LAPORAN PRAKTIKUM

MODUL V
“HASH TABLE”

Disusun oleh:
Andika Indra Prastawa
NIM: 2311102033

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd.,M.Eng.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

BAB I
TUJUAN PRAKTIKUM

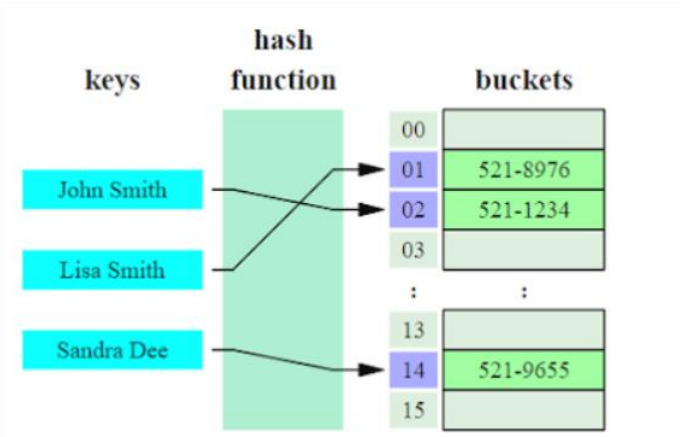
a. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code

b. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

BAB II

DASAR TEORI

Hashing merupakan teknik yang digunakan untuk menyusun dan mengakses elemen data dalam List dengan waktu yang relatif konstan melalui manipulasi key untuk mengidentifikasi lokasi dalam List. Hash function merupakan fungsi yang digunakan untuk memanipulasi key dari elemen data dalam List untuk mengidentifikasi lokasi aslinya di list. Fungsi ini akan memetakan list data yang ukurannya berubah-ubah ke ukuran tetap. Nilai kembalian dari fungsi hash disebut dengan Hash Values.



Hash table adalah sebuah struktur data yang terdiri atas sebuah tabel dan fungsi yang bertujuan untuk memetakan nilai kunci yang unik untuk setiap record menjadi angka (hash) lokasi record tersebut dalam sebuah tabel.

Fungsi Hash Table

Fungsi hash membuat pemetaan antara kunci dan nilai, hal ini dilakukan melalui penggunaan rumus matematika yang dikenal sebagai fungsi hash. Hasil dari fungsi hash disebut sebagai nilai hash atau hash. Nilai hash adalah representasi dari string karakter asli tetapi biasanya lebih kecil dari aslinya.

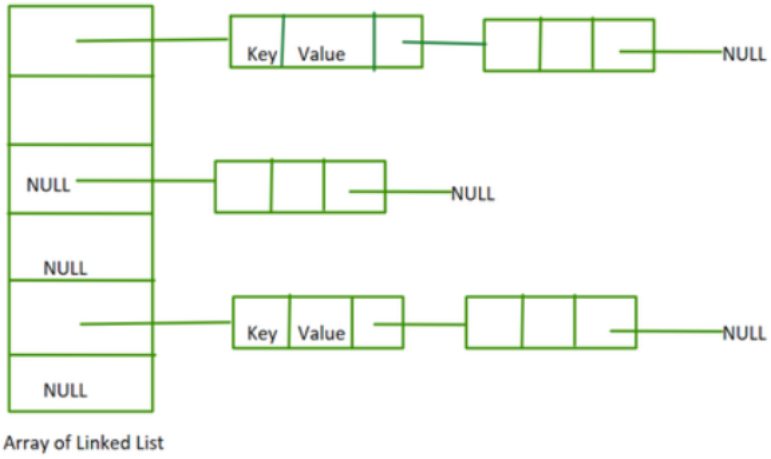
Beberapa Contoh Fungsi Hash:

- kunci % jumlah ember
- Nilai karakter ASCII * PrimeNumber x . Dimana x = 1, 2, 3….n

- Anda dapat membuat fungsi hash Anda sendiri tetapi fungsi hash tersebut harus bagus dan memberikan jumlah tabrakan yang lebih sedikit

