

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



Disusun Oleh :

NAMA : D'sharlendita Febianda Aurelia
NIM : 2311102069

Dosen :

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

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

A. Dasar Teori

1. Pengertian Hash Table

Hash table atau tabel hash adalah struktur data yang digunakan untuk menyimpan data dengan cara memetakan kunci (key) ke nilai (value) menggunakan fungsi hash. Fungsi hash menghasilkan indeks array, di mana nilai disimpan. Hash table menawarkan operasi penyisipan, pencarian, dan penghapusan data yang sangat cepat dengan waktu rata-rata $O(1)$, asalkan fungsi hash didesain dengan baik dan tidak terjadi tabrakan (collision).

2. Fungsi Hash

Fungsi hash adalah fungsi yang memetakan kunci (key) ke nilai integer yang disebut hash value. Hash value digunakan sebagai indeks array untuk menyimpan nilai yang terkait dengan kunci. Fungsi hash harus didesain dengan baik untuk meminimalkan tabrakan, yaitu situasi di mana dua kunci berbeda menghasilkan hash value yang sama.

3. Tabrakan dan Teknik Resolusi

Tabrakan adalah situasi di mana dua kunci berbeda menghasilkan hash value yang sama. Tabrakan dapat menyebabkan inefisiensi dan memperlambat operasi hash table. Ada beberapa teknik resolusi tabrakan yang umum digunakan, antara lain:

- **Chaining:** Menyimpan beberapa nilai dalam satu slot array, di mana setiap nilai dihubungkan ke nilai lain dalam daftar tertaut.
- **Open addressing:** Mencoba slot array lain ketika slot yang dihitung oleh hash function sudah terisi. Teknik open addressing yang umum digunakan adalah linear probing, quadratic probing, dan double hashing.

4. Keuntungan Hash Table

Hash table memiliki beberapa keuntungan dibandingkan struktur data lain, seperti array dan daftar tertaut:

- **Operasi yang sangat cepat:** Operasi penyisipan, pencarian, dan penghapusan data rata-rata membutuhkan waktu $O(1)$, asalkan fungsi hash didesain dengan baik dan tidak terjadi tabrakan.
- **Efisiensi ruang:** Hash table hanya menggunakan ruang yang diperlukan untuk menyimpan data, tidak seperti array yang perlu dialokasikan ruang untuk semua elemen meskipun tidak terisi.
- **Fleksibel:** Hash table dapat digunakan untuk menyimpan berbagai jenis data, seperti string, bilangan, dan objek.

5. Kekurangan Hash Table

Hash table juga memiliki beberapa kekurangan, antara lain:

- **Ketergantungan pada fungsi hash:** Kinerja hash table sangat bergantung pada kualitas fungsi hash. Fungsi hash yang buruk dapat menyebabkan banyak tabrakan, yang memperlambat operasi dan meningkatkan inefisiensi.
- **Memerlukan ruang overhead:** Hash table memerlukan ruang overhead untuk menyimpan struktur data internal, seperti tabel hash dan daftar tertaut untuk chaining.
- **Sensitif terhadap data duplikat:** Hash table tidak dapat menyimpan data duplikat dengan kunci yang sama secara efisien.

B. Guided

Guided 1

Program Hash Table

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;
    }
}

```

```

        }
        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;

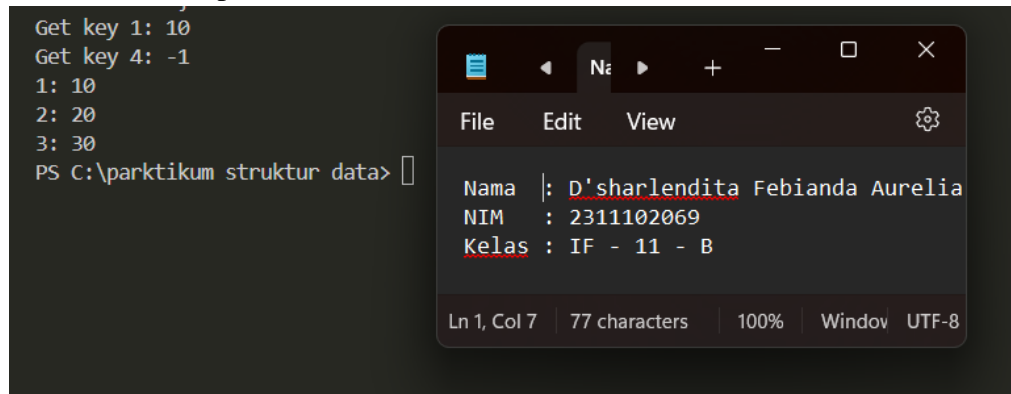
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

