

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL 5**



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

**Disusun oleh:**

**ABDA FIRAS RAHMAN**

**2311102049**

**IF-11-B**

**PROGRAM STUDI S1 INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

# BAB I

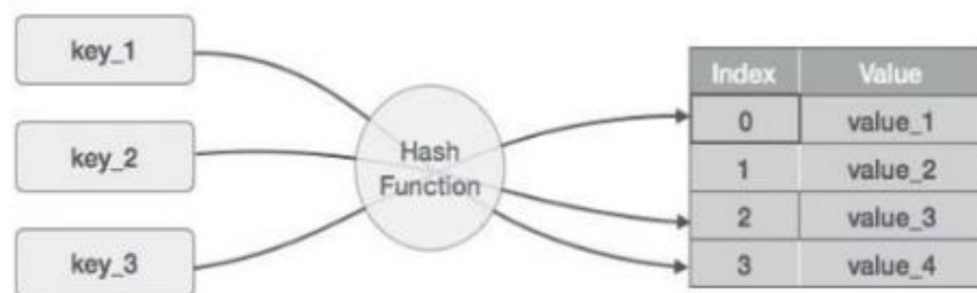
## DASAR TEORI

### HASH TABLE

#### 1. Pengertian

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.



## **2. Fungsi Hash Table**

Fungsi utamanya pada data adalah mempercepat proses akses data. Hal ini berkaitan dengan peningkatan data dalam jumlah besar yang diproses oleh jaringan data global dan lokal. Hash table adalah solusi untuk membuat proses akses data lebih cepat dan memastikan bahwa data dapat dipertukarkan dengan aman.

## **3. 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.

## **4. Teknik Hash Table**

### **1) Hashing**

Hashing merupakan sebuah proses mengganti kunci yang

diberikan atau string karakter menjadi nilai lain. Penggunaan hashing paling populer adalah pada hash table. Hash table menyimpan pasangan kunci dan nilai dalam daftar yang dapat diakses melalui indeksnya. Karena pasangan kunci dan nilai tidak terbatas, maka fungsinya akan memetakan kunci ke ukuran tabel dan kemudian nilainya menjadi indeks untuk elemen tertentu.

## **2) Probing**

- **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.

## GUIDED

### 1. Guided 1

```
#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;
}

```

### OUTPUT NYA:

The screenshot shows a Windows environment. In the background, a Command Prompt window displays the following commands and output:  

```
PS C:\Users\LENOVO\OneDrive\Desktop> cd ..  
PS C:\Users\LENOVO\OneDrive\Desktop> cd LGORITMA\PRAKTIKUM STRUKTUR DATA  
PS C:\Users\LENOVO\OneDrive\Desktop> cd LGORITMA\PRAKTIKUM STRUKTUR DATA\MEN FIRAS KULIAH\DOKUMEN  
PS C:\Users\LENOVO\OneDrive\Desktop> cd ..  
PS C:\Users\LENOVO\OneDrive\Desktop> cd ul 5 praktikum strukda  
PS C:\Users\LENOVO\OneDrive\Desktop> Get key 1: 10  
Get key 4: -1  
1: 10  
2: 20  
3: 30
```

  
In the foreground, a Notepad application window titled "Untitled - Notepad" is open. It contains the following text:  

```
NAMA      : ABDA FIRAS RAHMAN  
NIM       : 2311102049  
KELAS     : IF-11-B|
```

  
The status bar at the bottom of the Notepad window indicates "Ln 3, Col 16 | 57 characters | 100% | Window | UTF-8".

