LAPORAN PRAKTIKUM
MODUL 5
HASH TABLE

Disusun oleh:

Rasyid Nafsyarie

NIM : 2311102011

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

a. Pengertian Hash Table

Hash Table adalah struktur data yang mengorganisir data ke dalam pasangan kunci-nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vektor) dan fungsi hash. Hashing adalah teknik untuk mengubah rentang nilai kunci menjadi rentang indeks array. 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. Dengan cara ini, hash table memungkinkan pencarian data dalam waktu yang konstan (O(1)) dalam kasus terbaik. 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 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.

c. Operasi Hash Table

1. Insertion: Memasukkan data baru ke dalam hash table dengan memanggil fungsi hash untuk menentukan posisi bucket yang tepat, dan kemudian menambahkan data ke bucket tersebut.
2. Deletion: Menghapus data dari hash table dengan mencari data menggunakan fungsi hash, dan kemudian menghapusnya dari bucket yang sesuai.
3. Searching: Mencari data dalam hash table dengan memasukkan input kunci ke fungsi hash untuk menentukan posisi bucket, dan kemudian mencari data di dalam bucket yang sesuai.
4. Update: Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.
5. Traversal: Melalui seluruh hash table untuk memproses semua data yang ada dalam tabel.

d. Collision Resolution

Keterbatasan tabel hash adalah jika dua angka dimasukkan ke dalam fungsi hash menghasilkan nilai yang sama. Hal ini disebut dengan collision.

BAB III

GUIDED

1. GUIDED 1

SOURCE CODE

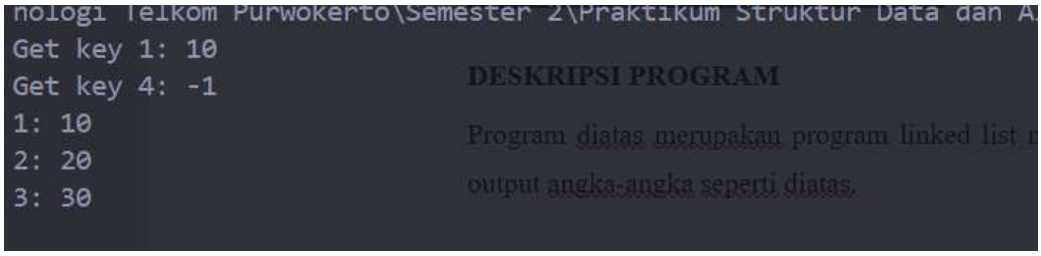
```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key)
{
    return key % MAX_SIZE;
}
// Struktur data untuk setiap node
struct Node
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key), value(value),
   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



DESKRIPSI PROGRAM

Fungsi Hash: Fungsi hash adalah fungsi yang mengonversi kunci menjadi indeks dalam tabel hash. Dalam contoh ini, kita menggunakan fungsi hash sederhana yang mengembalikan sisa pembagian kunci dengan ukuran maksimum tabel hash.

2. GUIDED 2

SOURCE CODE

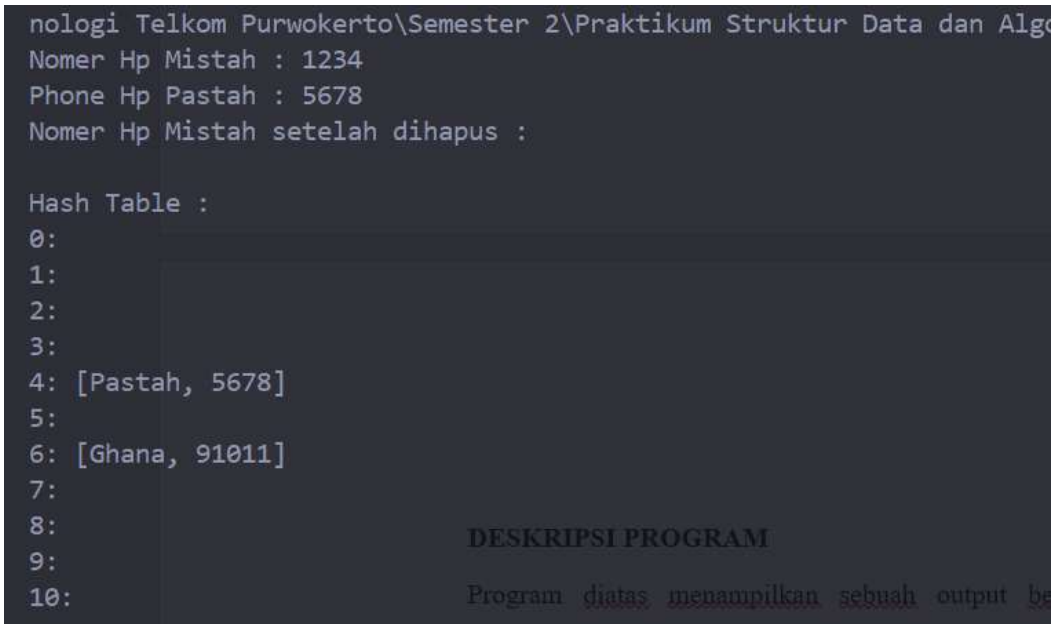
```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
string name;
string phone_number;
class HashNode
{
public:
    string name;
    string phone_number;
    HashNode(string name, string phone_number)
    {
        this->name = name;
        this->phone_number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

HashNode: Ini adalah kelas yang mewakili simpul dalam HashMap. Setiap simpul memiliki dua atribut, yaitu name (nama) dan phone_number (nomor telepon).