Fungsi Utama dalam Program Hash Table

- Insertion
- Deletion
- Searching
- Update
- Traversal



Hash Map

BAB III

GUIDED

1. Guided 1

Source code

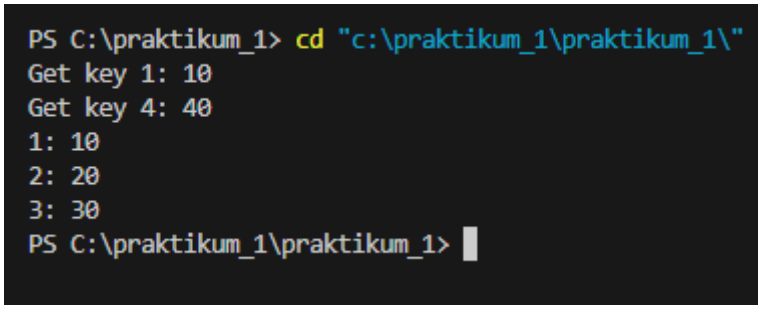
```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key)
{
    return key % MAX_SIZE;
}
// Struktur data untuk setiap node
struct Node
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key), value(value),
                               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 40;
    }
    // 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



Deskripsi program

Program ini merupakan penerapan dari hash table sederhana yang memanfaatkan fungsi hash untuk menghasilkan indeks tabel berdasarkan kunci. Program ini menyediakan fungsi untuk menambahkan data, mencari data, menghapus data, dan mencetak seluruh data yang disimpan. Program ini terbagi menjadi tiga bagian utama, yaitu struktur data untuk setiap node, kelas hash table, dan fungsi utama.

2. Guided 2

Source code

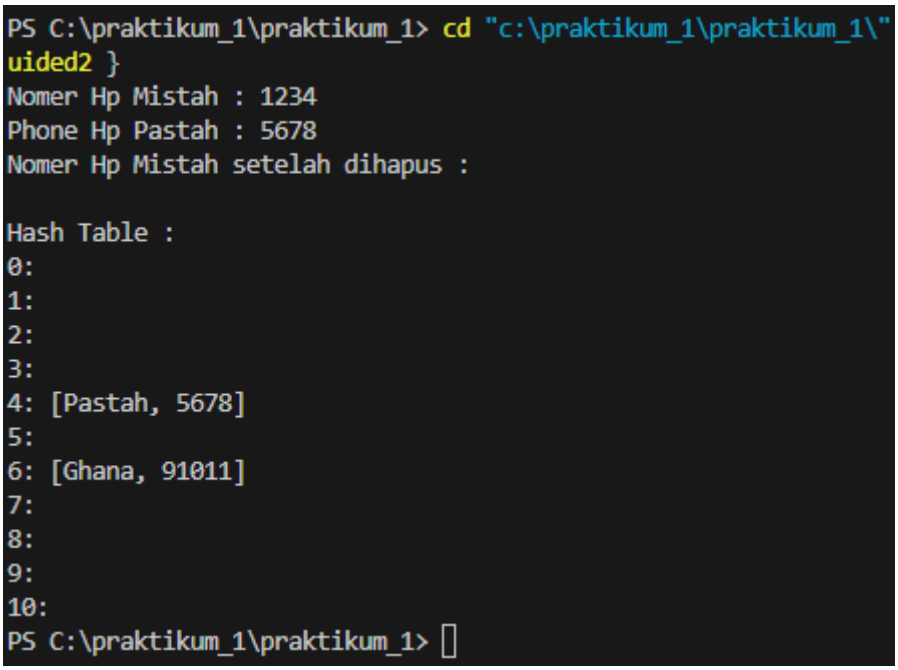
```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
```

```
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;
        }
    }
};
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



Deskripsi program

Program ini adalah implementasi dari HashMap yang digunakan untuk menyimpan data nama dan nomor telepon karyawan. Program ini memanfaatkan fungsi hash untuk menghasilkan indeks tabel berdasarkan nama karyawan. Program ini menyediakan fungsi untuk menambahkan data, mencari data, menghapus data, dan mencetak semua data yang tersimpan. Program ini terdiri dari tiga kelas utama: HashNode, HashMap, dan main. Kelas HashNode mewakili node dalam tabel hash, di mana setiap node berisi nama dan nomor telepon karyawan.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

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

struct dataMahasiswa
{
    long long Nim_2311102033;
    int nilai;
};

class HashTable
{
private:
    static const int hashGroup = 10;
    vector<list<dataMahasiswa>> table;

public:
    HashTable() : table(hashGroup) {}
    int hashFunction(long long Nim_2311102033)
    {
        return Nim_2311102033 % hashGroup;
    }

    void tambahData(long long Nim_2311102033, int nilai)
    {
        dataMahasiswa mhs;
        mhs.Nim_2311102033 = Nim_2311102033;
        mhs.nilai = nilai;
        int hashKey = hashFunction(Nim_2311102033);
        table[hashKey].push_back(mhs);
    }
    void hapusData(long long Nim_2311102033)
    {
        int hashKey = hashFunction(Nim_2311102033);
```

