

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL V  
HASH TABLE**



**Disusun Oleh :  
WIDARI DWI HAYATI  
2311102060**

**Dosen :  
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.**

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

## A. Dasar Teori

### a. Pengertian Hash Table

Tabel hash adalah struktur data yang mengatur data menjadi pasangan nilai kunci. Ini terdiri dari dua komponen utama: array (atau vektor) dan fungsi hash. Fungsi hash adalah teknik yang mengubah serangkaian kunci dan nilai menjadi serangkaian indeks array. Array menyimpan data dalam slot yang disebut bucket. Setiap keranjang dapat berisi satu atau lebih item data. Fungsi hash digunakan untuk menghasilkan nilai unik dari setiap bagian data dan digunakan sebagai indeks ke dalam array. Dengan demikian, tabel hash memungkinkan pencarian data dalam waktu yang konstan ( $O(1)$ ) dalam skenario kasus terbaik.

Sistem tabel hash bekerja dengan mengambil kunci input dan memetakannya ke nilai indeks array menggunakan fungsi hash. Data disimpan pada posisi indeks array yang dihasilkan oleh fungsi hash. Ketika data perlu diambil, kunci input digunakan sebagai parameter fungsi hash dan posisi indeks dari array yang dihasilkan digunakan untuk mengambil data. Ketika tabrakan hash terjadi ketika dua atau lebih data mempunyai nilai hash yang sama, tabel hash menggunakan teknik yang disebut chaining untuk menyimpan data dalam slot yang sama.

### 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 (Chaining)**

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.

B. Guided  
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;

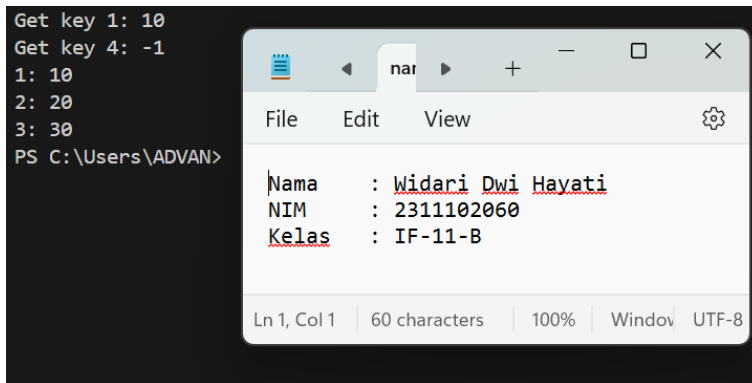
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

## Screenshots Output



The screenshot shows a terminal window on the left and a text editor window on the right. The terminal displays the output of a program: 'Get key 1: 10', 'Get key 4: -1', and a list of three items: '1: 10', '2: 20', and '3: 30'. The prompt is 'PS C:\Users\ADVAN>'. The text editor window, titled 'nar', shows a text file with the following content: 'Nama : Widari Dwi Hayati', 'NIM : 2311102060', and 'Kelas : IF-11-B'. The editor's status bar indicates 'Ln 1, Col 1', '60 characters', '100%', 'Window', and 'UTF-8'.

```
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\Users\ADVAN>
```

```
Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-B
```

## Deskripsi:

Kode program tersebut mengimplementasikan tabel hash. Tabel hash memiliki ukuran tetap 10, yang didefinisikan oleh konstanta `MAX_SIZE`. Fungsi `hash_func` adalah fungsi hash sederhana yang mengembalikan sisa bagi dari pembagian kunci input oleh `MAX_SIZE`. Struktur Node mewakili elemen tunggal dalam tabel hash, dengan kunci, nilai, dan pointer ke node berikutnya dalam rantai. Kelas `HashTable` berisi array pointer ke objek `Node`, yang membentuk basis tabel hash.