### 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.

## 2. GUIDED 2

```
#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;

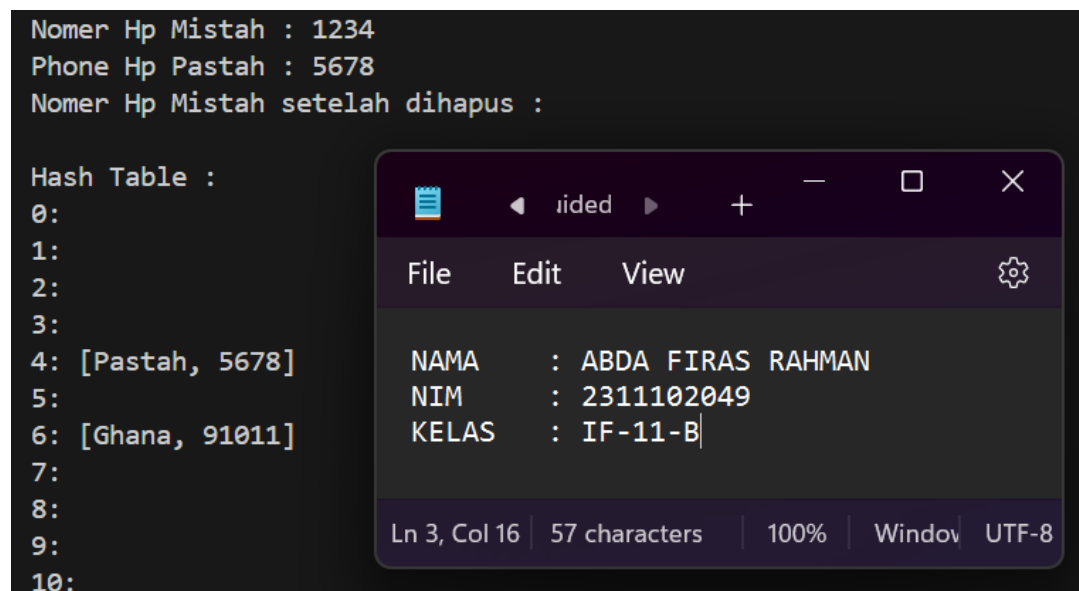
    cout << "Phone Hp Pastah : " <<
employee_map.searchByName("Pastah") <<
endl;

    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;
}

```

## OUTPUT NYA:



```

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:

```

File Edit View

NAMA : ABDA FIRAS RAHMAN  
NIM : 2311102049  
KELAS : IF-11-B

Ln 3, Col 16 | 57 characters | 100% | Window UTF-8

## DESKRIPSI PORGRAM:

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.

### UNGUIDED

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
const int TABLE_SIZE = 11;
string nim;
int nilai;
class HashNode
{
public:
    string nim;
    int nilai;
    HashNode(string nim, int nilai)
    {
        this->nim = nim;
        this->nilai = nilai;
    }
};
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 nim, int nilai)
    {
        int hash_val = hashFunc(nim);
        for (auto node : table[hash_val])
        {
            if (node->nim == nim)
            {
                node->nilai = nilai;
                return;
            }
        }
        table[hash_val].push_back(new
HashNode(nim, nilai));
    }
    void remove(string nim)
    {
        int hash_val = hashFunc(nim);
        for (auto it =

```



```

table[hash_val].begin(); it !=
table[hash_val].end(); it++)
    {
        if ((*it)->nim == nim)
        {
            table[hash_val].erase(it);
            return;
        }
    }
}

int search(string nim)
{
    int hash_val = hashFunc(nim);
    for (auto node : table[hash_val])
    {
        if (node->nim == nim)
        {
            return node->nilai;
        }
    }
    return -1; // return -1 if NIM is
not found
}

void findNimByScore(int minScore, int
maxScore)
{
    bool found = false;
    for (int i = 0; i < TABLE_SIZE;
i++)

```

```

        {
            for (auto pair : table[i])
            {
                if (pair != nullptr &&
pair->nilai >= minScore && pair->nilai <=
maxScore)
                    {
                        cout << pair->nim << "
memiliki nilai " << pair->nilai << endl;
                        found = true;
                    }
            }
        }
        if (!found)
        {
            cout << "Tidak ada mahasiswa
yang memiliki nilai antara " << minScore <<
" and " << maxScore << endl;
        }
    }
    void print()
    {
        for (int i = 0; i < TABLE_SIZE;
i++)
        {
            cout << i << ": ";
            for (auto pair : table[i])
            {
                if (pair != nullptr)

```

```

        {
            cout << "[" << pair-
>nim << ", " << pair->nilai << "];"
        }
    }
    cout << endl;
}
};

```

