

LAPORAN PRAKTIKUM

MODUL 5

HASH TABLE



Disusun oleh:

Aji Tri Prasetyo

NIM : 2311102064

Dosen Pengampu:

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
2. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

BAB II

DASAR TEORI

Pengertian Hash

Hash adalah teknik penempatan elemen-elemen secara random kedalam memori atau storage space yang disebut tabel hash(hash table).

Tujuan hash yaitu mempercepat pencarian elemen atau data.

Fungsi hash

- a. Fungsi hash (hashing function) yaitu fungsi atau rumus yang memetakan elemen kedalam hash tabel.
- b. Teknik hash membagi key dengan suatu nilai (m). Nilai m juga menunjukkan besarnya tabel hash yang dipilih.
- c. Fungsi: $H(\text{key}) = \text{key} \bmod m$

Rehashing Fungsi Pemotongan

- a. Home address dicari dengan memotong nilai key ke jumlah digit tertentu yang lebih pendek.
- b. Contoh: NIM yang tadinya 8 digit, dipotong hanya menjadi 2 digit!

Fungsi Kelipatan

- a. Dilakukan kelipatan terhadap record key dengan bagian yang sama panjang, lalu setiap bagian dijumlahkan
- b. NIM 8 digit dibagi dua digit, hingga menjadi 4 buah.
- c. Misal: 22002521, dibagi 22 00 25 21 kemudian dijumlahkan: 68 Fungsi utama rehashing

BAB III

GUIDED

1. GUIDED 1

SOURCE CODE

```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key)
{
    return key % MAX_SIZE;
}
// Struktur data untuk setiap node
struct Node
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key), value(value),
                                next(nullptr) {}
};
// Class hash table
class HashTable
{
private:
    Node **table;

public:
    HashTable()
    {
        table = new Node *[MAX_SIZE]();
    }
};
```

```

~HashTable()
{
    for (int i = 0; i < MAX_SIZE; i++)
    {
        Node *current = table[i];
        while (current != nullptr)
        {
            Node *temp = current;
            current = current->next;
            delete temp;
        }
    }
    delete[] table;
}

// Insertion
void insert(int key, int value)
{
    int index = hash_func(key);
    Node *current = table[index];
    while (current != nullptr)
    {
        if (current->key == key)
        {
            current->value = value;
            return;
        }
        current = current->next;
    }
    Node *node = new Node(key, value);
    node->next = table[index];
    table[index] = node;
}

// Searching
int get(int key)

```

```

{
    int index = hash_func(key);
    Node *current = table[index];
    while (current != nullptr)
    {
        if (current->key == key)
        {
            return current->value;
        }
        current = current->next;
    }
    return -1;
}

// Deletion
void remove(int key)
{
    int index = hash_func(key);
    Node *current = table[index];
    Node *prev = nullptr;
    while (current != nullptr)
    {
        if (current->key == key)
        {
            if (prev == nullptr)
            {
                table[index] = current->next;
            }
            else
            {
                prev->next = current->next;
            }
            delete current;
            return;
        }
    }
}

```

```

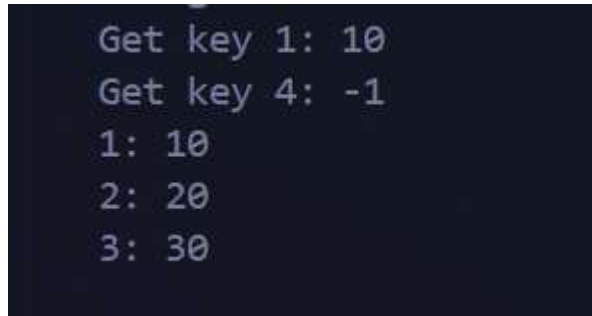
        prev = current;
        current = current->next;

    }
}
// Traversal
void traverse()
{
    for (int i = 0; i < MAX_SIZE; i++)
    {
        Node *current = table[i];
        while (current != nullptr)
        {
            cout << current->key << ": " << current->value
                << endl;
            current = current->next;
        }
    }
}
};
int main()
{
    HashTable ht;
    // Insertion
    ht.insert(1, 10);
    ht.insert(2, 20);
    ht.insert(3, 30);
    // Searching
    cout << "Get key 1: " << ht.get(1) << endl;
    cout << "Get key 4: " << ht.get(4) << endl;
    // Deletion
    ht.remove(4);
    // Traversal
    ht.traverse();
    return 0;
}

```

```
}
```

SCREENSHOOT PROGRAM



```
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
```

DESKRIPSI PROGRAM

Fungsi Hash: `hash_func(int key)` mengambil sebuah kunci (key) dan mengembalikan indeks dalam range dari 0 hingga `MAX_SIZE - 1`. Struktur Data

Node: Struktur Node menyimpan pasangan kunci-nilai dan pointer ke node berikutnya dalam hash table.