Fungsi `insert` menambahkan pasangan kunci-nilai baru ke tabel hash dengan menghitung indeks hash menggunakan fungsi `hash_func`, dan kemudian menelusuri rantai node pada indeks tersebut untuk melihat apakah kunci sudah ada. Jika demikian, nilai diperbarui; jika tidak, node baru dibuat dan ditambahkan ke depan rantai. Fungsi `get` mencari kunci dalam tabel hash dan mengembalikan nilai yang sesuai jika ditemukan. Fungsi ini melakukan ini dengan menghitung indeks hash dan kemudian menelusuri rantai node pada indeks tersebut hingga menemukan node dengan kunci yang sesuai. Jika tidak ada node seperti itu, maka mengembalikan -1. Fungsi `remove` menghapus pasangan kunci-nilai dari tabel hash dengan menghitung indeks hash dan kemudian menelusuri rantai node pada indeks tersebut hingga menemukan node dengan kunci yang sesuai. Jika node seperti itu ditemukan, maka dihapus dari rantai dan dihapus. Fungsi `traverse` mencetak semua pasangan kunci-nilai dalam tabel hash dengan menelusuri semua rantai dalam tabel.

Fungsi `main` mendemonstrasikan penggunaan tabel hash dengan menambahkan beberapa pasangan kunci-nilai, mencari kunci, menghapus kunci, dan kemudian menelusuri tabel untuk mencetak semua pasangan kunci-nilai yang tersisa.

## 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");
}

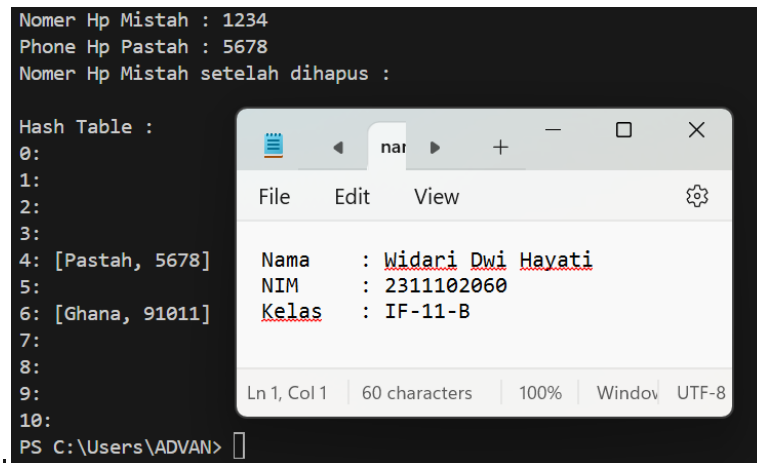
```

```

    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;
}

```

### Screenshots Output



The screenshot shows a terminal window with the following output:

```

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\ADVAN>

```

Overlaid on the terminal is a Notepad window titled 'nar' containing the following text:

```

Nama      : Widari Dwi Hayati
NIM       : 2311102060
Kelas    : IF-11-B

```

The Notepad window also shows a status bar at the bottom: 'Ln 1, Col 1 | 60 characters | 100% | Window | UTF-8'.

### Deskripsi:

Kode program tersebut memanfaatkan struktur data hash map dengan mengimplementasikan tabel hash dengan ukuran tetap 11. Program ini digunakan untuk menyimpan pasangan nama dan nomor telepon karyawan. Kelas HashNode mewakili elemen dalam hash map, dengan memuat nama dan nomor telepon. Kelas HashMap berisi array dari pointer ke objek HashNode, yang menyusun basis dari tabel hash.

Fungsi hashFunc menghitung indeks hash untuk nama yang diberikan dengan cara menjumlahkan nilai ASCII dari setiap karakter. Fungsi insert menambahkan pasangan nama-nomor telepon baru ke hash map dengan menghitung indeks hash menggunakan fungsi hashFunc, dan kemudian menelusuri rantai node pada indeks tersebut untuk melihat apakah nama sudah ada. Jika demikian, nomor telepon diperbarui; jika tidak, node baru dibuat dan ditambahkan ke depan rantai. Fungsi remove menghapus pasangan nama-nomor telepon dari hash map dengan menghitung indeks hash dan kemudian menelusuri rantai node pada indeks tersebut hingga menemukan node dengan nama yang sesuai. Jika node seperti itu ditemukan, maka dihapus dari rantai dan dihapus.

Fungsi `searchByName` mencari nama dalam hash map dan mengembalikan nomor telepon yang sesuai jika ditemukan. Fungsi ini melakukan ini dengan menghitung indeks hash dan kemudian menelusuri rantai node pada indeks tersebut hingga menemukan node dengan nama yang sesuai. Jika tidak ada node seperti itu, maka mengembalikan string kosong. Fungsi `print` mencetak semua pasangan nama-nomor telepon dalam hash map dengan menelusuri semua rantai dalam tabel.

Fungsi `main` menunjukkan penggunaan hash map dengan menambahkan beberapa pasangan nama-nomor telepon, mencari nama, menghapus nama, dan kemudian menelusuri tabel untuk mencetak semua pasangan nama-nomor telepon yang tersisa.

### C. Unguided/Tugas

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).

