LAPORAN PRAKTIKUM
MODUL V ALGORITMA DAN STRUKTUR DATA
”HASH TABLE ”

Disusun oleh :

Yasvin Syahgana (2311102065)

Dosen :

Wahyu Andi Syahputra, S.pd.,M.Eng

PROGRAM STUDI

S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA INSTITUT TEKNOLOGI

TELKOM PURWOKERTO 2024

BAB I

TUJUAN PRAKTIKUM

Tujuan parktikum adalah sebagai berikut :

- Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
- Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

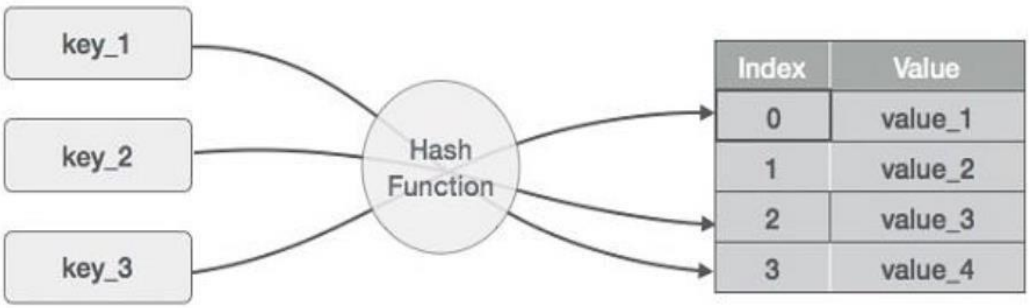
BAB II

DASAR TEORI

A. Pengertian Hash Table

Hash Table adalah struktur data yang digunakan untuk menyimpan pasangan kunci/nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vector) dan fungsi hash. Dengan menggunakan fungsi hash yangbaik hashing dapat berjalan dengan baik. 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 index 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



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 adalahrepresentasi dari string karakter asli tetapi biasanya lebih kecil dari aslinya.

Tipe fungsi hash :

- Division Method.
- Mid Square Method.
- Folding Method.
- Multiplication Method.

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:**

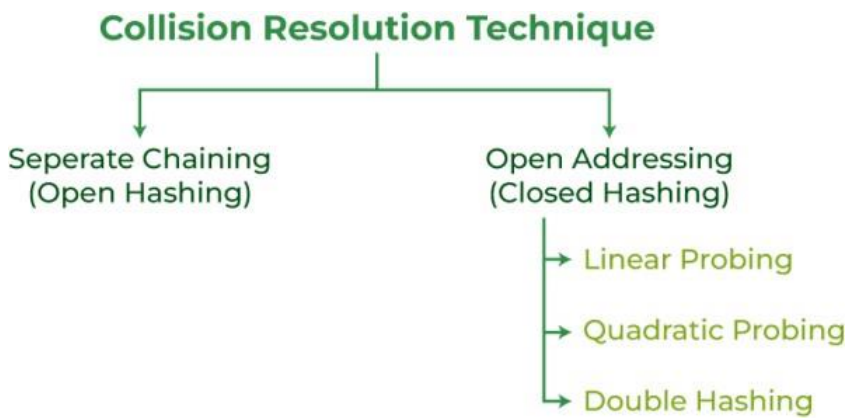
   Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yangditemukan.

5. **Traversal:**

   Melalui seluruh hash table untuk memproses semua data yang adadalam table.

D. Collision Resolution

Keterbatasan tabel hash adalah jika dua angka dimasukkan ke dalam fungsi hash menghasilkan nilai yang sama.ada 2 teknik untuk menyelesaikan masalah :



