LAPORAN PRAKTIKUM
MODUL V
HASH TABLE

Disusun oleh:

Raihan Ramadhan

NIM: 2311102040

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

Praktikum ini bertujuan untuk memastikan bahwa mahasiswa memiliki pemahaman yang baik tentang apa itu hash code, bagaimana cara kerjanya, dan mengapa penting dalam pemrograman. Mahasiswa diharapkan dapat menjelaskan dengan jelas apa itu hash code dan bagaimana ia bekerja sebagai bagian dari struktur data. Praktikum ini memberikan kesempatan kepada mahasiswa untuk mengimplementasikan hash code dalam bahasa pemrograman c++. Melalui latihan konkret, mahasiswa dapat belajar bagaimana mendefinisikan dan menggunakan hash code dalam konteks nyata. Mahasiswa mungkin akan diberikan situasi atau masalah yang memerlukan penggunaan hash code, dan mereka akan belajar bagaimana cara menguji keefektifan solusi mereka menggunakan hash code.

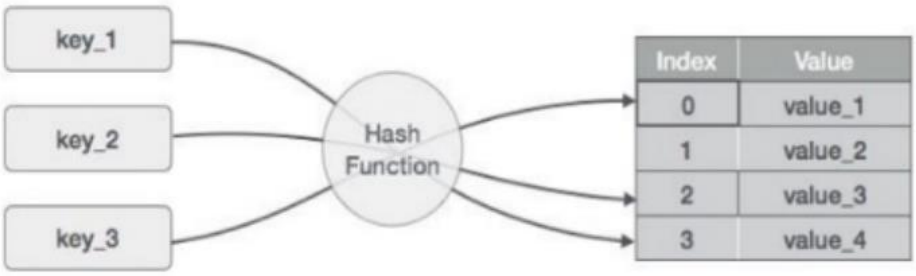
BAB II

DASAR TEORI

A. Pengertian Hash Table

Hash Table merupakan sebuah struktur data yang mengatur data dalam bentuk pasangan kunci-nilai. Umumnya, Hash Table terdiri dari dua elemen utama, yaitu sebuah array atau vektor, dan sebuah fungsi hash. Hashing sendiri adalah metode untuk mengonversi rentang nilai kunci menjadi rentang indeks dalam array. Dengan demikian, hash table memungkinkan untuk penyimpanan dan pencarian data yang efisien berdasarkan kunci yang diberikan. Array menyimpan data dalam slot-slot yang disebut bucket. Setiap bucket dapat menampung satu atau beberapa item data. Fungsi hash digunakan untuk menghasilkan nilai unik dari setiap item data, yang digunakan sebagai indeks array.

Sistem hash table bekerja dengan cara mengambil input kunci dan memetakkannya ke nilai indeks array menggunakan fungsi hash. Kemudian, data disimpan pada posisi indeks array yang dihasilkan oleh fungsi hash. Ketika data perlu dicari, input kunci dijadikan sebagai parameter untuk fungsi hash, dan posisi indeks array yang dihasilkan digunakan untuk mencari data. Dalam kasus hash collision, di mana dua atau lebih data memiliki nilai hash yang sama, hash table menyimpan data tersebut dalam slot yang sama dengan Teknik yang disebut chaining.



B. Fungsi Hash Table

Fungsi hash menghubungkan kunci dengan nilai dengan menggunakan suatu rumus matematika yang dikenal sebagai fungsi hash. Hasil dari proses ini disebut nilai hash atau hash. Nilai hash ini adalah representasi dari string karakter asli, namun biasanya memiliki ukuran yang lebih kecil dari aslinya.

C. Operaai Hash Table

1. Pencarian (Search): Digunakan untuk mencari elemen/data dalam Hash Table berdasarkan kunci atau indeksnya. Pencarian dilakukan dengan menggunakan fungsi hash untuk menghasilkan indeks elemen yang dicari.
2. Penyisipan (Insertion): Operasi ini digunakan untuk menyisipkan elemen/data baru ke dalam Hash Table. Elemen baru akan ditempatkan pada indeks yang dihasilkan oleh fungsi hash.
3. Penghapusan (Deletion): Digunakan untuk menghapus elemen/data dari Hash Table berdasarkan kunci atau indeksnya. Elemen yang dihapus akan dihapus dari indeks yang dihasilkan oleh fungsi hash.
4. Update: Operasi ini digunakan untuk mengubah nilai elemen/data yang sudah ada dalam Hash Table. Nilai elemen dapat diubah berdasarkan kunci atau indeksnya.
5. Collision Handling: Collision terjadi ketika dua atau lebih elemen memiliki indeks yang sama setelah melalui fungsi hash. Operasi ini digunakan untuk menangani collision dan memastikan bahwa elemen-elemen dengan indeks yang sama dapat disimpan dan diakses dengan benar.
6. Resize: Operasi ini digunakan untuk mengubah ukuran Hash Table jika jumlah elemen/data yang disimpan melebihi kapasitas yang ditentukan. Resize dilakukan untuk menjaga efisiensi dan kinerja Hash Table.
7. Iterasi: Operasi yang digunakan untuk mengakses dan memproses semua elemen/data yang ada dalam Hash Table secara berurutan. Ini memungkinkan untuk melakukan operasi seperti pencarian, penyisipan, dan penghapusan secara teratur pada setiap elemen dalam Hash Table.