```
#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 searchByNIM(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 searchByNilai(int nilaiMin, int nilaiMax)
{
    bool found = false;
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        for (auto pair : table[i])
        {
            if (pair != nullptr && pair->nilai >= nilaiMin && pair->nilai <=
nilaiMax)
            {
                cout << pair->nim << " memiliki nilai " << pair->nilai << endl;
            }
        }
    }
}

```

```

        found = true;
    }
}
}
if (!found)
{
    cout << "Tidak ada mahasiswa yang memiliki nilai antara " << nilaiMin
<< " and " << nilaiMax << 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.searchByNIM(NIM) << endl;
        break;
    case 4:
        int x, z;
        cout << "Masukkan rentang nilai min : ";
        cin >> x;
        cout << "Masukkan rentang nilai max : ";
        cin >> z;
        data.searchByNilai(x, z);
        break;
    case 5:
        data.print();
        break;
    case 6:
        return 0;
    default:
        cout << "Menu tidak tersedia!" << endl;
        break;
    }
}
return 0;
}

```

## Screenshots Output

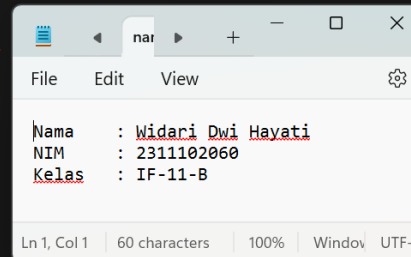
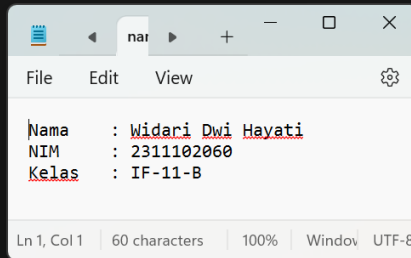
```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2311102060
Masukkan nilai : 87

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 : 2311102061
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 : 1
Masukkan NIM : 2311102062
Masukkan nilai : 77

Menu
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: [2311102060, 87]
2: [2311102061, 85]
3: [2311102062, 77]
4:
5:
6:
7:
8:
9:
10:

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 : 2311102061
Data berhasil dihapus
```



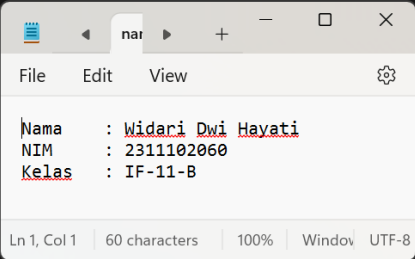
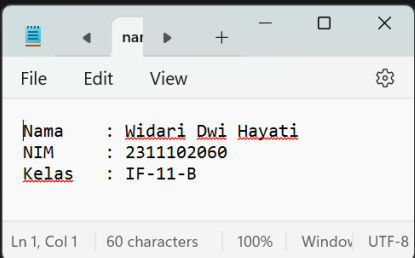