Guided 1

Source Code

```
//YASVIN SYAHGANA
//2311102065
//GUIDED 1 MODUL 5

#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key_2311102065) { return key_2311102065 %
MAX_SIZE; }
// Struktur data untuk setiap node
struct Node {
  int key_2311102065;
  int value_2311102065;
  Node *next_2311102065;
  Node(int key_2311102065, int value_2311102065) :
key_2311102065(key_2311102065),
value_2311102065(value_2311102065), next_2311102065(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_2311102065;
        delete temp;
      }
    }
    delete[] table;
  }
  // Insertion
  void insert(int key_2311102065, int value_2311102065) {
    int index = hash_func(key_2311102065);
    Node *current = table[index];
    while (current != nullptr) {
      if (current->key_2311102065 == key_2311102065) {
        current->value_2311102065 = value_2311102065;
```

```
        return;
      }
      current = current->next_2311102065;
    }
    Node *node = new Node(key_2311102065, value_2311102065);
    node->next_2311102065 = table[index];
    table[index] = node;
  }
  // Searching
  int get(int key_2311102065) {
    int index = hash_func(key_2311102065);
    Node *current = table[index];
    while (current != nullptr) {
      if (current->key_2311102065 == key_2311102065) {
        return current->value_2311102065;
      }
      current = current->next_2311102065;
    }
    return -1;
  }
  // Deletion
  void remove(int key_2311102065) {
    int index = hash_func(key_2311102065);
    Node *current = table[index];
    Node *prev = nullptr;
    while (current != nullptr) {
      if (current->key_2311102065 == key_2311102065) {
        if (prev == nullptr) {
          table[index] = current->next_2311102065;
        } else {
          prev->next_2311102065 = current->next_2311102065;
        }
        delete current;
        return;
      }
      prev = current;
      current = current->next_2311102065;
    }
  }
  // Traversal
  void traverse() {
    for (int i = 0; i < MAX_SIZE; i++) {
      Node *current = table[i];
      while (current != nullptr) {
        cout << current->key_2311102065 << ": " << current-
>value_2311102065 << endl;
        current = current->next_2311102065;
```

```
      }
    }
  }
};
int main() {
  HashTable ht;
  // Insertion
  ht.insert(1, 10);
  ht.insert(2, 20);
  ht.insert(3, 30);
  // Searching
  cout << "Get key_2311102065 1: " << ht.get(1) << endl;
  cout << "Get key_2311102065 4: " << ht.get(4) << endl;
  // Deletion
  ht.remove(4);
  // Traversal
  ht.traverse();
  cout<<endl;
  return 0;
}
```

Output

```
Guided1 }
Get key_2311102065 1: 10
Get key_2311102065 4: -1
1: 10
2: 20
3: 30
```

Deskripsi Program

1. Program ini menggunakan hash table untuk menyimpan data.
2. Setiap entri dalam hash table diwakili oleh node yang terdiri dari kunci (key) dan nilai (value).
3. Implementasi hash table dilakukan dengan menggunakan array dinamis dari pointer ke node.
4. Terdapat fungsi hash sederhana yang menghasilkan indeks dalam array berdasarkan kunci yang diberikan.
5. Program mendukung operasi dasar seperti penambahan (insertion), pencarian (searching), penghapusan (deletion), dan penelusuran

(traversal) pada hash table.

6. Operasi penambahan memasukkan pasangan kunci-nilai baru ke dalam hash table.
7. Operasi pencarian mencari nilai yang terkait dengan kunci yang diberikan.
8. Operasi penghapusan menghapus pasangan kunci-nilai dari hash table berdasarkan kunci yang diberikan.
9. Operasi penelusuran mencetak seluruh pasangan kunci-nilai yang disimpan dalam hash table.
10. Program juga mencakup contoh penggunaan hash table dengan memasukkan beberapa pasangan kunci-nilai, melakukan pencarian, dan penghapusan.

Overall, program ini menyajikan implementasi dasar dari hash table dalam C++, yang dapat digunakan sebagai dasar untuk pengembangan sistem yang lebih kompleks.