Screenshots Output :



The screenshot shows a terminal window on the left with the following output:

```
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\parktikum struktur data>
```

On the right, a text editor window displays a record:

```
Nama   |: D'sharlendita Febianda Aurelia
NIM    |: 2311102069
Kelas |: IF - 11 - B
```

The text editor's status bar at the bottom indicates: Ln 1, Col 7 | 77 characters | 100% | Window | UTF-8.

Deskripsi:

Program ini mengimplementasikan Hash Table sederhana untuk menyimpan pasangan key-value.

Guided 2

Program Hash Table

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;
}

```

Screenshots Output :

The screenshot shows the output of the C++ program in a terminal window. The output is as follows:

```

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:\parktikum struktur data>

```

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

```

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B

```

The Notepad window also shows a status bar at the bottom: 'Ln 1, Col 7 | 77 characters | 100% | Window UTF-8'.

Deskripsi:

Program ini menunjukkan implementasi sederhana buku telepon menggunakan hash table. Ini memungkinkan pengguna untuk menambahkan, menghapus, dan mencari nomor telepon dengan efisien.

C. Unguided

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 :

- Setiap mahasiswa memiliki NIM dan nilai.
- Program memiliki tampilan pilihan menu berisi poin 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 <iomanip>
using namespace std;

const int MAX_SIZE = 10;

// struktur data untuk setiap node
struct mahasiswa
{
    string nama;
    long long NIM;
    int nilai;
    mahasiswa *next; // pointer untuk menunjukkan node
    selanjutnya dalam collision handling

    mahasiswa(string nama, long long NIM, int nilai) :
    nama(nama), NIM(NIM), nilai(nilai), next(nullptr) {}
};

// class hash table
class HashTable
{
private:
    mahasiswa **table; // array pointer untuk menunjukkan ke
    elemen-elemen tabel hash

    // fungsi hash sederhana
```

```

    int hash_func(Long Long key)
    {
        return key % MAX_SIZE; // menggunakan modulus untuk
mendapatkan indeks
    }

public:
    // constructor
    HashTable()
    {
        table = new mahasiswa *[MAX_SIZE](); // inisialisasi
array pointer dengan nullptr
    }

    // destructor
    ~HashTable()
    {
        // menghapus semua node dan array pointer
        for (int i = 0; i < MAX_SIZE; ++i)
        {
            mahasiswa *current = table[i];
            while (current != nullptr)
            {
                mahasiswa *temp = current;
                current = current->next;
                delete temp;
            }
        }
        delete[] table;
    }

    // insertion
    void insert(string nama, Long Long NIM, int nilai)
    {
        int index = hash_func(NIM);
        // mendapatkan indeks berdasarkan NIM
        mahasiswa *new_mahasiswa = new mahasiswa(nama, NIM,
nilai); // membuat node baru
        new_mahasiswa->next = table[index];
        // menambahkan node baru ke depan daftar di indeks yang tepat
        table[index] = new_mahasiswa;
    }

```

```

// deletion
void remove(Long Long NIM)
{
    int index = hash_func(NIM); // mendapatkan indeks
    berdasarkan NIM
    mahasiswa *current = table[index]; // mengambil node
    pertama di indeks yang tepat
    mahasiswa *prev = nullptr; // pointer untuk node
    sebelumnya
    while (current != nullptr)
    {
        if (current->NIM == NIM)
        {
            if (prev == nullptr)
            {
                table[index] = current->next; // jika node
                yang dihapus adalah node pertama di indeks, atur node berikutnya
                sebagai node pertama
            }
            else
            {
                prev->next = current->next; // jika bukan,
                hubungkan node sebelumnya dengan node setelahnya
            }
            delete current; // hapus node yang ditemukan
            cout << "Mahasiswa dengan NIM " << NIM << " telah
            dihapus." << endl;
            return;
        }
        prev = current;
        current = current->next;
    }
    cout << "Mahasiswa dengan NIM " << NIM << " tidak
    ditemukan." << endl;
}

// searching by NIM
mahasiswa *cari_NIM(Long Long NIM)
{
    Long Long index = hash_func(NIM); // mendapatkan indeks
    berdasarkan NIM
    mahasiswa *current = table[index]; // mengambil node
    pertama di indeks yang tepat

