

LAPORAN PRAKTIKUM STRUKTUR DATA DAN PEMROGAMAN

MODUL 5 HASH TABLE



Disusun Oleh :

Baharuddin Barkah Pratama

2311102321

IF-11-A

Dosen Pengampu :

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

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAD INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PRWOKERTO
PURWOKERTO
2024**

BAB 1

TUJUAN PRAKTIKUM

- a. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
- b.** Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

BAB II

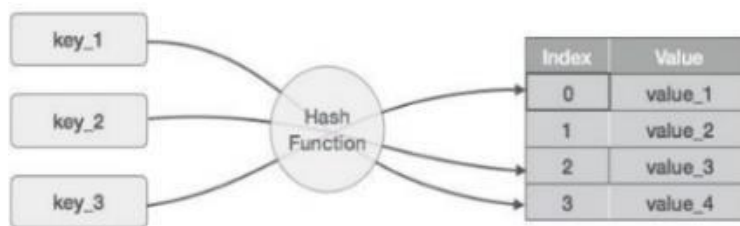
DASAR TEORI

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 memetakannya 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

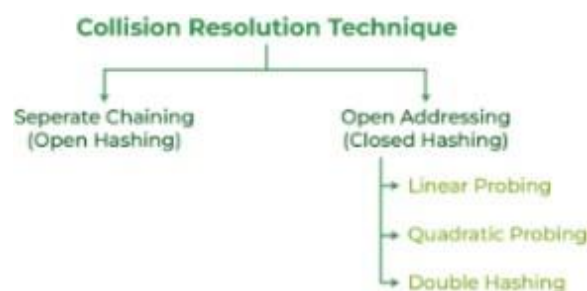
Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

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. Ada dua teknik untuk menyelesaikan masalah ini diantaranya :



1. Open Hashing

Metode chaining mengatasi collision dengan cara menyimpan semua item data dengan nilai indeks yang sama ke dalam sebuah linked list. Setiap node pada linked list merepresentasikan satu item data. Ketika ada pencarian atau penambahan item data, pencarian atau penambahan dilakukan pada linked list yang sesuai dengan indeks yang telah dihitung dari kunci yang di hash. Ketika linked list memiliki banyak node, pencarian atau penambahan item data menjadi lambat, karena harus mencari di seluruh linked list. Namun, chaining dapat mengatasi jumlah item data yang besar dengan efektif, karena keterbatasan array dihindari.

2. Closed Hashing

- **Linear Probing**

Pada saat terjadi collision, maka akan mencari posisi yang kosong di bawah tempat terjadinya collision, jika masih penuh terus ke bawah, hingga ketemu tempat yang kosong. Jika tidak ada tempat yang kosong berarti HashTable sudah penuh.

- **Quadratic Probing**

Penanganannya hampir sama dengan metode linear, hanya lompatannya tidak satu-satu, tetapi quadratic (12, 22, 32, 42, ...)

- **Double Hashing**

Pada saat terjadi collision, terdapat fungsi hash yang kedua untuk menentukan posisinya kembali

BAB III

GUIDED

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;
        }
    }
}

```

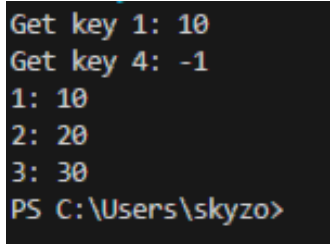
```

        }
    }
};

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
PS C:\Users\skyzo>

```

DESKRIPSI PROGRAM

Kode di atas menggunakan array dinamis “table” untuk menyimpan bucket dalam hash table. Setiap bucket diwakili oleh sebuah linked list dengan setiap node merepresentasikan satu item data. Fungsi hash sederhana hanya menggunakan modulus untuk memetakan setiap input kunci ke nilai indeks array.

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()
    {

```

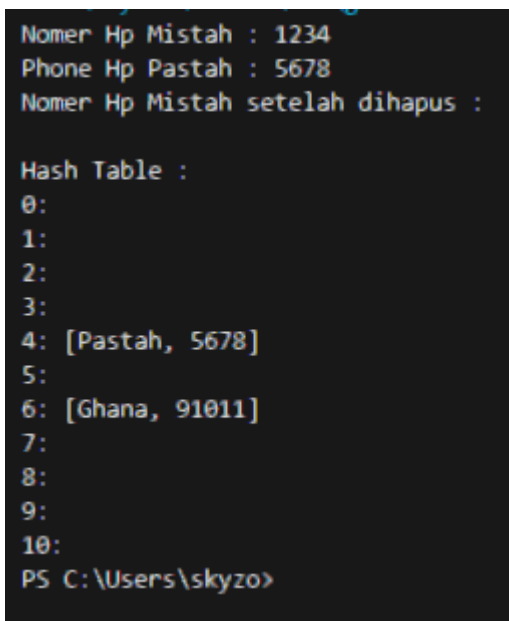
```

        for (int i = 0; i < TABLE_SIZE; i++)
        {
            cout << i << ": ";
            for (auto pair : table[i])
            {
                if (pair != nullptr)
                {
                    cout << "[" << pair->name << ", " << pair->phone_number
<< "]\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:
PS C:\Users\skyzo>