```
        for (auto it = table[hashKey].begin(); it != table[hashKey].end(); ++it)
        {

            if ((*it).Nim_2311102033 == Nim_2311102033)
            {
                table[hashKey].erase(it);
                cout << "Nim " << it->Nim_2311102033 << " telah di hapus" << endl;
                break;
            }
        }
    }
    void cariNIM(long long Nim_2311102033)
    {
        int hashKey = hashFunction(Nim_2311102033);
        for (auto mhs : table[hashKey])
        {
            if (mhs.Nim_2311102033 == Nim_2311102033)
            {
                cout << "Nim: " << mhs.Nim_2311102033 << ", Nilai: " << mhs.nilai << endl;
                return;
            }
        }
        cout << "Data mahasiswa dengan Nim " << Nim_2311102033 << " tidak." << endl;
    }
    void cariRentang()
    {
        for (const auto &group : table)
        {
            for (const auto &mhs : group)
            {
                if (mhs.nilai >= 80 && mhs.nilai <= 90)
                {
                    cout << "Nim: " << mhs.Nim_2311102033 << ", Nilai: " << mhs.nilai << endl;
                }
            }
        }
```

```
    }
};
int main()
{
    int Pilihanmu, nilai;
    long long Nim_2311102033;
    HashTable hashTable;

    cout << "\n\t\t     PROGRAM UNTUK MENYIMPAN DATA MAHASISWA\n\n";
    cout << "+-------------------------------------------------------
----------------------+\n";
    cout << "| Menu | Pilihan |                              Keterangan
|\n";
    cout << "+------+---------+-----------------------------------
----------------------+\n";
    cout  <<  "|    1       |  Tambah    |  Tambah  data  mahasiswa
|\n";
    cout  <<  "|    2       |  Hapus      |  Hapus  data  mahasiswa
|\n";
    cout << "|  3   | Cari    | Cari data mahasiswa berdasarkan Nim
|\n";
    cout << "|   4    | Rentang | Cari data mahasiswa berdasarkan
rentang nilai (80-90)   |\n";
    cout  <<  "|    5       |  Keluar    |  Keluar  dari  program
|\n";
    cout << "+------+---------+-----------------------------------
----------------------+\n";

    do
    {
        cout << "Pilih menu (1-5): ";
        cin >> Pilihanmu;
        switch (Pilihanmu)
        {
        case 1:
            cout << "Masukkan Nim mahasiswa: ";
            cin >> Nim_2311102033;
            cout << "Masukkan nilai mahasiswa: ";
            cin >> nilai;
            hashTable.tambahData(Nim_2311102033, nilai);
            break;
```

```
        case 2:
            cout << "Masukkan Nim mahasiswa yang ingin dihapus: ";
            cin >> Nim_2311102033;
            hashTable.hapusData(Nim_2311102033);
            break;
        case 3:
            cout << "Masukkan Nim mahasiswa yang ingin dicari: ";
            cin >> Nim_2311102033;
            hashTable.cariNIM(Nim_2311102033);
            break;
        case 4:
            cout << "Mahasiswa dengan nilai antara 80 dan 90:" <<
endl;
            hashTable.cariRentang();
            break;
        case 5:
            cout << "Terimakasih Telah Memakai program ini " << endl;
            break;
        default:
            cout << "NOT FOUND." << endl;
        }
    } while (Pilihanmu != 5);
    return 0;
}
```

Screenshoot program

- **Setiap mahasiswa memiliki NIM dan nilai**

```
PS C:\praktikum_1\praktikum_1> cd "c:\praktikum_1\praktikum_1\" ; if ($?) { g++
ak5Unguided1 }

                    PROGRAM UNTUK MENYIMPAN DATA MAHASISWA

+-------------------------------------------------------------------------+
| Menu | Pilihan |                    Keterangan                          |
+------+---------+--------------------------------------------------------+
|  1   | Tambah  | Tambah data mahasiswa                                  |
|  2   | Hapus   | Hapus data mahasiswa                                   |
|  3   | Cari    | Cari data mahasiswa berdasarkan Nim                    |
|  4   | Rentang | Cari data mahasiswa berdasarkan rentang nilai (80-90)  |
|  5   | Keluar  | Keluar dari program                                    |
+------+---------+--------------------------------------------------------+
Pilih menu (1-5): 1
Masukkan Nim mahasiswa: 2311102033
Masukkan nilai mahasiswa: 87
Pilih menu (1-5):
```

- **Program memiliki menu yang berisi poin C**

```
PS C:\praktikum_1\praktikum_1> cd "c:\praktikum_1\praktikum_1\" ; if ($?) { g++
ak5Unguided1 }