```

```

        while (current != nullptr)
        {
            if (current->NIM == NIM)
            {
                return current; // kembalikan node jika NIM cocok
            }
            current = current->next;
        }
        return nullptr; // kembalikan nullptr jika tidak
ditemukan
    }

    // searching by rentang nilai
    void cari_nilai(int awal, int akhir)
    {
        // mencetak header tabel
        cout << "-----" << endl;
        cout << "-----" << endl;
        cout << "|          Nama Mahasiswa          |          NIM" << endl;
        cout << "|      Nilai      |" << endl;
        cout << "-----" << endl;
        cout << "-----" << endl;
        // traverse semua bucket pada tabel hash
        for (int i = 0; i < MAX_SIZE; ++i) {
            mahasiswa* current = table[i]; // mengambil node
pertama di indeks yang tepat
            // traverse semua node dalam bucket
            while (current != nullptr) {
                // jika nilai berada dalam rentang yang
ditentukan, cetak informasi mahasiswa
                if (current->nilai >= awal && current->nilai <=
akhir) {
                    string namaSingkat = current->nama.substr(0,
30);
                    cout << "| " << setw(30) << left <<
namaSingkat << " | ";
                    cout << setw(20) << current->NIM << " | ";
                    cout << setw(10) << fixed << setprecision(2)
<< current->nilai << " |" << endl;
                }
                current = current->next;
            }
        }
    }
}

```

```

        cout << "-----"
-----" << endl;
    }

    // traversal
    void traverse() {
        cout << "-----"
-----" << endl;
        cout << "|          Nama Mahasiswa          |          NIM
|      Nilai      |" << endl;
        cout << "-----"
-----" << endl;

        for (int i = 0; i < MAX_SIZE; ++i) {
            mahasiswa* current = table[i];

            while (current != nullptr) {
                string namaSingkat = current->nama.substr(0, 30); //
Memotong 30 karakter pertama
                cout << "| " << setw(30) << left << namaSingkat << " |
";

                cout << setw(20) << current->NIM << " | ";
                cout << setw(10) << current->nilai << " |" << endl;

                current = current->next;
            }
        }

        cout << "-----"
-----" << endl;
    }
};

int main()
{
    HashTable ht;

    int pilih, nilai, awal, akhir;
    string nama;
    Long Long NIM;

    do
    {

```

```

// menu utama
cout << endl;
cout << " M E N U : " << endl << endl;
cout << "-----"
-----" << endl << endl;
cout << "1. Tambah Data Mahasiswa" << endl;
cout << "2. Hapus Data Mahasiswa" << endl;
cout << "3. Cari berdasarkan NIM" << endl;
cout << "4. Cari berdasarkan Rentang Nilai" << endl;
cout << "5. Tampilkan Semua Data" << endl;
cout << "6. Keluar" << endl << endl;
cout << "-----"
-----" << endl << endl;
cout << "Pilih menu : ";
cin >> pilih;
cout << endl;

switch (pilih) {
case 1:
    // tambah data mahasiswa
    cout << "Masukan nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukan NIM : ";
    cin >> NIM;
    cout << "Masukan nilai : ";
    cin >> nilai;
    ht.insert(nama, NIM, nilai);
    cout<<"Data berhasil ditambahkan"<<endl;
    cout << "-----"
-----" << endl << endl;

    break;
case 2:
    // hapus data mahasiswa berdasarkan NIM
    cout << "Masukan NIM yang ingin dihapus : ";
    cin >> NIM;
    ht.remove(NIM);
    cout << "-----"
-----" << endl << endl;

    break;
case 3:
    // cari mahasiswa berdasarkan NIM
    cout << "Masukan NIM yang ingin dicari : ";

