Практическая работа № 3

Хэш-таблицы

Цель работы: Изучить способы применения Хэш-таблиц

. **Оборудование:** Windows 10, Visual Studio, Microsoft Word.

1. КРАТКАЯ ТЕОРИЯ

Существует два основных вида хеш-таблиц: с цепочками и открытой адресацией. Хеш-таблица содержит некоторый массив HH, элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Хеш-код i=h(key)i=h(key) играет роль индекса в массиве HH, а зная индекс, мы можем выполнить требующуюся операцию (добавление, удаление или поиск).

Количество коллизий зависит от хеш-функции; чем лучше используемая хеш-функция, тем меньше вероятность их возникновения.

Пусть хеш-таблица имеет размер lenlen и в нее добавляют nn элементов. Рассмотрим p′(n)p′(n) — вероятность того, что не возникнет ни одной коллизии. Добавим два любых элемента в нашу хеш-таблицу. Вероятность того, что они не попадут в одну и ту же ячейку таблицы равна 1−1len1−1len. Возьмем еще один элемент. Тогда вероятность того, что третий элемент не попадет в одну из уже занятых ячеек равна 1−2len1−2len.

получим формулу: p′(n)=(1−1len)·(1−2len)…(1−n−1len)=len·(len−1)…(len−n+1)lenn=len!lenn·(len−n)!p′(n)=(1−1len)·(1−2len)…(1−n−1len)=len·(len−1)…(len−n+1)lenn=len!lenn·(len−n)!

Тогда p(n)p(n) — вероятность возникновения коллизии равна: p(n)=1−p′(n)p(n)=1−p′(n),

что в общем случае >12>12

◂ ◂

Способ разрешения коллизий — важная составляющая любой хеш-таблицы.

Полностью избежать коллизий для произвольных данных невозможно в принципе, и хорошая хеш-функция в состоянии только минимизировать их количество. Но, в некоторых специальных случаях их удаётся избежать. Если все ключи элементов известны заранее, либо меняются очень редко, то можно подобрать хеш-функцию, с помощью которой, все ключи будут распределены по хеш-таблице без коллизий. Это хеш-таблицы с прямой адресацией; в них все операции, такие как: поиск, вставка и удаление работают за O(1)O(1).

Если мы поделим число хранимых элементов на размер массива HH (число возможных значений хеш-функции), то узнаем коэффициент заполнения хеш-таблицы (англ. load factor). От этого параметра зависит среднее время выполнения операций.

Исходный код

```
#include <iostream>
#include <string>
```

```
#include <map>
using namespace std;

struct data1 {
int group, age, scholarship; // группа, возраст, стипендия

data1(void) { group = age = scholarship = 0; }

data1(int group, int age, int scholarship = 0)
: group(group), age(age), scholarship(scholarship) {}

data1(const data1& d) :
group(d.group), age(d.age), scholarship(d.scholarship) {}

inline friend ostream& operator <<(ostream& out, const data1& obj) {
return out << "[" << obj.group << " : " << obj.age
<< " : " << obj.scholarship << " ]";
}
};
// это и есть наша таблица, журнал деканата, ФИО здесь - ключ:
typedef map< string, data1 > faculty;

int main(void) {
setlocale(LC_ALL, "rus");

faculty filology;
// заполняем поочерёдно 3 записи в журнал:
filology.insert(pair< string, data1 >("Иванов Н.И.", data1(16, 19, 1000)));
filology.insert(pair< string, data1 >("Николаев В.П.", data1(15, 23)));
filology.insert(pair< string, data1 >("Петров А.П.", data1(10, 21)));
// вывод на экран ФИО и учётных данных
for (auto i = filology.begin(); i != filology.end(); i++) {
cout << i->first << " : " << i->second << endl;
}

data1 d1(filology["Иванов И.И."]);
data1 d2 = filology["Иванов И.И."];
cout << d2 << endl;
}
```

```
Консоль отладки Microsoft Visual Studio
Иванов Н.И. : [16 : 19 : 1000 ]
Николаев В.П. : [15 : 23 : 0 ]
Петров А.П. : [10 : 21 : 0 ]
[0 : 0 : 0 ]

C:\Users\Lisa Cool\source\repos\ConsoleApplication8\Debug\Console
м 0.
Чтобы автоматически закрывать консоль при остановке отладки, вклю
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно…
```