```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 5
0:
1: [2311102060, 87]
2:
3: [2311102062, 77]
4:
5:
6:
7:
8:
9:
10:

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 : 2311102062
NIM 2311102062 memiliki nilai 77

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 min : 80
Masukkan rentang nilai max : 90
2311102060 memiliki nilai 87

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkkan pilihan : 6
PS C:\Users\ADVAN>
```



### Deskripsi:

Kode program tersebut mengimplementasikan tabel hash dengan ukuran tetap 11. Program ini digunakan untuk menyimpan pasangan NIM dan nilai mahasiswa. Kelas HashNode mewakili elemen dalam hash map, dengan memuat NIM dan nilai mahasiswa. Kelas HashMap berisi array dari pointer ke objek HashNode, yang menyusun basis dari tabel hash.

Fungsi hashFunc menghitung indeks hash untuk NIM yang diberikan dengan cara menjumlahkan nilai ASCII dari setiap karakter. Fungsi insert menambahkan pasangan NIM-nilai mahasiswa baru ke hash map dengan menghitung indeks hash menggunakan fungsi hashFunc, dan kemudian menelusuri rantai node pada indeks tersebut untuk melihat apakah NIM sudah ada. Jika demikian, nilai diperbarui; jika tidak, node baru dibuat dan ditambahkan ke depan rantai. Fungsi remove menghapus pasangan NIM-nilai mahasiswa dari hash map dengan menghitung indeks hash dan kemudian menelusuri rantai node pada indeks tersebut hingga menemukan node dengan NIM yang sesuai. Jika node seperti itu ditemukan, maka dihapus dari rantai dan dihapus. Fungsi searchByNIM mencari NIM dalam hash map dan mengembalikan nilai mahasiswa yang sesuai jika ditemukan. Fungsi ini melakukan ini dengan menghitung indeks hash dan kemudian menelusuri rantai node pada indeks tersebut hingga menemukan node dengan NIM yang sesuai. Jika tidak ada node seperti itu, maka mengembalikan -1. Fungsi searchByNilai mencari mahasiswa dalam hash map dengan nilai yang berada dalam rentang tertentu. Fungsi ini melakukan ini dengan menelusuri semua rantai dalam tabel dan memeriksa nilai dari setiap node. Fungsi print mencetak semua pasangan NIM-nilai mahasiswa dalam hash map dengan menelusuri semua rantai dalam tabel.

Fungsi main menunjukkan penggunaan hash map dengan menambahkan beberapa pasangan NIM-nilai mahasiswa, mencari NIM, menghapus NIM, dan kemudian menelusuri tabel untuk mencetak semua pasangan NIM-nilai mahasiswa yang tersisa.

#### D. Kesimpulan

Hash Table adalah struktur data yang digunakan untuk mengorganisir data menjadi pasangan kunci-nilai. Ini terdiri dari dua komponen utama, yaitu array dan fungsi hash. Fungsi hash digunakan untuk mengubah kunci menjadi nilai indeks array, yang kemudian digunakan untuk menyimpan data. Dengan menggunakan tabel hash, operasi pencarian, penyisipan, dan penghapusan data dapat dilakukan dalam waktu konstan. Namun, terdapat masalah ketika dua atau lebih data mempunyai nilai hash yang sama, yang disebut collision. Ada dua teknik umum untuk menyelesaikan masalah ini, yaitu open hashing dan closed hashing. Open hashing menggunakan teknik chaining, yaitu menyimpan semua item data dengan nilai indeks yang sama ke dalam sebuah linked list. Sedangkan closed hashing menggunakan teknik seperti linear probing, quadratic probing, dan double hashing.

#### E. Referensi (APA)

Stroustrup, B. (2021). *A Tour of C++* (2nd ed.). Addison-Wesley Professional.

Lippman, S. L., Lajoie, J., & Moo, B. (2020). *C++ Primer* (6th ed.). Addison-Wesley Professional.

Malik, D. S. (2014). *C++ Programming: From Problem Analysis to Program Design* (7th ed.). Cengage Learning.