```

```

        cin >> NIM;
        {
            mahasiswa *mahasiswa_ptr = ht.cari_NIM(NIM);
            if (mahasiswa_ptr)
            {
                cout << "Ditemukan mahasiswa dengan NIM " <<
NIM << " bernama " << mahasiswa_ptr->nama << " dan memiliki nilai
" << mahasiswa_ptr->nilai << endl;
            }
            else
            {
                cout << "mahasiswa dengan NIM " << NIM << "
tidak ditemukan" << endl;
            }
        }
        cout << "-----"
-----" << endl << endl;

        break;
    case 4:
        // cari mahasiswa berdasarkan rentang nilai
        cout << "masukan nilai awal : ";
        cin >> awal;
        cout << "masukan nilai akhir : ";
        cin >> akhir;
        ht.cari_nilai(awal, akhir);
        break;
    case 5:
        // tampilkan semua data mahasiswa
        ht.traverse();
        break;
    case 6:
        cout << "Terimakasih" << endl; // keluar dari program
        cout << "-----"
-----" << endl << endl;

        break;
    default:
        cout << "Pilihan anda tidak tersedia!" << endl; //
jika pilihan tidak valid
        cout << "-----"
-----" << endl << endl;
    }
} while (pilih != 6); // ulangi sampai pengguna memilih untuk
keluar

```



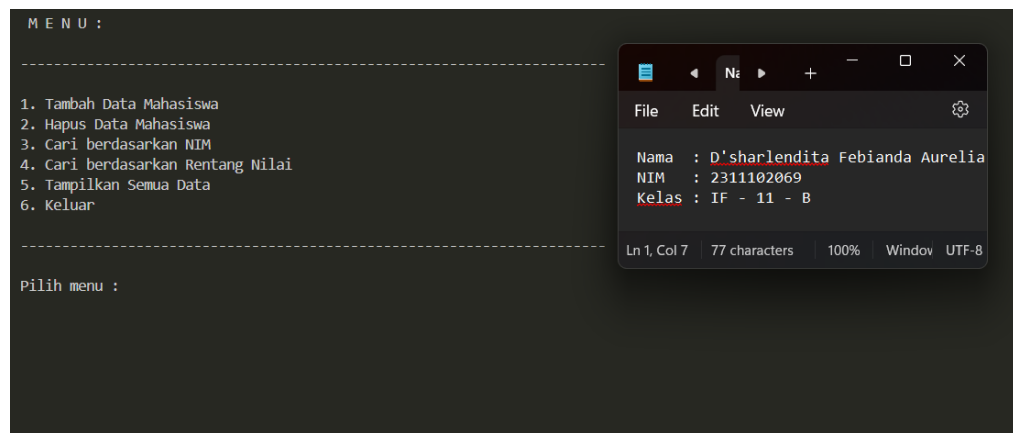
```

    return 0;
}

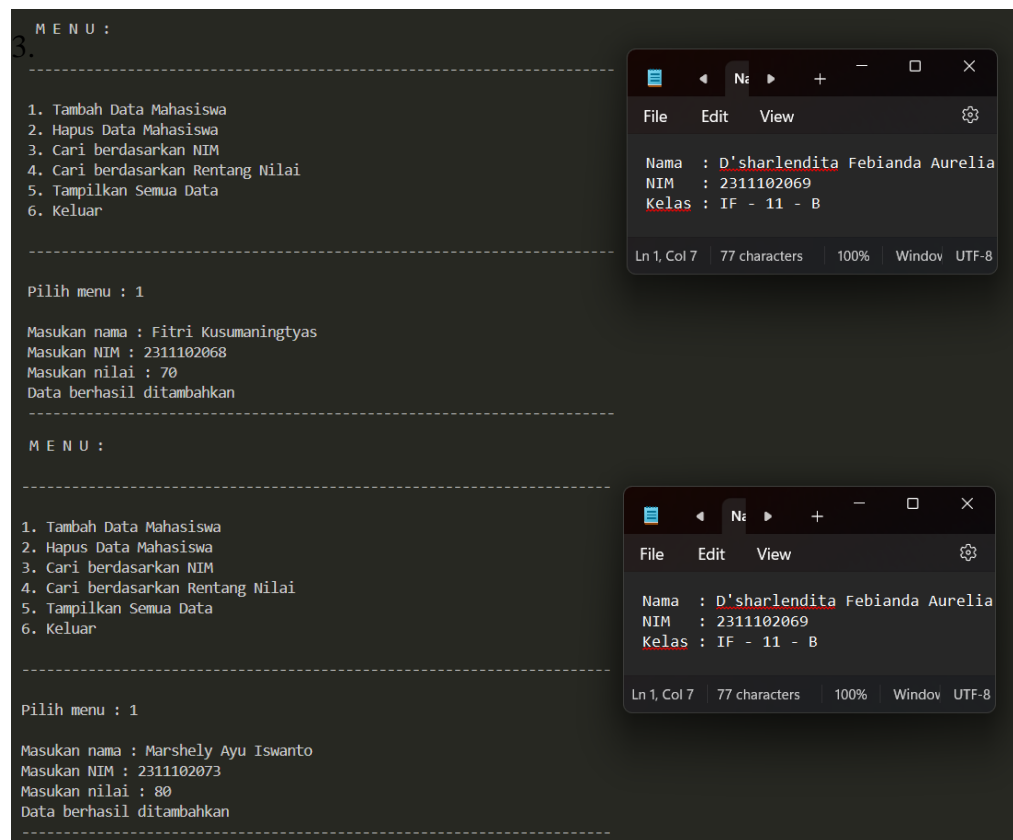
```