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>
//Rasyid Nafsyarie 2311102011

using namespace std;

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

// Ukuran tabel hash
const int hashTableSize = 100;

// Class untuk hash table
```

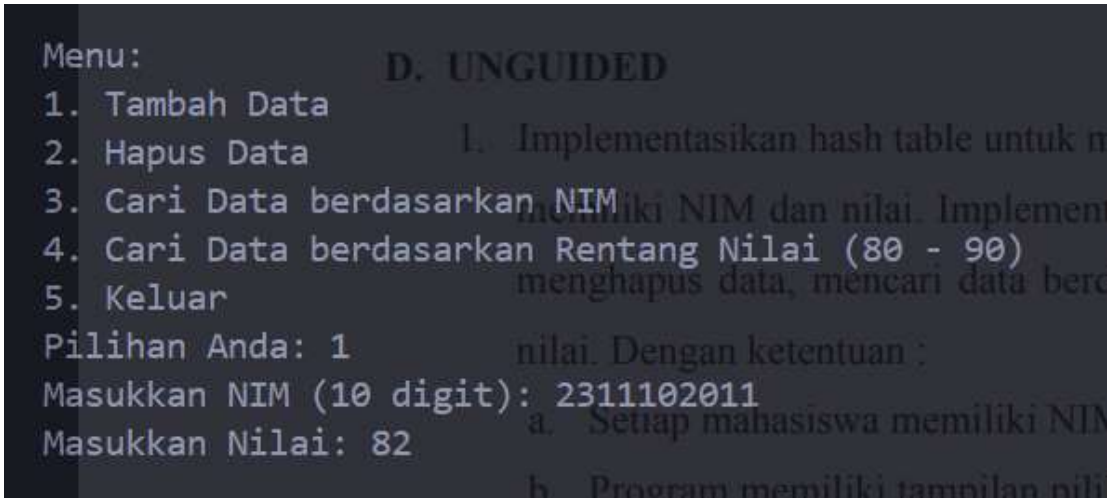
```
class HashTable {
private:
    vector<list<Mahasiswa>> table;

    // 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;
                }
            }
        }
    }
};
```

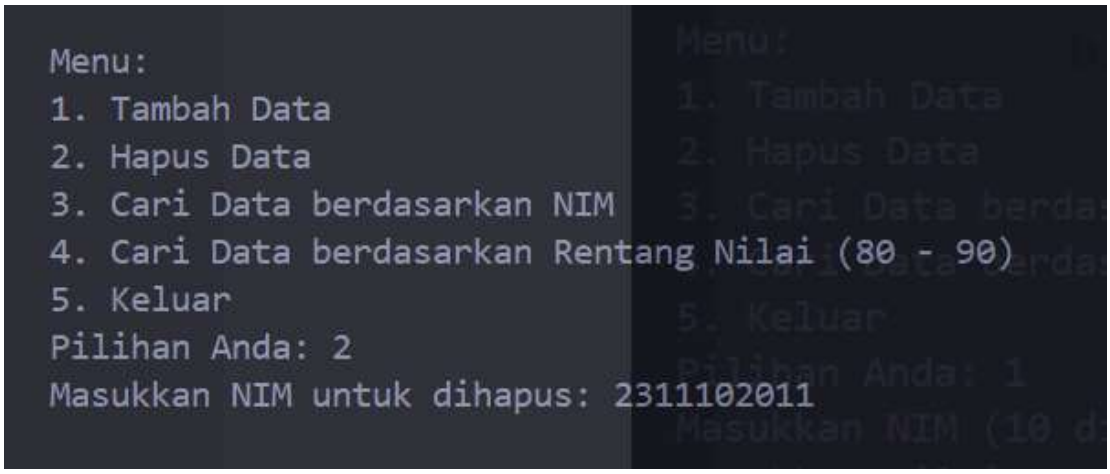
```
int main() {
    HashTable hashTable;
    int choice, nilai;
    string nim;