BAB III

GUIDED

1. Guided 1

Source Code

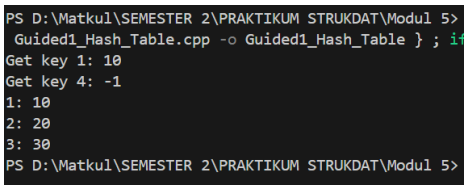
```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key)
{
    return key % MAX_SIZE;
}
// Struktur data untuk setiap node
struct Node
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key), value(value),
   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

```cpp
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;
}
```

Screenshot Output



Deskripsi Program

Program ini merupakan implementasi sederhana dari hash table dalam bahasa C++, menggunakan metode hashing modulo untuk menentukan indeks penyimpanan. Setiap elemen hash table direpresentasikan oleh struktur data Node yang memiliki kunci dan nilai, serta pointer ke node berikutnya. Kelas HashTable menyediakan metode untuk operasi dasar seperti insert, get, remove, dan traverse. Operasi-insert digunakan untuk menambahkan pasangan kunci-nilai, get untuk mencari nilai berdasarkan kunci, remove untuk menghapus pasangan kunci-nilai, dan traverse untuk melintasi dan mencetak seluruh isi hash table. Program ini menunjukkan cara penggunaan hash table untuk penyimpanan data dengan akses cepat berdasarkan kunci.

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)
    {
        this->name = name;
        this->phone_number = phone_number;
    }
};
class HashMap
{
private:
    vector<HashNode *> table[TABLE_SIZE];

public:
    int hashFunc(string key)
    {
        int hash_val = 0;
        for (char c : key)
        {
            hash_val += c;
        }
        return hash_val % TABLE_SIZE;
    }
    void insert(string name, string phone_number)
    {
        int hash_val = hashFunc(name);
        for (auto node : table[hash_val])
        {
            if (node->name == name)
            {
                node->phone_number = phone_number;
                return;
            }
        }
        table[hash_val].push_back(new HashNode(name,
                                               phone_number));
    }
```

```
    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 << "]";
                }
            }
            cout << endl;
        }
    }
};
```

```cpp
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;
}
```

Screenshot Output

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 5>
d2_Hash_Table.cpp -o Guided2_Hash_Table } ; if ($?)
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:
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 5>
```

Deskripsi Program

Program ini adalah implementasi sederhana dari hash map dalam bahasa C++, di mana setiap entri dalam tabel hash direpresentasikan oleh objek HashNode yang menyimpan pasangan nama dan nomor telepon. Kelas HashMap menyediakan fungsi-fungsi untuk operasi dasar seperti insert, remove, dan searchByName, menggunakan fungsi hash untuk menentukan indeks penyimpanan. Program ini menunjukkan cara penggunaan hash map untuk menyimpan dan mencari nomor telepon berdasarkan nama karyawan.

BAB IV

UNGUIDED

1. **Unguided 1**
   Source Code

```
#include <iostream>
#include <list>
#include <vector>
#include <string>

using namespace std;

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

// Class untuk hash table
class HashTable
{
private:
    static const int hashGroup = 10; // Jumlah grup hash
    vector<list<Mahasiswa>> table;   // Array dari linked list untuk
menyimpan data

    // Fungsi hashing sederhana untuk NIM
    int hashFunction(const string &NIM)
    {
        int total = 0;
        for (char ch : NIM)
        {
            total += ch;
        }
        return total % hashGroup;
    }
```

```
public:
    HashTable() : table(hashGroup) {}

    // Menambahkan data mahasiswa ke hash table
    void tambahData(const string &NIM, int nilai)
    {
        Mahasiswa mhs;
        mhs.NIM = NIM;
        mhs.nilai = nilai;
        int hashKey = hashFunction(NIM);
        table[hashKey].push_back(mhs);
    }