```

DESKRIPSI PROGRAM

Pada program di atas, class `HashNode` merepresentasikan setiap node dalam hash table, yang terdiri dari nama dan nomor telepon karyawan. Class `HashMap` digunakan untuk mengimplementasikan struktur hash table dengan menggunakan vector yang menampung pointer ke `HashNode`. Fungsi `hashFunc` digunakan untuk menghitung nilai hash dari nama karyawan yang diberikan, dan fungsi `insert`

digunakan untuk menambahkan data baru ke dalam hash table. Fungsi remove digunakan untuk menghapus data dari hash table, dan fungsi searchByName digunakan untuk mencari nomor telepon dari karyawan dengan nama yang diberikan.

BAB IV

UNGUIDED

UNGUIDED SOURCE CODE

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

using namespace std;

// Class Mahasiswa untuk merepresentasikan entitas mahasiswa
class Mahasiswa
{
public:
    string nim;
    int nilai;

    Mahasiswa(string nim, int nilai)
    {
        this->nim = nim;
        this->nilai = nilai;
    }
};

// Class HashTable untuk implementasi hash table
class HashTable
{
private:
    static const int table_size = 10; // Ukuran tabel hash
    vector<Mahasiswa *> table[table_size];

public:
    // Fungsi hash
    int hash_function(string nim)
    {
        int sum = 0;
        for (char c : nim)
        {
            sum += c;
        }
        return sum % table_size;
    }

    // Menambah data mahasiswa
    void tambah_data(Mahasiswa *mahasiswa)
    {
        int index = hash_function(mahasiswa->nim);
        table[index].push_back(mahasiswa);
    }

    // Menghapus data mahasiswa berdasarkan NIM
    void hapus_data(string nim)
    {
        int index = hash_function(nim);

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

```

        {
            if ((*it)->nim == nim)
            {
                delete *it;
                table[index].erase(it);
                break;
            }
        }
    }

// Mencari data mahasiswa berdasarkan NIM
Mahasiswa *cari_berdasarkan_nim(string nim)
{
    int index = hash_function(nim);

    for (Mahasiswa *mahasiswa : table[index])
    {
        if (mahasiswa->nim == nim)
        {
            return mahasiswa;
        }
    }
    return nullptr;
}

// Mencari data mahasiswa berdasarkan rentang nilai (80 - 90)
vector<Mahasiswa *> cari_berdasarkan_rentang_nilai(int nilai_min, int
nilai_max)
{
    vector<Mahasiswa *> hasil_pencarian;

    for (int i = 0; i < table_size; i++)
    {
        for (Mahasiswa *mahasiswa : table[i])
        {
            if (mahasiswa->nilai >= nilai_min && mahasiswa->nilai <=
nilai_max)
            {
                hasil_pencarian.push_back(mahasiswa);
            }
        }
    }

    return hasil_pencarian;
}
};

// Fungsi main
int main()
{
    HashTable hash_table;

    while (true)
    {
        cout << "Made by : Baharuddin Barkah Pratama" << endl; cout
<< "Nim : 2311102321" << endl;
        cout << "\nPilihan Menu:" << endl;
        cout << "1. Tambah Data Mahasiswa" << endl;
        cout << "2. Hapus Data Mahasiswa" << endl;
        cout << "3. Cari Mahasiswa Berdasarkan NIM" << endl;
        cout << "4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)" <<
endl;
    }
}

