5. В функции main() происходит вызов func() и отлов возможных ошибок при помощи блока try-catch.

**Пример работы программы.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5  0  -4 7 6 3 -12 | Array:  -4 7 6 3 -12  Bounds: left - 0, right - 4  Pivot element: -4  pivot = 0, i = 1  |-4 7| 6 3 -12  pivot = 0, i = 2  |-4 7 6| 3 -12  pivot = 0, i = 3  |-4 7 6 3| -12  pivot = 1, i = 4  -12 |-4 6 3 7|  Bounds: left - 2, right - 4  Pivot element: 6  pivot = 3, i = 3  3 |6| 7  pivot = 3, i = 4  3 |6 7|  Sorted array: -12 -4 3 6 7 |

**Тестирование.**

Результаты тестирования представлены в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Входные данные | Выходные данные | Комментарий |
| 1 | 10  0  -7 5 3 17 6 0 17 17 -17 5 | Array:  -7 5 3 17 6 0 17 17 -17 5  Bounds: left - 0, right - 9  Pivot element: -7  pivot = 0, i = 1  |-7 5| 3 17 6 0 17 17 -17 5  pivot = 0, i = 2  |-7 5 3| 17 6 0 17 17 -17 5  pivot = 0, i = 3  |-7 5 3 17| 6 0 17 17 -17 5  pivot = 0, i = 4  |-7 5 3 17 6| 0 17 17 -17 5  pivot = 0, i = 5  |-7 5 3 17 6 0| 17 17 -17 5  pivot = 0, i = 6  |-7 5 3 17 6 0 17| 17 -17 5  pivot = 0, i = 7  |-7 5 3 17 6 0 17 17| -17 5  pivot = 1, i = 8  -17 |-7 3 17 6 0 17 17 5| 5  pivot = 1, i = 9  -17 |-7 3 17 6 0 17 17 5 5|  Bounds: left - 2, right - 9  Pivot element: 3  pivot = 2, i = 3  |3 17| 6 0 17 17 5 5  pivot = 2, i = 4  |3 17 6| 0 17 17 5 5  pivot = 3, i = 5  0 |3 6 17| 17 17 5 5  pivot = 3, i = 6  0 |3 6 17 17| 17 5 5  pivot = 3, i = 7  0 |3 6 17 17 17| 5 5  pivot = 3, i = 8  0 |3 6 17 17 17 5| 5  pivot = 3, i = 9  0 |3 6 17 17 17 5 5|  Bounds: left - 4, right - 9  Pivot element: 6  pivot = 4, i = 5  |6 17| 17 17 5 5  pivot = 4, i = 6  |6 17 17| 17 5 5  pivot = 4, i = 7  |6 17 17 17| 5 5  pivot = 5, i = 8  5 |6 17 17 17| 5  pivot = 6, i = 9  5 5 |6 17 17 17|  Bounds: left - 4, right - 5  Pivot element: 5  pivot = 4, i = 5  |5 5|  Bounds: left - 7, right - 9  Pivot element: 17  pivot = 7, i = 8  |17 17| 17  pivot = 7, i = 9  |17 17 17|  Bounds: left - 8, right - 9  Pivot element: 17  pivot = 8, i = 9  |17 17|  Sorted array: -17 -7 0 3 5 5 6 17 17 17 | Проверка на массиве с повторяющимися значениями. |