                    PROGRAM UNTUK MENYIMPAN DATA MAHASISWA

+-------------------------------------------------------------------------+
| Menu | Pilihan |                    Keterangan                          |
+------+---------+--------------------------------------------------------+
|  1   | Tambah  | Tambah data mahasiswa                                  |
|  2   | Hapus   | Hapus data mahasiswa                                   |
|  3   | Cari    | Cari data mahasiswa berdasarkan Nim                    |
|  4   | Rentang | Cari data mahasiswa berdasarkan rentang nilai (80-90)  |
|  5   | Keluar  | Keluar dari program                                    |
+------+---------+--------------------------------------------------------+
```

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

```
                    PROGRAM UNTUK MENYIMPAN DATA MAHASISWA

+-------------------------------------------------------------------------+
| Menu | Pilihan |                    Keterangan                          |
+------+---------+--------------------------------------------------------+
|  1   | Tambah  | Tambah data mahasiswa                                  |
|  2   | Hapus   | Hapus data mahasiswa                                   |
|  3   | Cari    | Cari data mahasiswa berdasarkan Nim                    |
|  4   | Rentang | Cari data mahasiswa berdasarkan rentang nilai (80-90)  |
|  5   | Keluar  | Keluar dari program                                    |
+------+---------+--------------------------------------------------------+
Pilih menu (1-5): 1
Masukkan Nim mahasiswa: 2311102033
Masukkan nilai mahasiswa: 87
Pilih menu (1-5): 1
Masukkan Nim mahasiswa: 2311102099
Masukkan nilai mahasiswa: 82
Pilih menu (1-5): 1
Masukkan Nim mahasiswa: 2311102088
Masukkan nilai mahasiswa: 86
Pilih menu (1-5): 1
Masukkan Nim mahasiswa: 2311102077
Masukkan nilai mahasiswa: 81
Pilih menu (1-5): 1
Masukkan Nim mahasiswa: 2311102066
Masukkan nilai mahasiswa: 89
```

- **Menghapus Data**

```
Pilih menu (1-5): 2
Masukkan Nim mahasiswa yang ingin dihapus: 2311102099
Nim 2311102099 telah di hapus
```

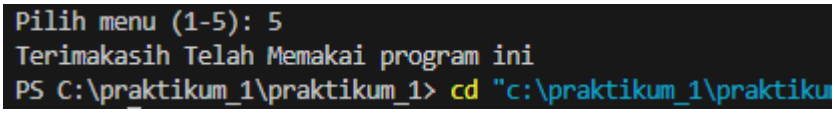
- **Mencari Data Menggunakan Nim**

```
Pilih menu (1-5): 3
Masukkan Nim mahasiswa yang ingin dicari: 2311102033
Nim: 2311102033, Nilai: 87
```

- **Mencari Data Berdasarkan Rentang Nilai (80 – 90)**

```
Pilih menu (1-5): 4
Mahasiswa dengan nilai antara 80 dan 90:
Nim: 2311102033, Nilai: 87
Nim: 2311102066, Nilai: 89
Nim: 2311102077, Nilai: 81
Nim: 2311102088, Nilai: 86
```

- **Keluar Dari Program**



Deskripsi program

Program ini merupakan implementasi dasar dari hash table yang digunakan untuk menyimpan data mahasiswa. Pengguna dapat melakukan penambahan data mahasiswa, penghapusan data berdasarkan NIM, pencarian data berdasarkan NIM, serta pencarian data mahasiswa dalam rentang nilai tertentu. Program ini menyediakan menu pilihan untuk menjalankan operasi-operasi tersebut. Data mahasiswa disimpan dalam struktur dataMahasiswa dengan NIM sebagai kunci dan nilai mahasiswa.

KESIMPULAN

Dari modul praktikum hash table, dapat disimpulkan bahwa hashing adalah teknik penting untuk memetakan data dari kumpulan besar ke kumpulan yang lebih kecil menggunakan fungsi hash, dengan metode seperti chaining atau probing untuk menangani tabrakan. Implementasi hash table memungkinkan pencarian data yang cepat dengan kompleksitas waktu rata-rata O(1), tetapi memerlukan pemilihan fungsi hash yang baik dan sensitif terhadap faktor muatan tabel. Meskipun hash table unggul dalam efisiensi pencarian data, ada kelemahan seperti overhead memori untuk penanganan tabrakan dan kinerja yang dapat menurun saat faktor muatan tinggi. Dengan pemahaman yang baik tentang konsep dan implementasi hash table, kita dapat menggunakannya secara efektif dalam pengembangan aplikasi untuk operasi pencarian dan penyimpanan data yang cepat.

DAFTAR PUSTAKA

[1] https://fajarbaskoro.blogspot.com/2021/06/hash-table.html

[2] https://www.geeksforgeeks.org/implementation-of-hash-table-in-c-using-separate-chaining/

[3] http://aren.cs.ui.ac.id/sda/resources/sda2010/15_hashtable.pdf