```

```

cout << "5. Keluar" << endl;

int pilihan;
cout << "Masukkan pilihan Anda: ";
cin >> pilihan;

if (pilihan == 1)
{
    string nim;
    int nilai;
    cout << "Masukkan NIM: ";
    cin >> nim;
    cout << "Masukkan nilai: ";
    cin >> nilai;
    Mahasiswa *mahasiswa_baru = new Mahasiswa(nim, nilai);
    hash_table.tambah_data(mahasiswa_baru);
    cout << "Data mahasiswa berhasil ditambahkan." << endl;
}
else if (pilihan == 2)
{
    string nim;
    cout << "Masukkan NIM mahasiswa yang akan dihapus: ";
    cin >> nim;
    hash_table.hapus_data(nim);
    cout << "Data mahasiswa dengan NIM " << nim << " telah dihapus."
<< endl;
}

else if (pilihan == 3)
{
    string nim;
    cout << "Masukkan NIM mahasiswa yang akan dicari: ";
    cin >> nim;
    Mahasiswa *mahasiswa = hash_table.cari_berdasarkan_nim(nim);
    if (mahasiswa != nullptr)
    {
        cout << "Mahasiswa dengan NIM " << nim << " ditemukan.
Nilai: " << mahasiswa->nilai << endl;
    }
    else
    {
        cout << "Mahasiswa dengan NIM " << nim << " tidak
ditemukan." << endl;
    }
}
else if (pilihan == 4)
{
    vector<Mahasiswa*> mahasiswa_ditemukan =
hash_table.cari_berdasarkan_rentang_nilai(80, 90);
    if (!mahasiswa_ditemukan.empty())
    {
        cout << "Mahasiswa dengan nilai antara 80 dan 90:" << endl;
        for (Mahasiswa *mahasiswa : mahasiswa_ditemukan)
        {
            cout << "NIM: " << mahasiswa->nim << " Nilai: " <<
mahasiswa->nilai << endl;
        }
    }
    else
    {
        cout << "Tidak ada mahasiswa dengan nilai antara 80 dan
90." << endl;
    }
}
}

```

```

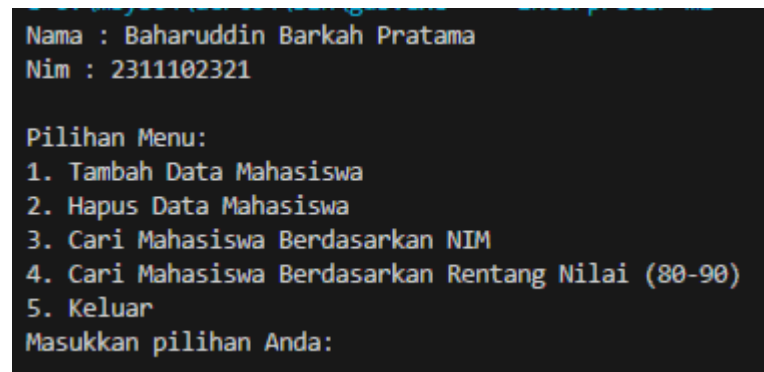
    }
}
else if (pilihan == 5)
{
    cout << "Program selesai." << endl;
    break;
}
else
{
    cout << "Pilihan tidak valid. Silakan masukkan pilihan yang
benar." << endl;
}
}

return 0;
}

```

SCREENSHOOT PROGRAM

- a. Setiap mahasiswa memiliki NIM dan nilai.
- b. Program memiliki tampilan pilihan menu berisi poin C.



```

Nama : Baharuddin Barkah Pratama
Nim : 2311102321

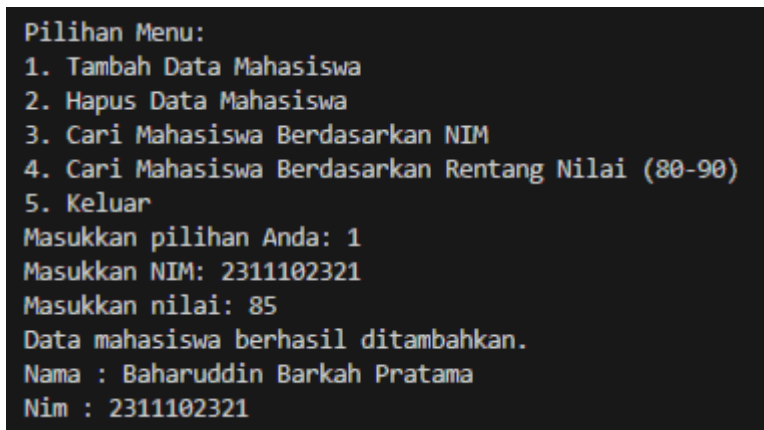
Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda:

```

DESKRIPSI PROGRAM

Yang pertama bagian Nama dan Nim, bagian tersebut mencetak Nama dan Nim dari si pengguna yang menjalankan program. Program tersebut juga terdapat tambah data mahasiswa, hapus data mahasiswa, cari mahasiswa berdasarkan nim, cari mahasiswa berdasarkan rentang nilai (80-90), dan yang terakhir yaitu keluar.

