Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №2**

**«ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Калинин Максим Константинович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы:** получить базовые сведения о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами.

**Вариант 11**

**Задача 1**

Реализовать сортировку данных с помощью алгоритма подсчетом.

**Алгоритм решения**



Рис.1 - Алгоритм задания №1

**Описание алгоритма**

1. Открывается файл input1.txt для чтения входных данных.
2. Создается файл output1.txt для сохранения отсортированных данных.
3. Создается массив data для хранения данных из файла.
4. Инициализируется переменная num для временного хранения считываемых чисел.
5. Инициализируется переменная i для отслеживания индекса в массиве data.
6. Числа считываются из файла input1.txt.
7. Каждое считанное число добавляется в массив data.
8. Счетчик i увеличивается для управления индексами массива data.
9. Вызывается функция cntsrt, передавая ей массив data, его размер и функцию сравнения comrawrar.
10. Файл output1.txt открывается для записи отсортированных чисел.
11. Отсортированные числа из массива data записываются в файл output1.txt.
12. Оба файла (input1.txt и output1.txt) закрываются.
13. Выводится сообщение о завершении сортировки (output1.txt).

**Код решения программы**

**type**

comparator = **function**(a, b: integer): boolean;

**procedure** cntsrt(**var** rawr: **array of** integer; sz: integer; comparer: comparator);

**var**

minv, maxv, i, j, ii, temp: integer;

cnt: **array of** integer;

**begin**

minv := rawr[0];

maxv := rawr[0];

//минимальное и максимальное значения в массиве

**for** i := 1 **to** sz - 1 **do**

**begin**

**if** rawr[i] < minv **then**

minv := rawr[i];

**if** rawr[i] > maxv **then**

maxv := rawr[i];

**end**;

SetLength(cnt, maxv - minv + 1);

**for** i := 0 **to** maxv - minv **do**

cnt[i] := 0;

**for** i := 0 **to** sz - 1 **do** //подсчет количества встреч каждого элемента

cnt[rawr[i] - minv] := cnt[rawr[i] - minv] + 1;

ii := 0;

**for** i := 0 **to** maxv - minv **do** //отсортированный массив

**for** j := 1 **to** cnt[i] **do**

**begin**

rawr[ii] := i + minv;

Inc(ii);

**end**;

**if not** comparer(1, 2) **then** //по убыванию при выборе

**begin**

**for** i := 0 **to** sz **div** 2 - 1 **do**

**begin**

j := sz - 1 - i;

temp := rawr[i];

rawr[i] := rawr[j];

rawr[j] := temp;

**end**;

**end**;

**end**;

**function** comrawrar(a, b: integer): boolean;

**begin**

//Result := a > b; //для сортировки по убыванию

Result := a < b; //для сортировки по возрастанию

**end**;

**var**

inputFile, outputFile: text;

data: **array of** integer;

num, i: integer;

**begin**

Assign(inputFile, 'input1.txt');

Reset(inputFile);

Assign(outputFile, 'output1.txt');

Rewrite(outputFile);

i := 0;

**while not** Eof(inputFile) **do**

**begin**

read(inputFile, num);

SetLength(data, i + 1);

data[i] := num;

i := i + 1;

**end**;

cntsrt(data, Length(data), comrawrar);

**for** i := 0 **to** Length(data) - 1 **do**

Write(outputFile, data[i], ' ');

Close(inputFile);

Close(outputFile);

writeln('Сортировка завершена. Результат записан в файл output1.txt');

**end**.

**Результат выполнения программы**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 16 16 16 1 16 47 28 98 19 37 45 72 37 463 0 82 | 0 1 16 16 16 16 19 28 37 37 45 47 72 82 98 463 |
| -100 28 219 3871 7 283 9 62 3 | -100 3 7 9 28 62 219 283 3871 |
| 0 127 1 372 83 27 49297 2983 28 19 | 0 1 19 27 28 83 127 372 2983 49297 |
| -2 -1783 281 18 37 63 199 728 74 283 1 | -1783 -2 1 18 37 63 74 199 281 283 728 |
| 29378 1763 61 382 33493 19 -137 193 -3 23 -2 823 -3 | -137 -3 -3 -2 19 23 61 193 382 823 1763 29378 33493 |

**Задача 2**

Реализовать сортировку данных с помощью пирамидального алгоритма.