Guided 2

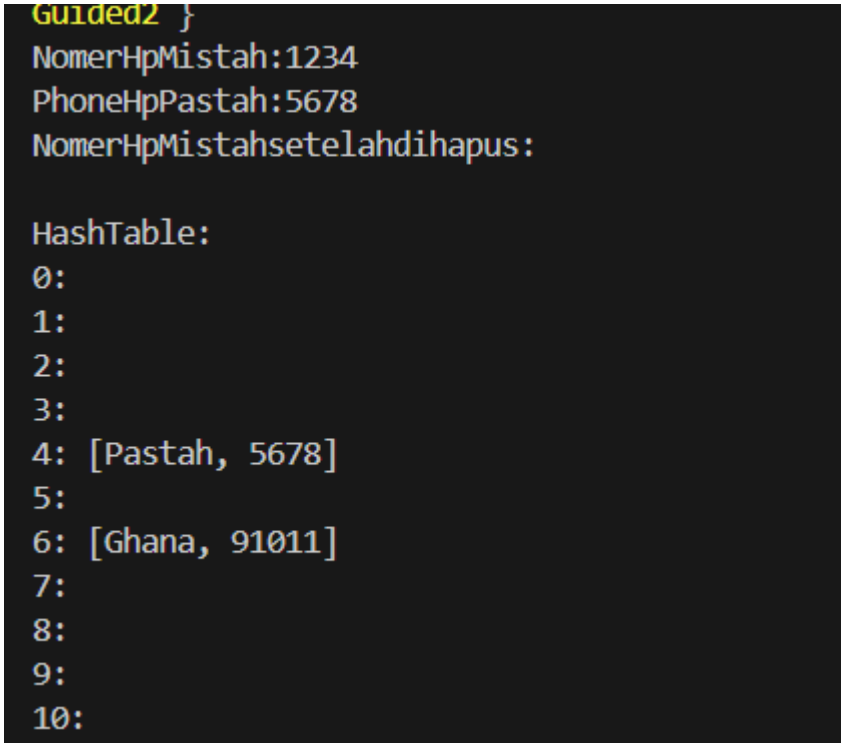
Source Code

```
//YASVIN SYAHGANA
//2311102065
//GUIDED 2 MODUL 5

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

using namespace std;
const int TABLE_SIZE = 11;
string nama_2311102065;
string phone_number_2311102065;
class HashNode {
public:
  string nama_2311102065;
  string phone_number_2311102065;
  HashNode(string nama_2311102065, string
phone_number_2311102065) {
    this->nama_2311102065 = nama_2311102065;
    this->phone_number_2311102065 = phone_number_2311102065;
```

```
  }
};
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 nama_2311102065, string
phone_number_2311102065) {
    int hash_val = hashFunc(nama_2311102065);
    for (auto node : table[hash_val]) {
      if (node->nama_2311102065 == nama_2311102065) {
        node->phone_number_2311102065 =
phone_number_2311102065;
        return;
      }
    }
    table[hash_val].push_back(new HashNode(nama_2311102065,
phone_number_2311102065));
  }
  void remove(string nama_2311102065) {
    int hash_val = hashFunc(nama_2311102065);
    for (auto it = table[hash_val].begin(); it !=
table[hash_val].end(); it++) {
      if ((*it)->nama_2311102065 == nama_2311102065) {
        table[hash_val].erase(it);
        return;
      }
    }
  }
  string searchByNama_2311102065(string nama_2311102065) {
    int hash_val = hashFunc(nama_2311102065);
    for (auto node : table[hash_val]) {
      if (node->nama_2311102065 == nama_2311102065) {
        return node->phone_number_2311102065;
      }
    }
    return "";
  }
  void print() {
```

```
    for (int i = 0; i < TABLE_SIZE; i++) {
      cout << i << ": ";
      for (auto pair : table[i]) {
        if (pair != nullptr) {
          cout << "[" << pair->nama_2311102065 << ", " <<
pair->phone_number_2311102065 << "]";
        }
      }
      cout << endl;
    }
  }
};
int main() {
  HashMap employee_map;
  employee_map.insert("Mistah", "1234");
  employee_map.insert("Pastah", "5678");
  employee_map.insert("Ghana", "91011");
  cout << "NomerHpMistah:" <<
employee_map.searchByNama_2311102065("Mistah") << endl;
  cout << "PhoneHpPastah:" <<
employee_map.searchByNama_2311102065("Pastah") << endl;
  employee_map.remove("Mistah");
  cout << "NomerHpMistahsetelahdihapus:" <<
employee_map.searchByNama_2311102065("Mistah") << endl <<
endl;
  cout << "HashTable:" << endl;
  employee_map.print();
  return 0;
}
```

Output



Deskripsi Program

Pada Program di atas adalah implementasi sederhana dari hash table dalam bahasa C++. Hash table digunakan untuk menyimpan pasangan kunci-nilai (key-value pairs) dengan efisiensi pencarian, penambahan, dan penghapusan yang tinggi.

