Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

Специальность прикладная математика и информатика

Отчёт по лабораторной работе

Таблицы

Выполнил:

Студент ПМИгр. 381703-1

Арефьев А. М.

Проверил:

Ассистент кафедры МОСТ ИИТММ

Волокитин В. Д.

Нижний Новгород

2019 г.

Оглавление

[Введение 3](#_Toc499325868)

[Постановка задачи 4](#_Toc499325869)

[Руководство пользователя 5](#_Toc499325870)

[Руководство программиста 6](#_Toc499325871)

[Описание структур данных 6](#_Toc499325872)

[Описание алгоритмов 6](#_Toc499325873)

[Описание структуры программы 7](#_Toc499325874)

[Заключение 7](#_Toc499325875)

[Литература 8](#_Toc499325876)

[Приложения 9](#_Toc499325877)

# Введение

С физической точки зрения таблица представляет собой вектор, элементами которого являются записи. Характерной логической особенностью таблиц, которая определяет их отдельное рассмотрение, является то, что доступ к элементам таблицы производится не по номеру (индексу), а по ключу - по значению одного из свойств объекта, описываемого элементом таблицы. Ключ - это свойство, идентифицирующее данную запись во множестве однотипных записей. Как правило, к ключу предъявляется требование уникальности в данной таблице. Ключ может включаться в состав записи и быть одним из ее полей, но может и не включаться в запись, а вычисляться по положению записи. Таблица может иметь один или несколько ключей.

Основной операцией при работе с таблицами является операция доступа к записи по ключу. Она реализуется процедурой поиска. Поскольку поиск может быть значительно более эффективным в таблицах, упорядоченных по значениям ключей, довольно часто над таблицами необходимо выполнять операции сортировки. Эти операции рассматриваются в следующих разделах.

Объективным критерием, позволяющим оценить эффективность того или иного алгоритма, является, так называемый, порядок алгоритма. Порядком алгоритма называется функция O(N), позволяющая оценить зависимость времени выполнения алгоритма от объема перерабатываемых данных (N - количество элементов в массиве или таблице). Эффективность алгоритма тем выше, чем меньше время его выполнения зависит от объема данных.

# Постановка задачи

Разработать программу, которая предоставляет возможность хранить данные в таблицах 3-х типов:

1. Неупорядоченная таблицы
2. Упорядоченная таблица
3. Хэш-таблица

В качестве данных использовать полиномы из лабораторной работы 4.

Особенности реализации:

Работа происходит сразу с таблицами всех типов.

Пользователь может положить в таблицу свой полином, с заданием ему некоторого наименования (ключа).

Пользователь может запросить (найти в/извлечь из) таблицы полином с заданным наименованием.

В ходе выполнения программы у пользователя должна оставаться возможность производить операции с полиномами, а также размещать полученный результат в таблицу.

Во время работы программы происходит логирование количества произведенных операций. Пример: Пользователь ввел свой полином и запросил положить его в таблица, программа в ответ выводит на экран или в файл, какое действие произошло и сколько операций потребовалось на нее у разных типов таблиц. Аналогично с поиском и извлечением.

# Руководство пользователя

1. Запустите программу
2. Выберите действие
3. Для выхода из программы нужно выбрать соответствующее действие

## Правила ввода полинома

# Сначала вводится имя полинома

# Степень должна быть не отрицательна и меньше 10

# Коэффициент монома не должен быть равен 0

# Сначала вводится количество мономов

# Затем для каждого монома требуется ввести его коэффициент и степени переменных

# Руководство программиста

## Описание структур данных

Мономы представляют собой структуру, в которой содержатся коэффициент монома и свернутая степень переменных, а также указатель на следующий моном.

Класс полиномов представляет из себя односвязный список мономов с фиктивной головой. Элементы списка хранятся упорядоченными.

Строки представляют собой структуру в которой содержится имя(имя полинома) и значение(полином).

Таблицы состоят из массива строк. У неупорядоченной и упорядоченной таблицы дополнительное поле данных - текущее количество записей. У хеш-таблицы есть дополнительный массив, в котором хранятся значения, которые означают- был ли помещен в строку с таким же номером полином. Добавление записи в неупорядоченную таблицу добавляет ее в конец массива. Поиск и удаление в неупорядоченной таблице линейный. В упорядоченной таблице добавление и поиск происходит с помощью алгоритма бинарного поиска. В хеш-таблице добавление, поиск и удаление происходят с помощью хеш-функции.

## Описание алгоритмов

1.Бинарный поиск.