    // Menghapus data mahasiswa dari hash table berdasarkan NIM
    void hapusData(const string &NIM)
    {
        int hashKey = hashFunction(NIM);
        for (auto it = table[hashKey].begin(); it !=
table[hashKey].end(); ++it)
        {
            if ((*it).NIM == NIM)
            {
                table[hashKey].erase(it);
                break;
            }
        }
    }

    // Mencari data mahasiswa berdasarkan NIM
    void cariByNIM(const string &NIM)
    {
        int hashKey = hashFunction(NIM);
        for (const auto &mhs : table[hashKey])
        {
            if (mhs.NIM == NIM)
            {
                cout << "NIM: " << mhs.NIM << ", Nilai: " << mhs.nilai
<< endl;
                return;
            }
        }
        cout << "Data mahasiswa dengan NIM " << NIM << " tidak
ditemukan." << endl;
    }
```

```
    // Mencari data mahasiswa berdasarkan rentang nilai (80 - 90)
    void cariByRange()
    {
        for (const auto &group : table)
        {
            for (const auto &mhs : group)
            {
                if (mhs.nilai >= 80 && mhs.nilai <= 90)
                {
                    cout << "NIM: " << mhs.NIM << ", Nilai: " <<
mhs.nilai << endl;
                }
            }
        }
    }
};

int main()
{
    HashTable hashTable;

    // Menu
    cout << "Menu:" << endl;
    cout << "1. Tambah data mahasiswa" << endl;
    cout << "2. Hapus data mahasiswa" << endl;
    cout << "3. Cari data mahasiswa berdasarkan NIM" << endl;
    cout << "4. Cari data mahasiswa berdasarkan rentang nilai (80-90)"
<< endl;
    cout << "5. Keluar" << endl;

    int choice;
    do
    {
        cout << "Pilih menu: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
        {
            string NIM;
            int nilai;
            cout << "Masukkan NIM mahasiswa: ";
            cin >> NIM;
            cout << "Masukkan nilai mahasiswa: ";
            cin >> nilai;
            hashTable.tambahData(NIM, nilai);
            break;
        }
```

```
    // Mencari data mahasiswa berdasarkan rentang nilai (80 - 90)
    void cariByRange()
    {
        for (const auto &group : table)
        {
            for (const auto &mhs : group)
            {
                if (mhs.nilai >= 80 && mhs.nilai <= 90)
                {
                    cout << "NIM: " << mhs.NIM << ", Nilai: " <<
mhs.nilai << endl;
                }
            }
        }
    }
};

int main()
{
    HashTable hashTable;

    // Menu
    cout << "Menu:" << endl;
    cout << "1. Tambah data mahasiswa" << endl;
    cout << "2. Hapus data mahasiswa" << endl;
    cout << "3. Cari data mahasiswa berdasarkan NIM" << endl;
    cout << "4. Cari data mahasiswa berdasarkan rentang nilai (80-90)"
<< endl;
    cout << "5. Keluar" << endl;

    int choice;
    do
    {
        cout << "Pilih menu: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
        {
            string NIM;
            int nilai;
            cout << "Masukkan NIM mahasiswa: ";
            cin >> NIM;
            cout << "Masukkan nilai mahasiswa: ";
            cin >> nilai;
            hashTable.tambahData(NIM, nilai);
            break;
        }
```

```
        case 2:
        {
            string NIM;
            cout << "Masukkan NIM mahasiswa yang ingin dihapus: ";
            cin >> NIM;
            hashTable.hapusData(NIM);
            break;
        }
        case 3:
        {
            string NIM;
            cout << "Masukkan NIM mahasiswa yang ingin dicari: ";
            cin >> NIM;
            hashTable.cariByNIM(NIM);
            break;
        }
        case 4:
            cout << "Mahasiswa dengan nilai antara 80 dan 90:" << endl;
            hashTable.cariByRange();
            break;
        case 5:
            cout << "Keluar dari program." << endl;
            break;
        default:
            cout << "Pilihan tidak valid. Silakan pilih lagi." << endl;
        }
    } while (choice != 5);

    return 0;
}
```

ScreenShot Output

a. Setiap mahasiswa memiliki NIM dan nilai.

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 5> cd "d:\Ma
ded1_Hash_Table.cpp -o Unguided1_Hash_Table } ; if ($?) { .\U
Menu:
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari data mahasiswa berdasarkan NIM
4. Cari data mahasiswa berdasarkan rentang nilai (80-90)
5. Keluar
Pilih menu: 1
Masukkan NIM mahasiswa: 2311102040
Masukkan nilai mahasiswa: 90
```