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

```
//YASVIN SYAHGANA
//2311102065
//UNGUIDED 1 MODUL 5

#include <iostream>
#include <string>
using namespace std;

struct Mahasiswa {
  string name_gana;
  string nim_2311102065;
  int nilai_2311102065;
};

class HashTable {
private:
  static const int tableSize = 100;
  struct Node {
    Mahasiswa data;
    Node *next;
  };
  Node *hashtable[tableSize];

public:
  HashTable() {
    for (int i = 0; i < tableSize; i++) {
      hashtable[i] = nullptr;
    }
  }
```

```
  int hashFunction(string key) {
    int hash = 0;
    for (int i = 0; i < key.length(); i++) {
      hash += (int)key[i];
    }
    return hash % tableSize;
  }

  void addData(Mahasiswa mhs) {
    int index = hashFunction(mhs.nim_2311102065);
    Node *newNode = new Node;
    newNode->data = mhs;
    newNode->next = hashtable[index];
    hashtable[index] = newNode;
    cout << "Data mahasiswa berhasil ditambahkan" << endl;
  }

  void removeData(string nim_2311102065) {
    int index = hashFunction(nim_2311102065);
    Node *currentNode = hashtable[index];
    Node *previousNode = nullptr;
    while (currentNode != nullptr) {
      if (currentNode->data.nim_2311102065 == nim_2311102065)
{
        if (previousNode == nullptr) {
          hashtable[index] = currentNode->next;
        } else {
          previousNode->next = currentNode->next;
        }
        delete currentNode;
        cout << "Data mahasiswa dengan NIM " << nim_2311102065
<< " berhasil dihapus"
             << endl;
        return;
      }
      previousNode = currentNode;
      currentNode = currentNode->next;
    }
    cout << "Data mahasiswa dengan NIM " << nim_2311102065 <<
" tidak ditemukan" << endl;
  }

  void findDataByNIM_2311102065(string nim_2311102065) {
    int index = hashFunction(nim_2311102065);
    Node *temp = hashtable[index];
    while (temp != nullptr) {
```