2. GUIDED 2

SOURCE CODE

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
string name;
string phone_number;
class HashNode
{
public:
    string name;
    string phone_number;
    HashNode(string name, string phone_number)
```


[illegible]

```

void remove(string name)
{
    int hash_val = hashFunc(name);
    for (auto it = table[hash_val].begin(); it !=
table[hash_val].end();
        it++)
    {
        if ((*it)->name == name)
        {
            table[hash_val].erase(it);
            return;
        }
    }
}

string searchByName(string name)
{
    int hash_val = hashFunc(name);
    for (auto node : table[hash_val])
    {
        if (node->name == name)
        {
            return node->phone_number;
        }
    }
    return "";
}

void print()
{
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        cout << i << ": ";
        for (auto pair : table[i])
        {

```

```

        if (pair != nullptr)
        {
            cout << "[" << pair->name << ", " << pair->phone_number << "]\n";
        }
    }
    cout << endl;
}

};

int main()
{
    HashMap employee_map;
    employee_map.insert("Mistah", "1234");
    employee_map.insert("Pastah", "5678");
    employee_map.insert("Ghana", "91011");
    cout << "Nomer Hp Mistah : "
         << employee_map.searchByName("Mistah") << endl;
    cout << "Phone Hp Pastah : "
         << employee_map.searchByName("Pastah") << endl;
    employee_map.remove("Mistah");
    cout << "Nomer Hp Mistah setelah dihapus : "
         << employee_map.searchByName("Mistah") << endl
         << endl;
    cout << "Hash Table : " << endl;
    employee_map.print();
    return 0;
}

```

SCREENSHOOT PROGRAM

```
Nomer Hp Mistah : 1234
Phone Hp Pastah : 5678
Nomer Hp Mistah setelah dihapus :

Hash Table :
0:
1:
2:
3:
4: [Pastah, 5678]
5:
6: [Ghana, 91011]
7:
8:
9:
10:
```

DESKRIPSI PROGRAM

Insertion: Fungsi `insert(string name, string phone_number)` menambahkan pasangan kunci-nilai ke dalam hash table. Jika kunci sudah ada, nilai yang sesuai diperbarui. Searching: Fungsi `searchByName(string name)` mencari nilai yang terkait dengan kunci (nama) yang diberikan dalam hash table.

UNGUIDED

1. UNGUIDED 1

Implementasikan hash table untuk menyimpan data mahasiswa. Setiap mahasiswa memiliki NIM dan nilai. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan nilai. Dengan ketentuan :

- a. Setiap mahasiswa memiliki NIM dan nilai.
- b. Program memiliki tampilan pilihan menu berisi poin C.
- c. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan rentang nilai (80 – 90).

SOURCE CODE

```
#include <iostream>
#include <vector>
#include <list>
using namespace std;

// Struktur data untuk mahasiswa
struct Mahasiswa {
    string nim;
    int nilai;
};

// Ukuran tabel hash
const int hashTableSize = 100;

// Class untuk hash table
class HashTable {
private:
```

```

vector<list<Mahasiswa>> table;

// Fungsi hash
int hashFunction(const string& key) {
    int sum = 0;
    for (char c : key) {
        sum += c;
    }
    return sum % hashTableSize;
}

public:
    // Constructor
    HashTable() {
        table.resize(hashTableSize);
    }

    // Fungsi untuk menambahkan data baru
    void tambahData(const string& nim, int nilai) {
        Mahasiswa mahasiswa;
        mahasiswa.nim = nim;
        mahasiswa.nilai = nilai;
        int index = hashFunction(nim);
        table[index].push_back(mahasiswa);
    }