1. Вход: Образец, массив, R - размер массива, L - ноль.
2. В цикле пока L<R;
   1. Вычислить номер среднего элемента массива(M = (R-L)/2)
      1. Образец равен этому элементу?
         1. Если да, то выход
         2. Если нет, то образец меньше среднего элемента ?
            1. Если да тоL = M-1;
            2. Если нет, то R = M+1;
3. Выход: M - номер образца в массиве

2.Хеш функция

1. Вход: имя, N- размер таблицы.
2. Переводим каждую в число
3. Суммируем эти числа
4. Выход: Сумма деленная по модулю на N

## Описание структуры программы

Программа состоит из 5 проектов:

1. main(Запускаемый проект, в котором реализован пользовательский интерфейс)
2. pol(Библиотека, содержащая реализацию класса полиномов)
3. tables(Библиотека, содержащая реализацию классов таблиц )
4. gtest(Библиотека, необходимая для запуска тестов )
5. test(Проект, в котором содержатся тесты)

# Заключение

В итоге удалось реализовать программу, которая предоставляет возможность хранить данные в таблицах 3-х типов.

Реализовано следующее:

1. Работа происходит сразу с таблицами всех типов.
2. Пользователь может положить в таблицу свой полином, с заданием ему некоторого наименования (ключа).
3. Пользователь может запросить (найти в/извлечь из) таблицы полином с заданным наименованием.
4. В ходе выполнения программы у пользователя есть возможность производить операции с полиномами, а также размещать полученный результат в таблицу.
5. Во время работы программы происходит логирование количества произведенных операций.

# Литература

1. Левитин А. В. Глава 4. Метод декомпозиции: Бинарный поиск // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 180–183. — 576 с
2. Вирт Н. Алгоритмы + структуры данных = программы. — М.: «Мир», 1985. — С. 28.
3. Дональд Кнут. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming, vol.3. Sorting and Searching. — 2-е издание. — М.: «Вильямс», 2007. — С. 824

# Приложения

Вставка в упорядоченную таблицу

voidord:: insert(rowk) {

inti = 0, j, len, kk=0;

while (i<size) {

if (k.key.length() >arow[i].key.length()) {

len = arow[i].key.length();

}

else {

len = k.key.length();

}

j = 0;

while ((arow[i].key[j] == k.key[j]) && (j <len)) {

j++;

kk++;

}

if (j == len) {

if(k.key.length() <= arow[i].key.length()) {

for (int k = size-1; k >i; k--) {

arow[k + 1] =arow[k];

kk++;

}

arow[i] =k;

return;

}

else {

i++;

kk++;

}

}

else {

if (arow[i].key[j]>= k.key[j]) {

for (int k = size- 1; k >i; k--) {

arow[k + 1] =arow[k];

kk++;

}

arow[i] =k;

return;

}

i++;

kk++;

}

}

arow[i] =k;

size++;

}

Поиск в отсортированной таблице

polord::find(stringk) {

inti, len, kk = 0;

int l = 0, r = size, mid;

while (l <= r) {

mid = (l + r) / 2;

if (arow[mid].key ==k) {

returnarow[mid].polinom;

}

i = 0;

if (k.length() >arow[mid].key.length()) {

len = arow[mid].key.length();

kk++;

}

else {

len = k.length();

kk++;

}

while ((arow[mid].key[i] == k[i]) && (i<len)) {

i++;

kk++;

}

if (i == len) {

kk++;

if (k.length() <arow[mid].key.length())

r = mid - 1;

else

l = mid + 1;

}

else {

kk++;

if (arow[mid].key[i]>k[i])

r = mid - 1;

else

l = mid + 1;

}

}

throw("Полинома с таким именем не существует");

}

Хеш-функция

inthasht::hf(stringk) {

int res = 0;

for (inti = 0; i<k.length(); i++)

res += k[i];

return res % 64;

}

Вставкавхеш-таблицу

voidhasht::insert(rowk) {

int t = hf(k.key),s = t , i=1, kk = 0;

while (bt[s] != false) {

s = (t + i \* 3) % 64;

kk++;

}

arow[s] =k;

bt[s] = true;

}

Поисквхеш-таблице

polhasht::find(stringk) {

int t = hf(k), kk = 0;

int s = t, i = 1;

while (arow[s].key !=k) {

s = (t + i \* 3) % 64;

kk++;

if (s == t) {

throw("Полинома с таким именем не существует");

}

}

returnarow[s].polinom;

}