|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 5 | | |
| по дисциплине «Информационные технологии и основы программирования» | | |
| **Управление таблицами** | | |
|  | | |
|  | Бригада | ФИО |
| №2 | Веселый Денис Андреевич |
|  | Ворончук Илья Игоревич |
| Группа | ПМИ-32 |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1) Цель работы**

Получить практические навыки организации таблиц, обработки таблиц и их использования при решении задач.

**2) Условие задачи**

* + 1. Элемент таблицы характеризуется двумя признаками: один – это последовательность не более чем из 15 символов и этот признак – ключ, другой признак – вещественное значение. Определить тип «таблица» и написать подпрограмму упорядочения таблицы методом прямого выборами. Подсчитать число сравнений и перемещений, выполненных в процессе упорядочения.
    2. Элемент таблицы характеризуется тремя признаками: один –вещественное значение, второй – целое положительное число и этот признак – ключ элемента, третий – это последовательность не более чем из 10 символов. Определить тип «таблица» и написать подпрограмму упорядочения таблицы методом прямого включения и подсчитать число просмотров и перемещений, выполненных в процессе упорядочения.

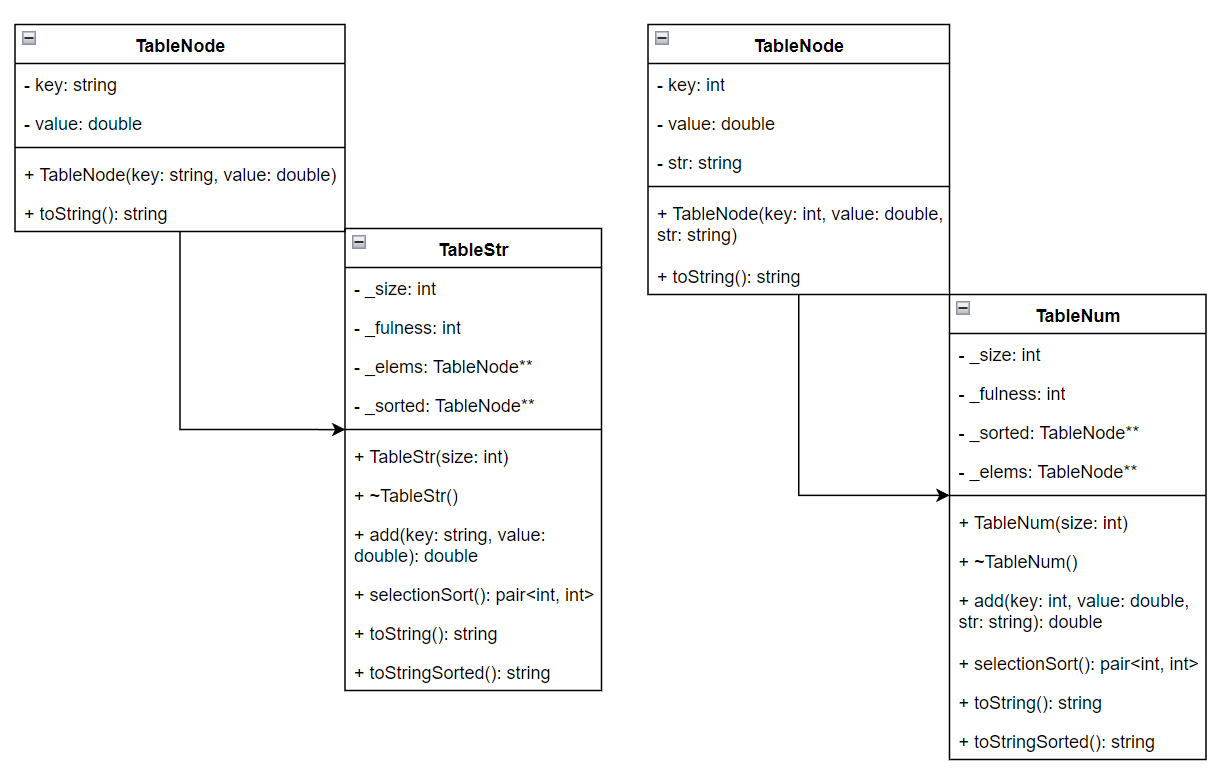
**2.1) Алгоритм решения задачи**

* + 1. Описание алгоритма сортировки выбором.