```

int main()
{
    HashMap data;
    string NIM;
    int nilai_mhs;
    while (true)
    {
        int menu;
        cout << "\nMenu" << endl;
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Mencari data
berdasarkan NIM" << endl;
        cout << "4. Mencari data
berdasarkan nilai" << endl;
        cout << "5. Cetak data" << endl;
        cout << "6. Exit" << endl;
        cout << "Masukkkkan pilihan : ";
        cin >> menu;
    }
}

```

```

switch (menu)
{
case 1:
    cout << "Masukkan NIM : ";
    cin >> NIM;
    cout << "Masukkan nilai : ";
    cin >> nilai_mhs;
    data.insert(NIM, nilai_mhs);
    break;
case 2:
    cout << "Masukkan NIM yang akan
dihapus : ";
    cin >> NIM;
    data.remove(NIM);
    cout << "Data berhasil dihapus"
<< endl;
    break;
case 3:
    cout << "Masukkan NIM yang akan
dicari : ";
    cin >> NIM;
    cout << "NIM " << NIM << "
memiliki nilai " << data.search(NIM) <<
endl;
    break;
case 4:
    int a, b;
    cout << "Masukkan rentang nilai
minimal : ";

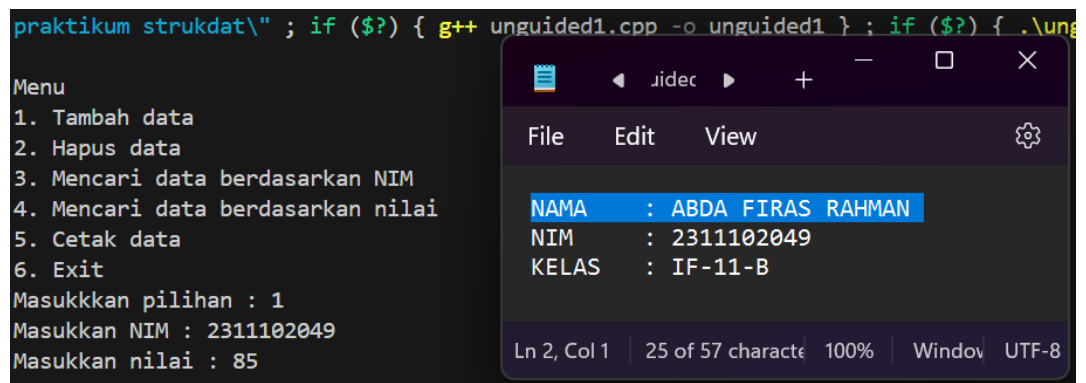
```

```

        cin >> a;
        cout << "Masukkan rentang nilai
maksimal : ";
        cin >> b;
        data.findNimByScore(a, b);
        break;
case 5:
        data.print();
        break;
case 6:
        return 0;
default:
        cout << "Menu tidak tersedia!"
<< endl;
        break;
    }
}
}

```

### OUTPUTNYA:



The screenshot shows a terminal window on the left and a JDE editor window on the right. The terminal window displays the program's output, which includes a menu with six options: 1. Tambah data, 2. Hapus data, 3. Mencari data berdasarkan NIM, 4. Mencari data berdasarkan nilai, 5. Cetak data, and 6. Exit. The user has selected option 1, entered a NIM of 2311102049, and entered a value of 85. The JDE editor window shows the program's output, which is the same as the terminal window's output.

```

praktikum strukdat\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) { .\ung
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2311102049
Masukkan nilai : 85

```

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 2
Masukkan NIM yang akan dihapus : 2311102049
Data berhasil dihapus
```

File Edit View

NAMA : ABDA FIRAS RAHMAN  
NIM : 2311102049  
KELAS : IF-11-B

Ln 1, Col 25 | 57 characters | 100% | Window | UTF-8

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 3
Masukkan NIM yang akan dicari : 2311102049
NIM 2311102049 memiliki nilai -1
```

File Edit View

NAMA : ABDA FIRAS RAHMAN  
NIM : 2311102049  
KELAS : IF-11-B

Ln 1, Col 25 | 57 characters | 100% | Window | UTF-8

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 4
Masukkan rentang nilai minimal : 80 90
Masukkan rentang nilai maksimal : Tidak ada mahasiswa yang memiliki nilai antara 80 and 90
```