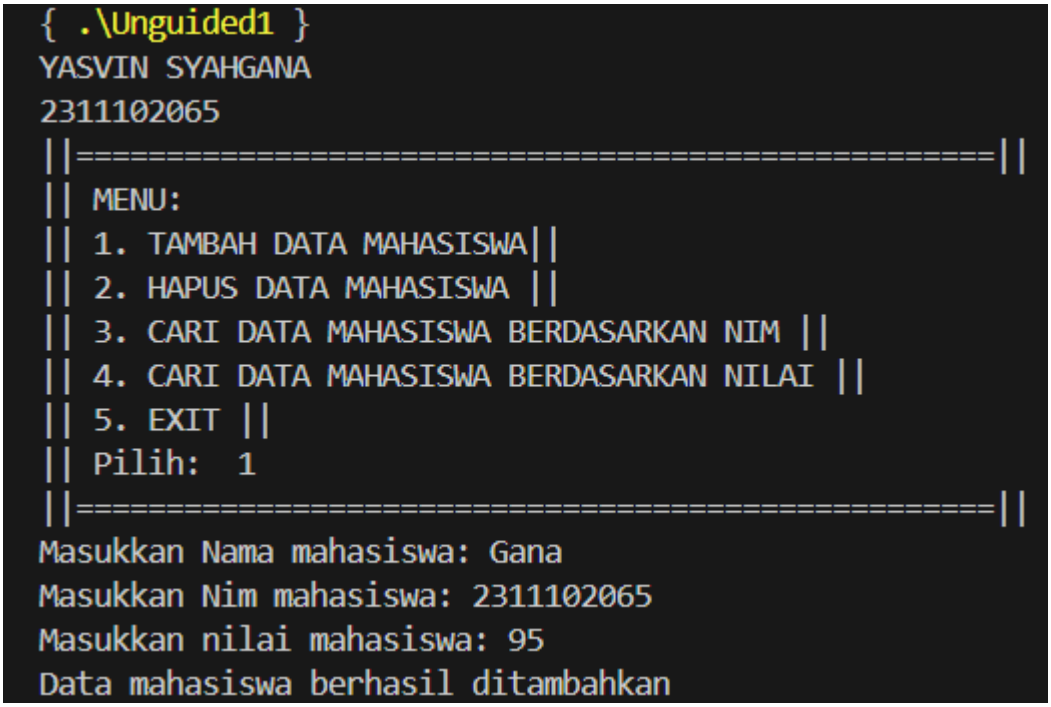
```
      if (temp->data.nim_2311102065 == nim_2311102065) {
        cout << "Data mahasiswa dengan NIM " << nim_2311102065
<< ":" << endl;
        cout << "Nama : " << temp->data.name_gana << endl;
        cout << "Nilai: " << temp->data.nilai_2311102065 <<
endl;
        return;
      }
      temp = temp->next;
    }
    cout << "Data mahasiswa dengan NIM " << nim_2311102065 <<
" tidak ditemukan" << endl;
  }

  void findDataByRange(int min, int max) {
    int count = 0;
    for (int i = 0; i < tableSize; i++) {
      Node *temp = hashtable[i];
      while (temp != NULL) {
        if (temp->data.nilai_2311102065 >= min && temp-
>data.nilai_2311102065 <= max) {
          if (count == 0) {
            cout << "Data mahasiswa dengan nilai antara " <<
min<< " dan " << max << " : " << endl;
          }
          cout << "NIM: " << temp->data.nim_2311102065 <<
endl;
          cout << "Nama: " << temp->data.name_gana << endl;
          cout << "Nilai: " << temp->data.nilai_2311102065 <<
endl;
          count++;
        }
        temp = temp->next;
      }
    }
    if (count == 0) {
      cout << "Tidak ditemukan data mahasiswa dengan nilai
antara " << min
           << " dan " << max << endl;
    }
  }
};

int main() {
  cout<<"YASVIN SYAHGANA"<<endl;
  cout<<"2311102065"<<endl;
  HashTable hashTable;
```

```
  int choice;
  do {
    cout <<
"||================================================||" <<
endl;
    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 NILAI ||"
<< endl;
    cout << "|| 5. EXIT ||" << endl;
    cout << "|| Pilih:  ";
    cin >> choice;
    cout <<
"||================================================||" <<
endl;

    switch (choice) {
    case 1: {
      Mahasiswa mhs;
      cout << "Masukkan Nama mahasiswa: ";
      cin.ignore();
      getline(cin, mhs.name_gana);
      cout << "Masukkan Nim mahasiswa: ";
      cin >> mhs.nim_2311102065;
      cout << "Masukkan nilai mahasiswa: ";
      cin >> mhs.nilai_2311102065;
      hashTable.addData(mhs);
      break;
    }
    case 2: {
      string nim_2311102065;
      cout << "Masukkan Nim mahasiswa yang ingin dihapus: ";
      cin >> nim_2311102065;
      hashTable.removeData(nim_2311102065);
      break;
    }
    case 3: {
      string nim_2311102065;
      cout << "Masukkan Nim mahasiswa yang ingin dicari: ";
      cin >> nim_2311102065;
      hashTable.findDataByNIM_2311102065(nim_2311102065);
      break;
    }
    case 4: {
```

```
      int min, max;
      cout << "Masukkan rentang nilai (minimal dan maksimal):
";
      cin >> min >> max;
      hashTable.findDataByRange(min, max);
      break;
    }
    case 5:
      cout << "Terima kasih" << endl;
      break;
    default:
      cout << "Pilihan tidak valid" << endl;
    }
  } while (choice != 5);

  return 0;
}
```