Screenshots Output

1. Buatlah menu untuk menambahkan, menghapus, mencari data mahasiswa berdasarkan NIM, mencari data mahasiswa berdasarkan rentang nilai, dan melihat semua data mahasiswa yang tersimpan pada Hash Table. Tampilan menu:



2. Menambahkan data mahasiswa



M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

Masukan nama : Nia Novela Ariandini
Masukan NIM : 2311102057
Masukan nilai : 80
Data berhasil ditambahkan

M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

Masukan nama : Trie Nabilla Farhah
Masukan NIM : 2311102071
Masukan nilai : 90
Data berhasil ditambahkan

M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

Masukan nama : Widari Dwi Hayati
Masukan NIM : 2311102060
Masukan nilai : 70
Data berhasil ditambahkan

M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

Masukan nama : Anisa Yasaroh
Masukan NIM : 2311102063
Masukan nilai : 90
Data berhasil ditambahkan

```
File Edit View
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B
Ln 1, Col 7 77 characters 100% Window UTF-8
```

```
File Edit View
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B
Ln 1, Col 7 77 characters 100% Window UTF-8
```

```
File Edit View
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B
Ln 1, Col 7 77 characters 100% Window UTF-8
```

```
File Edit View
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B
Ln 1, Col 7 77 characters 100% Window UTF-8
```

M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

Masukan nama : Anisah Syifa Mustika Riyanto
Masukan NIM : 2311102080
Masukan nilai : 70
Data berhasil ditambahkan

M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

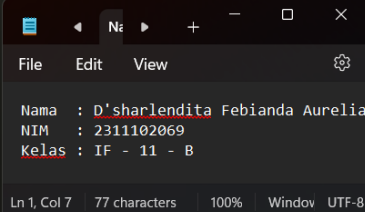
Masukan nama : D'sharlendita Febianda Aurelia
Masukan NIM : 2311102069
Masukan nilai : 90
Data berhasil ditambahkan

M E N U :

1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar

Pilih menu : 1

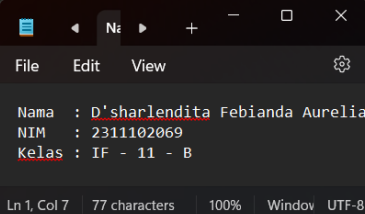
Masukan nama : Yasvin Syahgana
Masukan NIM : 2311102065
Masukan nilai : 70
Data berhasil ditambahkan



A screenshot of a terminal window with a dark theme. The window title is 'Nz'. The menu bar shows 'File', 'Edit', 'View', and a settings icon. The content displays a student record: 'Nama : D'sharlendita Febianda Aurelia', 'NIM : 2311102069', and 'Kelas : IF - 11 - B'. The status bar at the bottom shows 'Ln 1, Col 7', '77 characters', '100%', 'Window', and 'UTF-8'.

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B

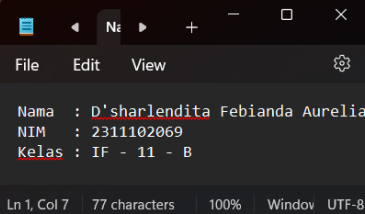
Ln 1, Col 7 77 characters 100% Window UTF-8



A screenshot of a terminal window with a dark theme. The window title is 'Nz'. The menu bar shows 'File', 'Edit', 'View', and a settings icon. The content displays a student record: 'Nama : D'sharlendita Febianda Aurelia', 'NIM : 2311102069', and 'Kelas : IF - 11 - B'. The status bar at the bottom shows 'Ln 1, Col 7', '77 characters', '100%', 'Window', and 'UTF-8'.

