МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «АиСД»

ТЕМА: Сортировки

Студент гр. 9382 Пя С.

Преподаватель Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться со сортировками. Научиться реализовывать заданную сортировку на примере языка C++, освоить применение сортивок, сравнить правильность выполнения реализованной сортировки и станартной, выявить положительные и отрицательные стороны выбранной сортировки. Выполнить работу в соответствии с заданием.

Основные теоретические положения.

Сортировка — последовательное расположение или разбиение на группы чего-либо в зависимости от выбранного критерия. Алгоритм сортировки (множество сортировок).а) имеется один специально обозначенный узел, называемый корнем данного дерева;

Поразрядная сортировка (англ. radix sort) — алгоритм сортировки, который выполняется за линейное время. Существуют стабильные варианты. Исходно предназначен для сортировки целых чисел, записанных цифрами.

Задание

Вариант №20.

В сложности учитывается не только сложность самогó алгоритма, но и:

- рассматривается ли алгоритм на лекциях;

- нужно ли самостоятельно придумывать алгоритм.

Ниже для некоторых вариантов приведены ссылки на описания алгоритмов под соответствующими номерами.

20. Поразрядная сортировка.

Ход работы.

1) Разработан алгоритм:

На вход подается начало линейного списка и максимальная длина ячейки данных (в нашем случае числа). Циклически проходимся по всем разрядам чисел, попутно рассортировывая их. Для этого в начале мы создали два массива (начала списков и концы списков) списков, за членом каждого из которых закреплена определенная уникальная цифра. Она означает разряд числа. В первом вхождении определяем первый разряд входного числа, а затем кладем его в массив с индексом, совпадающим с разрядом. Если в массиве уже лежит число, то кладем текущее за ним. Так у нас получается упорядоченность по первому

разряду. Далее соединяем концы списков предыдущих разрядов и начала следующих для того, чтобы получить наш упорядоченный по определенному разряду список и дальше работать с ним. Так как до этого мы сортируем данные по младшим разрядам, то в массивы более старших разрядов они попадают уже упорядоченными. После сортировки по последнему разряду выводится уже отсортированный массив. Были устранены все утечки памяти, написана реакция на не открытие файла и на некорректные данные.

Предусмотрен механизм простейшего взаимодействия с пользователем, позволяющий понять алгоритм исполнения программы, с помощью вывода сообщений. Также был предусмотрен ввод данных с клавиатуры.

Преимущества и недостатки алгоритма: является более подходящим для применения в компьютерной графике. Поразрядная сортировка растет линейным образом по n(количество данных), так как k (максимальное количество разрядов в ячейке данных), m(разрядность данных, то есть количество членов массива) – константы. Данная сортировка не очень подходит для маленького количества данных. Поразрядная сортировка является устойчивой. Выигрывает по эффективности стандартную сортировку только после около 500000 данных.

2) Использована структура:

node

Назначение: используется для создания линейного списка данных.

Описание содержимого: val : long int – число, которое обрабатывается; next : struct node – указатель на структуру, служащий для создания линейного списка. Также написаны конструктор и деструктор структуры.

Использованы функции:

1. printArray

Сигнатура: void printArray(Node* shift).

Описание аргументов: shift : Node – указатель на начало линейного списка.

Назначение: выводит линейный список на экран.

2. deleteArray

Сигнатура: void deleteArray(Node* shift).

Назначение: освобождает память, выделенную для линейного списка.

Описание аргументов: shift : Node – указатель на начало линейного списка.

3. radixSort

Сигнатура: Node* radixSort(Node *current, int length).

Назначение: сортирует линейный список поразрядно.

Описание аргументов: current : Node - указатель на начало линейного списка, length : int – максимальное количество разрядов числа.

Возвращаемое значение: указатель на отсортированный линейный список.

4. fcn

Сигнатура: void fcn(std::istream &fin).

Назначение: позволяет работать универсально для разных потоков ввода, вызывает разные сортировки, вызывает вывод списков и удаление памяти.

Описание аргументов: fin : istream – поток ввода.

Реализация сортирующего метода:

На вход подаются указатель на начало линейного списка и максимальная длина ячейки данных. Реализуется цикл для прогона по всем разрядам. Еще один цикл для записи данных по определенному разряду в массив. Далее концы списков-членов массива соединяются с началами следующих членов для дальнейшей работы со связным линейным списком.

Пример работы программы.

Входные данные	Выходные данные

3 10 3 654 234 765 345 345 234 654 123 192 65	What do you prefer? (0 - terminal, n - number of file) 3 Nice job! Then my turn to work.. Demonstration of radix sort work: This is sorting by category of 1 192 123 654 234 234 654 765 345 345 65 This is sorting by category of 10 123 234 234 345 345 654 654 765 65 192 This is sorting by category of 100 65 123 192 234 234 345 345 654 654 765 Demonstration of result: 65 123 192 234 234 345 345 654 654 765 Demonstration of standard sort work: Demonstration of result: 65 123 192 234 234 345 345 654 654 765

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 6 2 24 56 4 3 4 23	What do you prefer? (0 - terminal, n - number of file) 0 Enter count of array, max length of one data, then data: 6 2 24 56 4 3 4 23 Nice job! Then my turn to work.. Demonstration of radix sort work: This is sorting by category of 1 3 23 24 4 4 56 This is sorting by category of 10 3 4 4 23 24 56 Demonstration of result: 3 4 4 23 24 56 Demonstration of standard sort work: Demonstration of result: 3 4 4 23 24 56	Проверка на корректность работы программы с терминалом