**Алгоритм решения**



Рис.1 – Алгоритм задания №2

**Описание алгоритма**

1. Открытие файла input1.txt для чтения входных данных.
2. Создание файла output1.txt для записи отсортированных данных.
3. Создание массива data для хранения данных из файла.
4. Инициализация переменной num для временного хранения считываемых чисел.
5. Инициализация переменной i для отслеживания индекса в массиве data.
6. Чтение чисел из файла input1.txt.
7. Добавление каждого считанного числа в массив data.
8. Увеличение счетчика i для управления индексами массива data.
9. Вызов функции trsrt, передавая ей массив data, его размер и функцию сравнения comrawrar.
10. Открытие файла output1.txt для записи отсортированных чисел.
11. Запись отсортированных чисел из массива data в файл output1.txt.
12. Закрытие обоих файлов (input1.txt и output1.txt).
13. Вывод сообщения о завершении сортировки (output1.txt).

**Код решения программы**

**type**

comparator = **function**(a, b: integer): boolean;

**procedure** tr(**var** rawr: **array of** integer; sz, i: integer; comprawr: comparator);

**var**

big, l, r, temp: integer;

**begin**

big := i;

l := 2 \* i + 1;

r := 2 \* i + 2;

**if** (l < sz) **and** comprawr(rawr[l], rawr[big]) **then**

big := l;

**if** (r < sz) **and** comprawr(rawr[r], rawr[big]) **then**

big := r;

**if** big <> i **then**

**begin**

temp := rawr[i];

rawr[i] := rawr[big];

rawr[big] := temp;

tr(rawr, sz, big, comprawr);

**end**;

**end**;

**procedure** trsrt(**var** rawr: **array of** integer; sz: integer; comprawr: comparator);

**var**

i, temp: integer;

**begin**

**for** i := sz **div** 2 - 1 **downto** 0 **do**

tr(rawr, sz, i, comprawr);

**for** i := sz - 1 **downto** 0 **do**

**begin**

temp := rawr[0];

rawr[0] := rawr[i];

rawr[i] := temp;

tr(rawr, i, 0, comprawr);

**end**;

**end**;

**function** comrawrar(a, b: integer): boolean;

**begin**

//Result := a < b; //для сортировки по убыванию

Result := a > b; // для сортировки по возрастанию

**end**;

**var**

inputFile, outputFile: text;

data: **array of** integer;

num, i: integer;

**begin**

Assign(inputFile, 'input1.txt');

Reset(inputFile);

Assign(outputFile, 'output1.txt');

Rewrite(outputFile);

i := 0;

**while not** Eof(inputFile) **do**

**begin**

read(inputFile, num);

SetLength(data, i + 1);

data[i] := num;

i := i + 1;

**end**;

trsrt(data, Length(data), comrawrar);

**for** i := 0 **to** Length(data) - 1 **do**

Write(outputFile, data[i], ' ');

Close(inputFile);

Close(outputFile);

writeln('Сортировка завершена. Результат записан в файл output1.txt');

**end**.

**Результат выполнения программы**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 16 16 16 1 16 47 28 98 19 37 45 72 37 463 0 82 | 0 1 16 16 16 16 19 28 37 37 45 47 72 82 98 463 |
| -100 28 219 3871 7 283 9 62 3 | -100 3 7 9 28 62 219 283 3871 |
| 0 127 1 372 83 27 49297 2983 28 19 | 0 1 19 27 28 83 127 372 2983 49297 |
| -2 -1783 281 18 37 63 199 728 74 283 1 | -1783 -2 1 18 37 63 74 199 281 283 728 |
| 29378 1763 61 382 33493 19 -137 193 -3 23 -2 823 -3 | -137 -3 -3 -2 19 23 61 193 382 823 1763 29378 33493 |

**Вывод:**

1. Освоил алгоритмы сортировки, включая подсчётные и пирамидальные, для эффективной работы с данными.
2. Приобрёл глубокие знания о функциях языка Pascal и их использовании для реализации сортировок.
3. Улучшил навыки работы с массивами, особенно при переводе данных из файла в структурированный массив.
4. Понял важность использования противоположных условий при написании сортировочных алгоритмов для различных порядков сортировки.
5. Освоил быстрый доступ и манипуляции элементами массива с помощью цикла for.
6. Изучил альтернативные способы работы со строками и их реализацию, расширяя спектр доступных функций и методов обработки текстовых данных.
7. Применил различные индексы для одного массива, оптимизируя доступ к отдельным элементам данных.
8. Понял принцип использования второго массива для хранения дополнительных значений при сортировке или обработке данных, что повышает гибкость работы с информацией.
9. Улучшил навыки работы с условиями и их объединением для более сложных операций над данными, что сделало код более читаемым и эффективным.