File Edit View

NAMA : ABDA FIRAS RAHMAN  
NIM : 2311102049  
KELAS : IF-11-B

Ln 1, Col 25 | 57 characters | 100% | Windows (C | UTF-8

```
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 5
0:
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
```

File Edit View

NAMA : ABDA FIRAS RAHMAN  
NIM : 2311102049  
KELAS : IF-11-B

Ln 1, Col 25 | 57 characters | 100% | Windows (C | UTF-8

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkan pilihan : 6

File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 1, Col 25 | 57 characters | 100% | Windows (C | UTF-8
```

## DESKRIPSI PROGRAM

Program ini adalah implementasi struktur data Hash Table dalam C++ yang digunakan untuk menyimpan dan mengelola data nilai mahasiswa berdasarkan Nomor Induk Mahasiswa (NIM). Berikut penjelasan lebih rinci:

- A. Program mendefinisikan sebuah kelas HashNode yang merupakan node untuk menyimpan data NIM dan nilai mahasiswa.
- B. Kelas HashMap adalah implementasi dari struktur data Hash Table. Kelas ini memiliki beberapa fungsi utama:
  - hashFunc(string key): Fungsi untuk menghitung nilai hash dari NIM menggunakan penjumlahan nilai ASCII dari karakter NIM.
  - insert(string nim, int nilai): Fungsi untuk menambahkan data NIM dan nilai ke dalam Hash Table. Jika NIM sudah ada, nilai akan diperbarui.
  - remove(string nim): Fungsi untuk menghapus data NIM dari Hash Table.
  - search(string nim): Fungsi untuk mencari nilai berdasarkan NIM. Jika NIM tidak ditemukan, akan mengembalikan nilai -1.
  - findNimByScore(int minScore, int maxScore): Fungsi untuk mencari dan mencetak NIM beserta nilainya yang berada dalam rentang nilai tertentu.
  - print(): Fungsi untuk mencetak seluruh isi Hash Table.
- C. Dalam fungsi main(), program menampilkan menu pilihan untuk user, seperti menambah data, menghapus data, mencari data berdasarkan NIM, mencari data berdasarkan rentang nilai, mencetak seluruh data, dan keluar dari program.
- D. Ketika user memilih opsi 1 (Tambah data), program akan meminta user memasukkan NIM dan nilai, kemudian memanggil fungsi insert() untuk menambahkan data ke Hash Table.

- E.** Ketika user memilih opsi 2 (Hapus data), program akan meminta user memasukkan NIM, kemudian memanggil fungsi `remove()` untuk menghapus data dari Hash Table.
- F.** Ketika user memilih opsi 3 (Mencari data berdasarkan NIM), program akan meminta user memasukkan NIM, kemudian memanggil fungsi `search()` untuk mencari nilai berdasarkan NIM tersebut.
- G.** Ketika user memilih opsi 4 (Mencari data berdasarkan nilai), program akan meminta user memasukkan rentang nilai minimal dan maksimal, kemudian memanggil fungsi `findNimByScore()` untuk mencari dan mencetak NIM beserta nilainya yang berada dalam rentang nilai tersebut.
- H.** Ketika user memilih opsi 5 (Cetak data), program akan memanggil fungsi `print()` untuk mencetak seluruh isi Hash Table.
- I.** Ketika user memilih opsi 6 (Exit), program akan keluar dari perulangan dan mengakhiri eksekusi.

Program ini menggunakan Hash Table untuk menyimpan data NIM dan nilai mahasiswa sehingga operasi pencarian, penyisipan, dan penghapusan data dapat dilakukan dengan efisien.



## **DAFTAR PUSTAKA**

PENGERTIAN HASH TABLE DAN CARA PENGGUNAANYA :

<https://algorit.ma/blog/hash-table-adalah-2022/>