| 2 | 10  1  t e h r i l 1 7 z a | Array:  t e h r i l 1 7 z a  Bounds: left - 0, right - 9  Pivot element: t  pivot = 1, i = 1  e |t| h r i l 1 7 z a  pivot = 2, i = 2  e h |t| r i l 1 7 z a  pivot = 3, i = 3  e h r |t| i l 1 7 z a  pivot = 4, i = 4  e h r i |t| l 1 7 z a  pivot = 5, i = 5  e h r i l |t| 1 7 z a  pivot = 6, i = 6  e h r i l 1 |t| 7 z a  pivot = 7, i = 7  e h r i l 1 7 |t| z a  pivot = 7, i = 8  e h r i l 1 7 |t z| a  pivot = 8, i = 9  e h r i l 1 7 a |t z|  Bounds: left - 0, right - 7  Pivot element: e  pivot = 0, i = 1  |e h| r i l 1 7 a  pivot = 0, i = 2  |e h r| i l 1 7 a  pivot = 0, i = 3  |e h r i| l 1 7 a  pivot = 0, i = 4  |e h r i l| 1 7 a  pivot = 1, i = 5  1 |e r i l h| 7 a  pivot = 2, i = 6  1 7 |e i l h r| a  pivot = 3, i = 7  1 7 a |e l h r i|  Bounds: left - 0, right - 2  Pivot element: 1  pivot = 0, i = 1  |1 7| a  pivot = 0, i = 2  |1 7 a|  Bounds: left - 1, right - 2  Pivot element: 7  pivot = 1, i = 2  |7 a|  Bounds: left - 4, right - 7  Pivot element: l  pivot = 5, i = 5  h |l| r i  pivot = 5, i = 6  h |l r| i  pivot = 6, i = 7  h i |l r|  Bounds: left - 4, right - 5  Pivot element: h  pivot = 4, i = 5  |h i|  Sorted array: 1 7 a e h i l r t z | Проверка на массиве символов. |
| 3 | 8  2  5.5 -32.7 32.2 -2.28 13.037 6 7 8 | Array:  5.5 -32.7 32.2 -2.28 13.037 6 7 8  Bounds: left - 0, right - 7  Pivot element: 5.5  pivot = 1, i = 1  -32.7 |5| 32.2 -2.28 13.037 6 7 8  pivot = 1, i = 2  -32.7 |5 32.2| -2.28 13.037 6 7 8  pivot = 2, i = 3  -32.7 -2.28 |5 32| 13.037 6 7 8  pivot = 2, i = 4  -32.7 -2.28 |5 32 13.037| 6 7 8  pivot = 2, i = 5  -32.7 -2.28 |5 32 13.037 6| 7 8  pivot = 2, i = 6  -32.7 -2.28 |5 32 13.037 6 7| 8  pivot = 2, i = 7  -32.7 -2.28 |5 32 13.037 6 7 8|  Bounds: left - 0, right - 1  Pivot element: -32.7  pivot = 0, i = 1  |-32.7 -2.28|  Bounds: left - 3, right - 7  Pivot element: 32  pivot = 4, i = 4  13.037 |32| 6 7 8  pivot = 5, i = 5  13.037 6 |32| 7 8  pivot = 6, i = 6  13.037 6 7 |32| 8  pivot = 7, i = 7  13.037 6 7 8 |32|  Bounds: left - 3, right - 6  Pivot element: 13.037  pivot = 4, i = 4  6 |13| 7 8  pivot = 5, i = 5  6 7 |13| 8  pivot = 6, i = 6  6 7 8 |13|  Bounds: left - 3, right - 5  Pivot element: 6  pivot = 3, i = 4  |6 7| 8  pivot = 3, i = 5  |6 7 8|  Bounds: left - 4, right - 5  Pivot element: 7  pivot = 4, i = 5  |7 8|  Sorted array: -32.7 -2.28 5 6 7 8 13 32 | Проверка на массиве чисел с плавающей запятой одинарной точности (float). |

**Выводы.**

В результате выполнения лабораторной работы:

1. Были изучены различные сортировки, принципы их работы, оценки сложности.

2. Были изучены шаблоны в языке С++, способы их применения.

3. Была написана программа, решающая поставленную задачу.

4. Была написана серия тестов, позволяющих качественно оценить работу программы (тесты находятся в файле tests.txt).

Код программы размещён в Приложении 1.