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



```

Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda: 1
Masukkan NIM: 2311102321
Masukkan nilai: 85
Data mahasiswa berhasil ditambahkan.
Nama : Baharuddin Barkah Pratama
Nim : 2311102321

```

DESKRIPSI

Pengguna memasukkan nomer 1 yaitu tambah data mahasiswa, Program akan menambahkan informasi data mahasiswa ke dalam system. Proses penambahan data mahasiswa dalam program ini melibatkan beberapa poin diantaranya yaitu, menerima input nim dan nilai dari pengguna, menambahkan objek mahasiswa ke dalam hash table menggunakan fungsi hash untuk menentukan posisi penyimpanan yang sesuai.

Menghapus data

```
Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda: 2
Masukkan NIM mahasiswa yang akan dihapus: 2311102321
Data mahasiswa dengan NIM 2311102321 telah dihapus.
Nama : Baharuddin Barkah Pratama
Nim : 2311102321
```

DESKRIPSI

Pengguna memasukkan nomer 2 yaitu tambah data mahasiswa, Program akan menghapus data mahasiswa di dalam system. Fungsi hapus data di dalam program tersebut bertujuan untuk mencari dan menghapus data mahasiswa berdasarkan NIM dari table hash. Fungsi ini memastikan bahwa data yang dihapus dikeluarkan dari memori dan dari bucket tabel hash untuk mencegah kebocoran memori serta menjaga integritas data.

Cari Mahasiswa Berdasarkan NIM

```
Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda: 3
Masukkan NIM mahasiswa yang akan dicari: 2311102321
Mahasiswa dengan NIM 2311102321 tidak ditemukan.
Nama : Baharuddin Barkah Pratama
Nim : 2311102321
```

DESKRIPSI

Di bagian 'cari_berdasarkan_nim' bertugas mencari data mahasiswa berdasarkan NIM dengan cara yang pertama yaitu menghitung indeks hash untuk NIM yang diberikan menggunakan hash_function, terus vector yang ada di dalam table hash mencari untuk menemukan mahasiswa dengan NIM yang cocok. Jika NIM ditemukan, maka mengembalikan pointer ke objek 'Mahasiswa'. Jika NIM tidak ditemukan, maka mengembalikan 'nullptr'.

Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)

```
Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda: 1
Masukkan NIM: 2311102321
Masukkan nilai: 88
Data mahasiswa berhasil ditambahkan.
Nama : Baharuddin Barkah Pratama
Nim : 2311102321

Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda: 4
Mahasiswa dengan nilai antara 80 dan 90:
NIM: 2311102321 Nilai: 88
Nama : Baharuddin Barkah Pratama
Nim : 2311102321
```

DESKRIPSI

Di bagian ini program bertugas untuk mencari mahasiswa yang rentang nilainya 80 sampai 90. Jika program menemukan mahasiswa maka nama dan NIM mereka akan ditampilkan. Dan jika mahasiswa tidak ditemukan maka program akan menampilkan “Tidak ada mahasiswa dengan nilai 80-90”. Pemanggilan nilai 80-90 itu dipanggil dengan parameter

Keluar

```
Pilihan Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Mahasiswa Berdasarkan NIM
4. Cari Mahasiswa Berdasarkan Rentang Nilai (80-90)
5. Keluar
Masukkan pilihan Anda: 5
Program selesai.
PS C:\Users\skyzo>
```

DESKRIPSI

Jika user menginputkan nomor 5 yaitu keluar maka program akan menampilkan “Program selesai.”

BAB V

KESIMPULAN

KESIMPULAN

Hash table merupakan struktur data yang istimewa karena menawarkan cara penyimpanan dan pengambilan data yang sangat efisien, dengan waktu akses rata-rata yang luar biasa cepat, yaitu $O(1)$. Kecepatan ini dimungkinkan dengan penggunaan fungsi hash yang memetakan kunci ke indeks dalam sebuah array. Hasilnya, data terorganisir dengan rapi dan mudah diakses. Meskipun hash table memiliki potensi kendala berupa tabrakan (collision), yaitu ketika dua kunci berbeda dipetakan ke indeks yang sama, teknik seperti chaining dan open addressing dapat diterapkan untuk mengatasinya. Keunggulan hash table ini menjadikannya pilihan populer dalam berbagai aplikasi, seperti database, caching, dan pengindeksan. Di aplikasi-aplikasi tersebut, hash table berperan penting dalam menangani operasi pencarian, penyisipan, dan penghapusan data dengan efisiensi yang tinggi.

DAFTAR PUSTAKA

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