    // Fungsi untuk menghapus data
    void hapusData(const string& nim) {
        int index = hashFunction(nim);
        for (auto it = table[index].begin(); it !=
table[index].end(); ++it) {
            if ((*it).nim == nim) {
                table[index].erase(it);
                break;
            }
        }
    }

```

```

    }

    }

}

// Fungsi untuk mencari data berdasarkan NIM
void cariByNIM(const string& nim) {
    int index = hashFunction(nim);
    for (auto it = table[index].begin(); it !=
table[index].end(); ++it) {
        if ((*it).nim == nim) {
            cout << "Data ditemukan - NIM: " << (*it).nim <<
", Nilai: " << (*it).nilai << endl;
            return;
        }
    }
    cout << "Data tidak ditemukan" << endl;
}

// Fungsi untuk mencari data berdasarkan rentang nilai (80 -
90)
void cariByRange() {
    for (int i = 0; i < hashTableSize; ++i) {
        for (auto it = table[i].begin(); it !=
table[i].end(); ++it) {
            if ((*it).nilai >= 80 && (*it).nilai <= 90) {
                cout << "NIM: " << (*it).nim << ", Nilai: "
<< (*it).nilai << endl;
            }
        }
    }
}

};

int main() {

```

```

HashTable hashTable;
int choice, nilai;
string nim;

do {
    cout << "\nMenu:\n";
    cout << "1. Menambah Data\n";
    cout << "2. Menghapus Data\n";
    cout << "3. Cari mahasiswa menggunakan NIM\n";
    cout << "4. Cari Nilai (80 - 90)\n";
    cout << "Masukkan Pilihan: ";
    cin >> choice;

    switch (choice) {
        case 1:
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan Nilai: ";
            cin >> nilai;
            hashTable.tambahData(nim, nilai);
            break;
        case 2:
            cout << "Masukkan NIM untuk dihapus: ";
            cin >> nim;
            hashTable.hapusData(nim);
            break;
        case 3:
            cout << "Masukkan NIM yang ingin dicari: ";
            cin >> nim;
            hashTable.cariByNIM(nim);
            break;
        case 4:
            cout << "Mahasiswa dengan nilai antara 80 -
90:\n";

```



```
        hashTable.cariByRange();  
        break;  
    case 5:  
        cout << "Keluar dari program.\n";  
        break;  
    default:  
        cout << "Pilihan tidak valid.\n";  
    }  
} while (choice != 5);  
  
return 0;  
}
```

SCREENSHOOT PROGRAM

```
Menu:
1. Menambah Data
2. Menghapus Data
3. Cari mahasiswa menggunakan NIM
4. Cari Nilai (80 - 90)
Masukkan Pilihan: 1
Masukkan NIM: 2311102064
Masukkan Nilai: 88

Menu:
1. Menambah Data
2. Menghapus Data
3. Cari mahasiswa menggunakan NIM
4. Cari Nilai (80 - 90)
Masukkan Pilihan: 3
Masukkan NIM yang ingin dicari: 2311102064
Data ditemukan - NIM: 2311102064, Nilai: 88

Menu:
1. Menambah Data
2. Menghapus Data
3. Cari mahasiswa menggunakan NIM
4. Cari Nilai (80 - 90)
Masukkan Pilihan: 4
Mahasiswa dengan nilai antara 80 - 90:
NIM: 2311102064, Nilai: 88
```

DESKRIPSI PROGRAM

Fungsi Utama: Fungsi main digunakan untuk menguji implementasi tabel hash. Di dalamnya, kita membuat objek HashTable dan memberikan opsi kepada pengguna untuk menambahkan data, menghapus data, mencari data berdasarkan NIM, atau mencari data berdasarkan rentang nilai.

BAB IV

KESIMPULAN

Setelah melakukan pembelajaran mengenai Hash Table di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Hash adalah penempatan elemen-elemen secara random kedalam meory yang di sebut tabelhash
2. Hash juga berguna dalm mempercepat pencarian elemen
3. Fungsi hash ada 2 yaitu dengan cara penguadratan dan penabahan kode ascllAda 3 hal penting dalm hash-home adress-collision-rehashing

DAFTAR PUSTAKA

Philip, Wheeler. (2014, 16 Juni) Laporan Resmi Hash. diakses pada 10 Mei 2024 dari <https://www.scribd.com/doc/229952345/Laporan-resmi-Hash>