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B

Ln 1, Col 7 77 characters 100% Window UTF-8



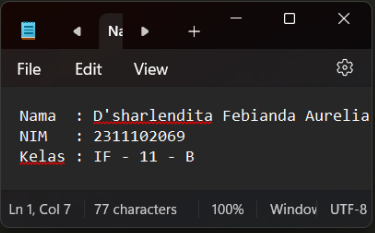
A screenshot of a terminal window with a dark theme. The window title is 'Nz'. The menu bar shows 'File', 'Edit', 'View', and a settings icon. The content displays a student record: 'Nama : D'sharlendita Febianda Aurelia', 'NIM : 2311102069', and 'Kelas : IF - 11 - B'. The status bar at the bottom shows 'Ln 1, Col 7', '77 characters', '100%', 'Window', and 'UTF-8'.

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B

Ln 1, Col 7 77 characters 100% Window UTF-8

4. Menampilkan data mahasiswa yang tersimpan pada hash table

```
M E N U :  
-----  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari berdasarkan NIM  
4. Cari berdasarkan Rentang Nilai  
5. Tampilkan Semua Data  
6. Keluar  
-----  
Pilih menu : 5  
-----  
| Nama Mahasiswa | NIM | Nilai |  
-----  
| Anisah Syifa Mustika Riyanto | 2311102080 | 70 |  
| Widari Dwi Hayati | 2311102060 | 70 |  
| Trie Nabilla Farhah | 2311102071 | 90 |  
| Anisa Yasaroh | 2311102063 | 90 |  
| Marshely Ayu Iswanto | 2311102073 | 80 |  
| Yasvin Syahgana | 2311102065 | 70 |  
| Nia Novela Ariandini | 2311102057 | 80 |  
| Fitri Kusumaningtyas | 2311102068 | 70 |  
| D'sharlendita Febianda Aurelia | 2311102069 | 90 |  
-----
```

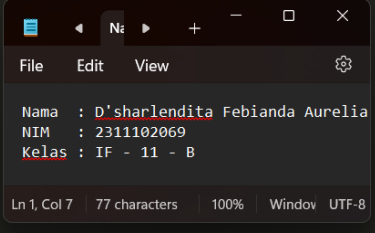


Notepad++ window content:

```
Nama : D'sharlendita Febianda Aurelia  
NIM : 2311102069  
Kelas : IF - 11 - B  
Ln 1, Col 7 77 characters 100% Window UTF-8
```

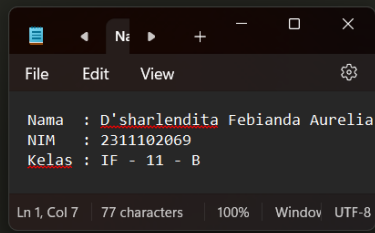
5. Menghapus salah satu data mahasiswa pada hash table menggunakan NIM

```
M E N U :  
-----  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari berdasarkan NIM  
4. Cari berdasarkan Rentang Nilai  
5. Tampilkan Semua Data  
6. Keluar  
-----  
Pilih menu : 2  
Masukan NIM yang ingin dihapus : 2311102065  
Mahasiswa dengan NIM 2311102065 telah dihapus.  
-----  
M E N U :  
-----  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari berdasarkan NIM  
4. Cari berdasarkan Rentang Nilai  
5. Tampilkan Semua Data  
6. Keluar  
-----  
Pilih menu : 5  
-----  
| Nama Mahasiswa | NIM | Nilai |  
-----  
| Anisah Syifa Mustika Riyanto | 2311102080 | 70 |  
| Widari Dwi Hayati | 2311102060 | 70 |  
| Trie Nabilla Farhah | 2311102071 | 90 |  
| Anisa Yasaroh | 2311102063 | 90 |  
| Marshely Ayu Iswanto | 2311102073 | 80 |  
| Nia Novela Ariandini | 2311102057 | 80 |  
| Fitri Kusumaningtyas | 2311102068 | 70 |  
| D'sharlendita Febianda Aurelia | 2311102069 | 90 |  
-----
```



Notepad++ window content:

```
Nama : D'sharlendita Febianda Aurelia  
NIM : 2311102069  
Kelas : IF - 11 - B  
Ln 1, Col 7 77 characters 100% Window UTF-8
```



Notepad++ window content:

```
Nama : D'sharlendita Febianda Aurelia  
NIM : 2311102069  
Kelas : IF - 11 - B  
Ln 1, Col 7 77 characters 100% Window UTF-8
```