ПРИЛОЖЕНИЕ 1

ИСХОДНЫЙ КОД ПРОГРАММЫ

Имя файла: main.cpp

#include <iostream>

#include <string>

#include <fstream>

#include <typeinfo>

#define EPS 0.0000005

template<class T>

void swap(T& a, T& b){

int tmp = a;

a = b;

b = tmp;

}

template<class T>

int partition(T\* A, int left, int right){

int pivot = left; //Опорным элементом выбираем первый

std::cout<<"Bounds: left - "<<left<<", right - "<<right<<"\nPivot element: "<<A[pivot]<<'\n';

for (int i = left + 1; i <= right; i++){

if (A[i] < A[pivot]){ //Если элемент меньше опорного - меняем его местами с первым элементом больше опорного, а затем меняем его же местами с самим опорным (для соблюдения условий задачи)

if (pivot + 1 != i) swap(A[pivot+1], A[i]);

swap(A[pivot], A[pivot+1]);

pivot++;

}

std::cout<<"pivot = "<<pivot<<", i = "<<i<<'\n';

for(int j = left; j <= right; j++){ //Каждый шаг выводим текущее состояние массива на экран

if (j == pivot) std::cout<<'|';

std::cout<<A[j];

if (j == i) std::cout<<'|';

std::cout<<" ";

}

std::cout<<'\n';

}

return pivot;

}

template <class T>

void quickSort(T\* A, int left, int right){

if (left < right){

int q = partition(A, left, right);

quickSort(A, left, q-1);

quickSort(A, q+1, right);

}

}

int func(){ //Выбираем опцию ввода, вводим количество элементов в массиве и сами элементы или вводим из файла

int a;

int elemmode;

std::cout<<"Choose input option(0 - file input, 1 - console input):\n";

std::cin>>a;

int n;

if(a){

std::cout<<"Choose array elements type(<=0 - int, 1 - char, >=2 - float):";

std::cin>>elemmode;

std::cout<<"Insert array size:\n";

std::cin>>n;

std::cout<<"Insert array elements:\n";

if (elemmode <= 0){

int\* B;

B = new int[n];

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

std::cin>>B[i];

}

quickSort(B, 0, n - 1);

std::cout<<"Sorted array: ";

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

std::cout<<B[i]<<" ";

}

delete[] B;

}

else if (elemmode == 1){

char\* B;

B = new char[n];

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

std::cin>>B[i];

}

quickSort(B, 0, n - 1);

std::cout<<"Sorted array: ";

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

std::cout<<B[i]<<" ";

}

delete[] B;

}

else{

float\* B;

B = new float[n];

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

std::cin>>B[i];

}

quickSort(B, 0, n - 1);

std::cout<<"Sorted array: ";

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

std::cout<<B[i]<<" ";

}

delete[] B;

}

std::cout<<"\n--------------------------------\n";

return 0;

}

std::ifstream f("tests.txt");

if (!f){

std::cout<<"Couldn't open file.\n";

}

while(!f.eof()){

f>>n;

f>>elemmode;

if (elemmode <= 0){

int\* B;

B = new int[n];

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

f>>B[i];

}

std::cout<<"Array:\n";

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

std::cout<<B[i]<<' ';

}

std::cout<<'\n';

quickSort(B, 0, n - 1);

std::cout<<"Sorted array: ";

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

std::cout<<B[i]<<" ";

}

delete[] B;

}

else if (elemmode == 1){

char\* B;

B = new char[n];

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

f>>B[i];

}

std::cout<<"Array:\n";

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

std::cout<<B[i]<<' ';

}

std::cout<<'\n';

quickSort(B, 0, n - 1);

std::cout<<"Sorted array: ";

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

std::cout<<B[i]<<" ";

}

delete[] B;

}

else{

float\* B;

B = new float[n];

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

f>>B[i];

}

std::cout<<"Array:\n";

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

std::cout<<B[i]<<' ';

}

std::cout<<'\n';

quickSort(B, 0, n - 1);

std::cout<<"Sorted array: ";

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

std::cout<<B[i]<<" ";

}

delete[] B;

}

std::cout<<"\n--------------------------------\n";

}

return 0;

}

int main(){

try{

func();

}

catch(...){

std::cout<<"An unexpected error occurred.\n";

}

return 0;

}