		Process finished with exit code 0	
2.	1 5 1 5 2 4 3 1	What do you prefer? (0 - terminal, n - number of file) 1 Nice job! Then my turn to work.. Demonstration of radix sort work: This is sorting by category of 1 1 2 3 4 5 Demonstration of result: 1 2 3 4 5 Demonstration of standard sort work: Demonstration of result: 1 2 3 4 5	Проверка на корректность работы с файлом
4.	2 9	What do you prefer? (0 - terminal, n - number of file) 2 Error data!	Проверка на корректность работы с полупустыми данными
5.	4 6 2 345 234 4 3 4 22	What do you prefer? (0 - terminal, n - number of file) 4 Nice job! Then my turn to work.. Demonstration of radix sort work: This is sorting by category of 1 22 3 234 4 4 345 This is sorting by category of 10 3 4 4 22 234 345 Demonstration of result: 3 4 4 22 234 345 Demonstration of standard sort work: Demonstration of result: 3 4 4 22 234 345	Проверка на корректность работы

Выводы.

В ходе работы была освоена реализация поразрядной сортировки на основе линейного списка, отработано понимание его применения, и отработаны навыки

письма в C++. Также корректность работы была прослежена с помощью стандартной сортировки.

Код программы можно найти в приложении А.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <list>

typedef struct node {//структура для создания односвязных списков
    long int val;//данные узла
    struct node* next;//указатель на следующий узел
    node() {//конструктор структуры
        val = 0;
        next = nullptr;
    }
    ~node() {//деструктор
        next = nullptr;
    }
} Node;

void printArray(Node* shift) {//функция для вывода отсортировонного списка
данных
    while (shift->next != nullptr) {
        std::cout << shift->val << " ";
        shift = shift->next;
    }
    std::cout << shift->val << "\n";
}

void deleteArray(Node* shift) {//освобождение памяти
    while (shift->next != nullptr) {
       Node* t = shift;
       shift = shift->next;
       delete t;
    }
    delete shift;
}
//функция для сортировки данных с помощью поразрядной сортировки
Node* radixSort(Node *current, int length) {//length - максимальная длина одной
ячейки данных
    Node *shift, *head[10], *tail[10];//shift - указатель, служащий временным
хранилищем, head - начало списка данных в определенном списке разрядности, а
tail - конец соответственно
    int i, y, n, d = 1;//i, y - счетчики, n - переменная для хранения текущего
разряда числа, d - переменная, нужная для выделения определенного разряда
    for (y = 0; y < length; y++) {//цикл для прохождения всех разрядов
        for (i = 0; i <= 9; i++)//цикл для обнуления указателя начала и конца
массивов списков
            head[i] = (tail[i] = NULL);
        while (current != NULL) {//проход по всем данным
```

```
            n = (current->val / d) % 10;//получение разряда
            shift = tail[n];
            if (head[n] == NULL)//если данных, разряд которых равен, несколько,
то они присоединяются к head
                head[n] = current;
            else
                shift->next = current;
            shift = tail[n] = current;//в tail только последняя структура данных
            current = current->next;
            shift->next = NULL;
        }
        for (i = 0; i <= 9; i++)//определяем начало данных для следующего витка
цикла
            if (head[i] != NULL)
                break;
        current = head[i];
        shift = tail[i];
        for (n = i + 1; n <= 9; n++) {//связываем начала и концы списков
последовательно
            if (head[n] != NULL) {
                shift->next = head[n];
                shift = tail[n];
            }
        }
        std::cout << "This is sorting by category of " << d << "\n";
        printArray(current);
        d *= 10;
    }
    return current;
}

void fcn(std::istream &fin) {//функция для универсальной работы с потоком ввода
    int n = 0, length = 0;//n - количество данных, length - максимальная длина
ячейки данных
    fin >> n >> length;
    Node* head;
    std::list <int> nodes;//используется для стандартной сортировки
    if (!(n == 0 || length == 0)) {
        Node* shift = new Node();
        head = shift;
        for (int i = 0; i < n;i++) {//создание списка данных
            fin >> shift->val;
            nodes.push_back(shift->val);
            if (i != n -1)
                shift->next = new Node();
            shift = shift->next;
        }
        std::cout << "Nice job! Then my turn to work..\n";
        std::cout << "Demonstration of radix sort work:\n";
        head = radixSort(head, length);//поразрядная сортировка
        std::cout << "Demonstration of result:\n";
        printArray(head);//вывод отсортированных данных
        deleteArray(head);
        std::cout << "Demonstration of standard sort work:\n";
        nodes.sort();//стандартная сортировка для list
        std::list <int> :: iterator it;
        std::cout << "Demonstration of result:\n";
        for ( it = nodes.begin(); it != nodes.end(); it++) {//вывод
отсортированных данных
            std::cout << (*it) << " ";
        }
    } else std::cout << "Error data!";
}
```

```
int main() {
    std::cout << "What do you prefer? (0 - terminal, n - number of file)\n";
    char n;//выбор источника входных данных
    std::cin >> n;
    if (n == 48) {//ввходные данные из терминала
        std::cout << "Enter count of array, max length of one data, then
data:\n";
        fcn(std::cin);
    } else {
        char *filename = new char[30]();//ввходные данные из файла
        strcat(filename, "Tests//");
        strncat(filename, &n, 1);
        strcat(filename, ".txt");
        std::ifstream in(filename);
        if (!in.is_open()) {
            std::cout << "File wasn't opened!";
            return 0;
        }
        fcn(in);
    }
    return 0;
}
```