Screenshoot Program

1. **Setiap Mahasiswa Memiliki Nama dan Nim**



2. **Menambahkan Data Baru**

```
||=================================================||
|| MENU:
|| 1. TAMBAH DATA MAHASISWA||
|| 2. HAPUS DATA MAHASISWA ||
|| 3. CARI DATA MAHASISWA BERDASARKAN NIM ||
|| 4. CARI DATA MAHASISWA BERDASARKAN NILAI ||
|| 5. EXIT ||
|| Pilih:  1
||=================================================||
Masukkan Nama mahasiswa: Falah
Masukkan Nim mahasiswa: 2311102045
Masukkan nilai mahasiswa: 90
Data mahasiswa berhasil ditambahkan
||=================================================||
|| MENU:
|| 1. TAMBAH DATA MAHASISWA||
|| 2. HAPUS DATA MAHASISWA ||
|| 3. CARI DATA MAHASISWA BERDASARKAN NIM ||
|| 4. CARI DATA MAHASISWA BERDASARKAN NILAI ||
|| 5. EXIT ||
|| Pilih:  1
||=================================================||
Masukkan Nama mahasiswa: Alfin
Masukkan Nim mahasiswa: 2311102047
Masukkan nilai mahasiswa: 88
Data mahasiswa berhasil ditambahkan
```

3. **Menghapus Data**

```
||================================================||
|| MENU:
|| 1. TAMBAH DATA MAHASISWA||
|| 2. HAPUS DATA MAHASISWA ||
|| 3. CARI DATA MAHASISWA BERDASARKAN NIM ||
|| 4. CARI DATA MAHASISWA BERDASARKAN NILAI ||
|| 5. EXIT ||
|| Pilih:  2
||================================================||
Masukkan Nim mahasiswa yang ingin dihapus: 2311102047
Data mahasiswa dengan NIM 2311102047 berhasil dihapus
```

4. **Mencari Data Menggunakan Nim**

```
||================================================||
|| MENU:
|| 1. TAMBAH DATA MAHASISWA||
|| 2. HAPUS DATA MAHASISWA ||
|| 3. CARI DATA MAHASISWA BERDASARKAN NIM ||
|| 4. CARI DATA MAHASISWA BERDASARKAN NILAI ||
|| 5. EXIT ||
|| Pilih:  3
||================================================||
Masukkan Nim mahasiswa yang ingin dicari: 2311102045
Data mahasiswa dengan NIM 2311102045:
Nama : Falah
Nilai: 90
```

5. **Mencari Data Berdasarkan Rentan Nilai**

```
||=================================================||
|| MENU:
|| 1. TAMBAH DATA MAHASISWA||
|| 2. HAPUS DATA MAHASISWA ||
|| 3. CARI DATA MAHASISWA BERDASARKAN NIM ||
|| 4. CARI DATA MAHASISWA BERDASARKAN NILAI ||
|| 5. EXIT ||
|| Pilih:  4
||=================================================||
Masukkan rentang nilai (minimal dan maksimal): 90
95
Data mahasiswa dengan nilai antara 90 dan 95 :
NIM: 2311102065
Nama: Gana
Nilai: 95
NIM: 2311102045
Nama: Falah
Nilai: 90
```

Kesimpulan

Keuntungan utama dari hash table dibandingkan struktur data lainnya adalah efisiensi dan kecepatan. Waktu yang dibutuhkan untuk mengakses sebuah elemen cukup cepat sehingga bisa lebih diandalkan. Jadi, Anda tidak perlu memakan waktu atau usaha besar untuk menyimpan dan mencari data yang diperlukan.

Daftar Pustaka

1. https://ichi.pro/id/tabel-hash-dalam-struktur-data-dan-algoritma-264158795991298
2. Struktur-Data-Modul-Praktikum-11-Hashing-Table.pdf
3. https://m.youtube.com/watch?v=2_3fR-k-LzI
4. https://www.tutorialspoint.com/data_structures_algorithms/pdf/hash_data_structure.pdf
5. https://cplusplus.com/forum/
6. https://stackoverflow.com/