Выбирается i-й элемент такой, что i = [0, n] и назначается минимальный элемент min. Начиная с j = i+1 элемента, происходит обход элементов до n-го: если j-й элемент оказался меньше min, то минимальным элементом становится j-й элемент и назначается переменная k = j, прибавление единицы к числу сравнений. По окончании цикла, вместо k-го элемента помещается i-й, а min элемент назначается на позицию i-го и увеличивается число перестановок на единицу.

* + 1. Описание алгоритма сортировка включением.

Выбирается i-й элемент такой, что i = [1, n]. Начиная с j = i-1 элемента, происходит обход элементов, j уменьшается до 0: если оказалось, что j-й элемент больше i-го, то j-й элемент “проталкивается” на позицию j+1, происходит инкремент числа сравнений и перестановок. По окончании цикла, на позицию j+1 помещается i-й элемент, число перестановок увеличивается на один.

**3) Стандартная схема программы**

**4) Код программы**

**TableStr.cpp**

TableStr::TableNode::TableNode(std::string key, double value) : key{key}, value{value} {}

std::string TableStr::TableNode::toString() {

return key + ": " + std::to\_string(value);

}

TableStr::TableStr(int size) : \_size{size}, \_fulness{0} {

\_elems = new TableStr::TableNode \*[size];

for (int i = 0; i < \_size; i++)

\_elems[i] = nullptr;

}

TableStr::~TableStr() {

delete[] \_elems;

\_elems = nullptr;

}

// получение хэша по ключу, где h – флаг рехеширования

int TableStr::getHash(std::string key, bool h) {

int hash = 0;

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

hash = (hash << 5) - hash + int(pow(-1, h)) \* key[i];

return abs(hash);

}

// получение хэша по ключу, с учетом номера исследования

int TableStr::getIndex(std::string key, int research) {

return abs(research == 0

? getHash(key, 0) % \_size

: (getHash(key, 0) + research \* getHash(key, 1)) % \_size);

}

// добавление нового элемента в таблицу

double TableStr::add(std::string key, double value) {

int index = getIndex(key, 0);

for (int i = 1; i < \_size; i++) {

if (!\_elems[index]) {

\_elems[index] = new TableNode(key, value);

\_fulness++;

return value;

}

index = getIndex(key, i);

}

return false;

}

// создание массива существующих элементов в таблице

TableStr::TableNode \*\*TableStr::fromTableToArray() {

TableNode \*\*array = new TableNode \*[\_fulness];

for (int i = 0, k = 0; i < \_size; i++) {

if (\_elems[i]) {

array[k] = \_elems[i];

k++;

}

}

return array;

}

// сортировка выбором, возвращает количество сравнений и перестановок

std::pair<int, int> TableStr::selectionSort() {

if (\_fulness == 0) return std::pair(0, 0);

\_sorted = fromTableToArray();

int cmpN = 0, swpN = 0, k;

TableNode \*min;

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

min = \_sorted[i];

k = i;

for (int j = i + 1; j < \_fulness; j++) {

cmpN++;

if (\_sorted[j]->key < min->key) || (\_sorted[j]->key == min->key && \_sorted[j]->value < min->value)) {

min = \_sorted[j];

k = j;

}

}

if (min != \_sorted[i])

swpN++;

\_sorted[k] = \_sorted[i];

\_sorted[i] = min;

}

return std::pair(cmpN, swpN);

}

// преобразование таблицы в строку

std::string TableStr::toString() {

std::string result;

for (int i = 0; i < \_size; i++)

if (\_elems[i])

result += \_elems[i]->toString() + "\n";

return result;

}

// преобразование отсортированной части таблицы в строку

std::string TableStr::toStringSorted() {

std::string result;

for (int i = 0; i < \_fulness; i++)

if (\_sorted[i])

result += \_sorted[i]->toString() + "\n";

return result;