    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Data\n";
        cout << "2. Hapus Data\n";
        cout << "3. Cari Data berdasarkan NIM\n";
        cout << "4. Cari Data berdasarkan Rentang Nilai (80 -
90)\n";
        cout << "5. Keluar\n";
        cout << "Pilihan Anda: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Masukkan NIM (10 digit): ";
                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;
}
```

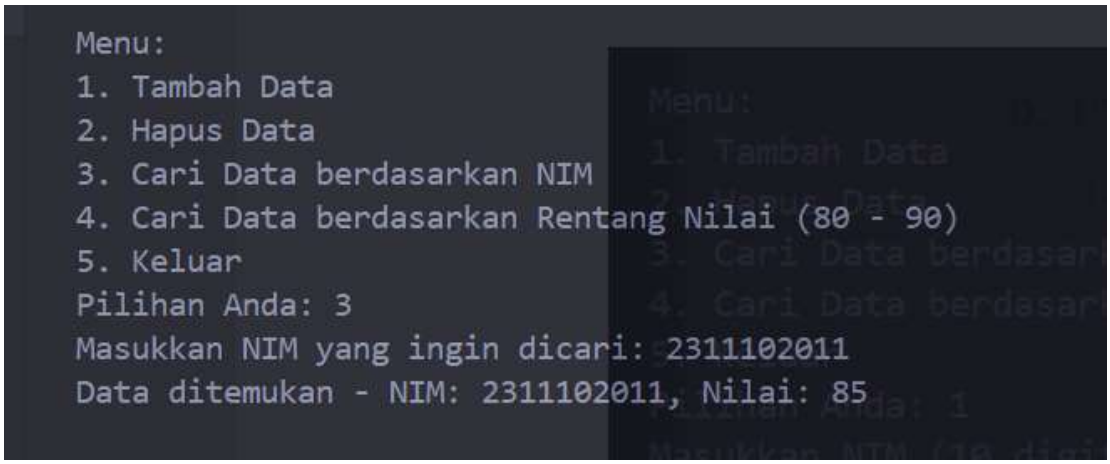
SCREENSHOOT PROGRAM



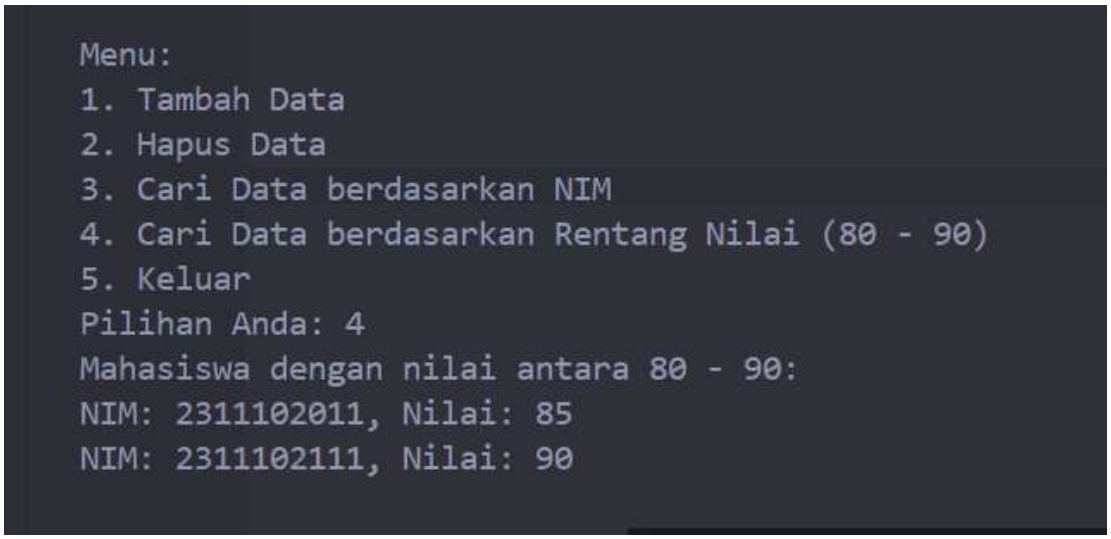
Program ketika menambahkan data mahasiswa



Program Ketika menhapus data mahasiswa



Program ketika mencari data berdasarkan NIM



Program ketika mencari data berdasarkan rentang nilai diantara 80-90

DESKRIPSI PROGRAM

Ukuran Tabel Hash: menggunakan konstanta hashTableSize untuk menentukan ukuran tabel hash. Dalam contoh ini, kita menggunakan ukuran 100. Class Tabel Hash: Class HashTable digunakan untuk mengimplementasikan tabel hash. Ini memiliki beberapa fungsi utama, seperti menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan 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 Table adalah struktur data yang mengorganisir data dalam pasangan kunci-nilai.
2. Terdiri dari array (atau vektor) dan fungsi hash.
3. Fungsi hash membuat pemetaan antara kunci dan nilai melalui rumus matematika yang dikenal sebagai fungsi hash.
4. Insertion: Memasukkan data baru ke dalam hash table dengan memanggil fungsi hash untuk menentukan posisi bucket yang tepat, dan kemudian menambahkan data ke bucket tersebut.

DAFTAR PUSTAKA

Karumanchi, N. (2016). Data Structures and algorithms made easy: Concepts, problems, Interview Questions. CareerMonk Publications.