b. Program memiliki tampilan pilihan menu berisi poin C

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 5> cd "
ded1_Hash_Table.cpp -o Unguided1_Hash_Table } ; if ($?)
Menu:
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari data mahasiswa berdasarkan NIM
4. Cari data mahasiswa berdasarkan rentang nilai (80-90)
5. Keluar
```

c. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan rentang nilai (80 – 90).

- Menambahkan data baru

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Modul 5> cd "
ded1_Hash_Table.cpp -o Unguided1_Hash_Table } ; if ($?)
Menu:
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari data mahasiswa berdasarkan NIM
4. Cari data mahasiswa berdasarkan rentang nilai (80-90)
5. Keluar
Pilih menu: 1
Masukkan NIM mahasiswa: 2311102040
Masukkan nilai mahasiswa: 90
Pilih menu: 1
Masukkan NIM mahasiswa: 2311102034
Masukkan nilai mahasiswa: 86
Pilih menu: 1
Masukkan NIM mahasiswa: 2311102030
Masukkan nilai mahasiswa: 80
Pilih menu: 1
Masukkan NIM mahasiswa: 2311102041
Masukkan nilai mahasiswa: 56
Pilih menu: 1
Masukkan NIM mahasiswa: 2311102050
Masukkan nilai mahasiswa: 76
```

- Menghapus Data

```
Pilih menu: 2
Masukkan NIM mahasiswa yang ingin dihapus: 2311102041
Pilih menu: 3
Masukkan NIM mahasiswa yang ingin dicari: 2311102041
Data mahasiswa dengan NIM 2311102041 tidak ditemukan.
```

- Mencari data berdasarkan NIM

```
Pilih menu: 3
Masukkan NIM mahasiswa yang ingin dicari: 2311102040
NIM: 2311102040, Nilai: 90
```

- Mencari data berdasarkan rentang nilai (80 – 90)

```
Pilih menu: 4
Mahasiswa dengan nilai antara 80 dan 90:
NIM: 2311102030, Nilai: 80
NIM: 2311102040, Nilai: 90
NIM: 2311102034, Nilai: 86
```

Deskripsi Program

Program di atas merupakan implementasi dari sebuah hash table yang digunakan untuk menyimpan data mahasiswa. Hash table tersebut menggunakan chaining untuk menangani tabrakan hash (collision), dimana setiap elemen dari array table adalah linked list yang menyimpan data mahasiswa. Fungsi hash sederhana digunakan untuk menghasilkan indeks dari NIM mahasiswa, dimana total dari nilai ASCII dari karakter-karakter dalam NIM diambil modulus dengan jumlah grup hash. Program menyediakan menu interaktif yang memungkinkan pengguna untuk menambahkan data mahasiswa, menghapus data berdasarkan NIM, mencari data berdasarkan NIM, mencari data berdasarkan rentang nilai (80-90), dan keluar dari program. Saat menambahkan data, pengguna diminta untuk memasukkan NIM dan nilai mahasiswa, sedangkan saat mencari atau menghapus data, pengguna diminta untuk memasukkan NIM mahasiswa yang bersangkutan.

BAB V

KESIMPULAN

Hash table adalah sebuah struktur data yang memungkinkan penyimpanan dan pencarian data secara efisien menggunakan kunci. Dengan memanfaatkan fungsi hash, kunci diubah menjadi indeks dalam sebuah array, yang memungkinkan akses data dalam waktu yang sangat cepat. Namun, perlu ada penanganan untuk situasi ketika dua kunci menghasilkan indeks yang sama, yang dikenal sebagai tabrakan hash. Walaupun hash table memiliki keunggulan dalam kecepatan akses data, ada juga kelemahan seperti kebutuhan ruang penyimpanan yang lebih besar dan kompleksitas analisis yang bergantung pada kualitas fungsi hash. Meskipun demikian, hash table tetap menjadi pilihan yang sangat efisien untuk aplikasi yang membutuhkan operasi pencarian, penambahan, dan penghapusan data dengan cepat.

DAFTAR PUSTAKA

Asisten Praktikum “MODUL V HASH TABLE”. Learning Management System,13 Maret 2024.

UM FT 2010 ” Praktikum Algoritma dan Struktur Data 2010”.Chrome,12 Maret 2024. https://elektro.um.ac.id/wp-content/uploads/2016/04/Struktur-Data-Modul-Praktikum-11-Hashing-Table.pdf .