}

**TableNum.cpp**

TableNum::TableNode::TableNode(int key, double value, std::string str) : key{key}, value{value}, str{str} {}

std::string TableNum::TableNode::toString() {

return std::to\_string(key) + ": [" + std::to\_string(value) + ", " + str + "]";

}

TableNum::TableNum(int size) : \_size{size}, \_fulness{0} {

\_elems = new TableNum::TableNode \*[size];

for (int i = 0; i < \_size; i++)

\_elems[i] = nullptr;

}

TableNum::~TableNum() {

delete[] \_elems;

\_elems = nullptr;

}

// получение хэша по ключу, где h – флаг рехеширования

int TableNum::getHash(int key, bool h) {

int hash = (key << 5) + pow(-1, h) \* key;

return abs(hash);

}

// получение хэша по ключу, с учетом номера исследования

int TableNum::getIndex(int key, int research) {

return abs(research == 0

? getHash(key, 0) % \_size

: (getHash(key, 0) + research \* getHash(key, 1)) % \_size);

}

// добавление элемента в таблицу

double TableNum::add(int key, double value, std::string str) {

int index = getIndex(key, 0);

for (int i = 1; i < \_size; i++) {

if (!\_elems[index]) {

\_elems[index] = new TableNode(key, value, str);

\_fulness++;

return value;

}

index = getIndex(key, i);

}

return false;

}

// создание массива существующих элементов в таблице

TableNum::TableNode \*\*TableNum::fromTableToArray() {

TableNode \*\*array = new TableNode \*[\_fulness];

for (int i = 0, k = 0; i < \_size; i++) {

if (\_elems[i]) {

array[k] = \_elems[i];

k++;

}

}

return array;

}

// сортировка вставками, возвращает количество сравнений и перестановок

std::pair<int, int> TableNum::insertionSort() {

if (\_fulness == 0) return std::pair(0, 0);

\_sorted = fromTableToArray();

int cmpN = 0, swpN = 0;

TableNode \*elem;

for (int i = 1; i < \_fulness; i++) {

elem = \_sorted[i];

int j = i - 1;

while (j >= 0 && ((\_sorted[j]->key > elem->key) || (\_sorted[j]->key == elem->key && \_sorted[j]->value > elem->value) || (\_sorted[j]->value == elem->value && \_sorted[j]->str > elem->str))) {

cmpN++;

\_sorted[j + 1] = \_sorted[j];

j--;

}

if (\_sorted[j + 1] != elem)

swpN++;

\_sorted[j + 1] = elem;

}

return std::pair(cmpN, swpN);

}

// преобразование таблицы в строку

std::string TableNum::toString() {

std::string result;

for (int i = 0; i < \_size; i++)

if (\_elems[i])

result += \_elems[i]->toString() + "\n";

return result;

}

// преобразование отсортированной части таблицы в строку

std::string TableNum::toStringSorted() {

std::string result;

for (int i = 0; i < \_fulness; i++)

if (\_sorted[i])

result += \_sorted[i]->toString() + "\n";

return result;

}

**5) Тестирование программы**

|  |  |  |
| --- | --- | --- |
| № | Команды | Комментарий |
| 1 | add eloy 80  add cream 90.14  add jimihendrix 50  add moodyblues 600.8  prts --sorted | 1. Добавление элементов вида (ключ, значение) в таблицу. 2. Вывод отсортированной коллекции в консоль.   Число сравнений: 6, число перестановок: 2. |
| 2 | add --num 4 10 traffic  add --num 9 1 yes  add --num 400 11 greenday  add --num 8 3 the5:55  prtn --sorted | 1. Добавление элементов вида (ключ, значение, строка) в таблицу. 2. Вывод отсортированной коллекции в консоль.   Число сравнений: 4, число перестановок: 2. |
| 3 | add cream 89  add cream 777  add cream 41.3  add cream 12  add cream 12  prts --sorted | 1. Добавление элементов вида (ключ, значение) в таблицу. Все элементы имеют одинаковый ключ. Два из них – одинаковые. 2. Вывод отсортированной коллекции в консоль.   Число сравнений: 10, число перестановок: 2. |

**6) Результаты отладки**

Результаты работы программы показали, что программа работает успешно.