6. Mencari data mahasiswa berdasarkan NIM

```
M E N U :  
-----  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari berdasarkan NIM  
4. Cari berdasarkan Rentang Nilai  
5. Tampilkan Semua Data  
6. Keluar  
-----  
Pilih menu : 3  
  
Masukan NIM yang ingin dicari : 2311102069  
Ditemukan mahasiswa dengan NIM 2311102069 bernama D'sharlendita Febianda Aurelia dan memiliki nilai 90  
-----
```



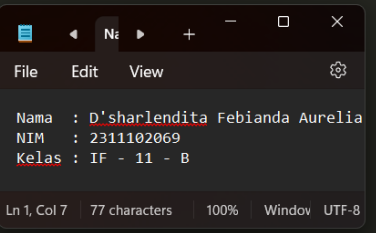
The Notepad window displays the following text:

```
Nama : D'sharlendita Febianda Aurelia  
NIM : 2311102069  
Kelas : IF - 11 - B
```

Ln 1, Col 7 77 characters 100% Window UTF-8

7. Mencari data mahasiswa berdasarkan rentang nilai

```
M E N U :  
-----  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari berdasarkan NIM  
4. Cari berdasarkan Rentang Nilai  
5. Tampilkan Semua Data  
6. Keluar  
-----  
Pilih menu : 4  
  
masukan nilai awal : 80  
masukan nilai akhir : 90  
-----  
| Nama Mahasiswa | NIM | Nilai |  
-----  
| Trie Nabilla Farhah | 2311102071 | 90 |  
| Anisa Yasaroh | 2311102063 | 90 |  
| Marshely Ayu Iswanto | 2311102073 | 80 |  
| Nia Novela Ariandini | 2311102057 | 80 |  
| D'sharlendita Febianda Aurelia | 2311102069 | 90 |  
-----
```



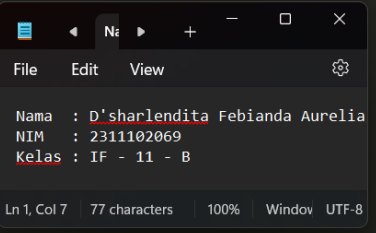
The Notepad window displays the following text:

```
Nama : D'sharlendita Febianda Aurelia  
NIM : 2311102069  
Kelas : IF - 11 - B
```

Ln 1, Col 7 77 characters 100% Window UTF-8

8. Keluar dari program

```
M E N U :  
-----  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari berdasarkan NIM  
4. Cari berdasarkan Rentang Nilai  
5. Tampilkan Semua Data  
6. Keluar  
-----  
Pilih menu : 6  
  
Terimakasih  
-----
```



The Notepad window displays the following text:

```
Nama : D'sharlendita Febianda Aurelia  
NIM : 2311102069  
Kelas : IF - 11 - B
```

Ln 1, Col 7 77 characters 100% Window UTF-8

Deskripsi:

Program ini menyediakan implementasi Hash Table yang sederhana dan mudah digunakan untuk mengelola data mahasiswa. Program ini

menawarkan fungsi-fungsi dasar untuk menambahkan, menghapus, mencari, dan menampilkan data mahasiswa.

D. Kesimpulan

Hash Table merupakan pilihan yang tepat untuk aplikasi yang membutuhkan operasi pencarian, penyisipan, dan penghapusan data yang cepat dan efisien. Namun, perlu diperhatikan pemilihan fungsi Hash yang tepat untuk meminimalkan tabrakan dan menjaga performa Hash Table secara optimal.

E. Referensi

- Asisten Praktikum. 08 Mei 2024. "Modul 5 Hash Table". Diakses pada 10 Mei 2024, dari Learning Management System. 2024
- Aziz, N., & Arief, M. (2020). Implementasi Hash Table Menggunakan C++. Diakses pada 10 Mei 2024, dari <https://www.belajarstatistik.com/blog/2022/03/08/implementasi-hash-dalam-bahasa-c/>
- Groetz, B., & Colburn, M. (2021). Data Structures and Algorithms in C++ (5th Edition). Diakses pada 10 Mei 2024, dari <https://www.pearson.com/en-us/subject-catalog/p/introduction-to-c-programming-and-data-structures/P200000003313/9780137454181>
- GeeksforGeeks. (08 May, 2024). Hash Table Data Structure. Diakses pada 10 Mei 2024, dari <https://www.geeksforgeeks.org/hash-table-data-structure/>.