

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 5
“HASH TABLE”**



**DISUSUN OLEH :
RIZKY PERLINTA SEMBIRNG
2311102061**

**DOSEN
WAHYU ANDI SAPUTRA, S.PD., M.PD.**

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

A. DASAR TEORI

1. Algoritma Hash Table di C++

Di bawah ini adalah prosedur langkah demi langkah yang diikuti untuk mengimplementasikan hash table di C++ menggunakan fungsi hash:

- Menginisialisasi ukuran tabel ke beberapa nilai integer.
- Membuat struktur tabel hash `hashTableEntry` untuk deklarasi pasangan kunci dan nilai.
- Membuat konstruktor `hashMapTable`.
- Buat `hashFunction()` dan temukan nilai hash yang akan menjadi indeks untuk menyimpan data aktual dalam tabel hash menggunakan rumus:

```
hash_value = hashFunction(key);  
index = hash_value % (table_size)
```

- Fungsi masing-masing seperti `Insert()`, `searchKey()`, dan `Hapus()` digunakan untuk menyisipkan elemen pada kunci, mencari elemen pada kunci, dan menghapus elemen pada kunci masing-masing.
- Destructor dipanggil untuk menghancurkan semua objek `hashMapTable`.

2. Bagaimana Hash Table Bekerja di C++?

Seperti dibahas di atas, tabel hash menyimpan pointer ke data/nilai aktual. Ia menggunakan kunci untuk menemukan indeks di mana data/nilai perlu disimpan. Mari kita pahami ini dengan bantuan diagram di bawah ini:

Anggaplah ukuran tabel hash menjadi 10

Kunci	Indeks (menggunakan fungsi hash)	Data
12	$12 \% 10 = 2$	23
10	$10 \% 10 = 0$	34

6	$6 \% 10 = 6$	54
23	$23 \% 10 = 3$	76
54	$54 \% 10 = 4$	75
82	$81 \% 10 = 1$	87

Posisi elemen dalam hash table adalah:

0 1 2 3 4 5 6 7 8 9.

34	87	23	76	75		54			
----	----	----	----	----	--	----	--	--	--

Seperti yang bisa kita lihat di atas, ada kemungkinan besar terjadinya tabrakan karena mungkin ada 2 atau lebih kunci yang menghitung kode hash yang sama sehingga menghasilkan indeks elemen yang sama dalam hash table. Tabrakan tidak dapat dihindari jika terjadi hashing meskipun kita memiliki ukuran tabel yang besar. Kita dapat mencegah tabrakan dengan memilih fungsi hash yang baik dan metode implementasinya.

Meskipun ada banyak teknik implementasi yang digunakan seperti Linear probing, open hashing, dll. Kita akan memahami dan menerapkan teknik dasar Open hashing yang juga disebut Separate Chaining. Dalam teknik ini, daftar tertaut digunakan untuk merangkai nilai. Setiap entri dalam hash table adalah daftar tertaut. Jadi, ketika entri baru perlu dibuat, indeks dihitung menggunakan kunci dan ukuran tabel. Setelah dihitung, itu dimasukkan ke dalam daftar yang sesuai dengan indeks tersebut. Ketika ada 2 nilai atau lebih yang memiliki nilai hash/indeks yang sama, kedua entri disisipkan sesuai dengan indeks yang dihubungkan satu sama lain. Misalnya,

2 → 12 → 22 → 32

Di mana, 2 adalah indeks tabel hash yang diambil menggunakan fungsi hash.

12, 22, 32 adalah nilai data yang akan disisipkan saling terkait satu sama lain.

B. Guided

Guided 1.

Source Code:

```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key){
    return key % MAX_SIZE;
}
// Struktur data untuk setiap node
struct Node
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key), value(value),
                                next(nullptr) {}
};
// Class hash table
class HashTable
{
private:
    Node **table;
public:
    HashTable()
    {
        table = new Node *[MAX_SIZE]();
    }
    ~HashTable()
    {
        for (int i = 0; i < MAX_SIZE; i++)
        {
            Node *current = table[i];
            while (current != nullptr)
            {
                Node *temp = current;
                current = current->next;
                delete temp;
            }
        }
        delete[] table;
    }
    // Insertion
    void insert(int key, int value)
    {

```

```

    int index = hash_func(key);
    Node *current = table[index];
    while (current != nullptr)
    {
        if (current->key == key)
        {
            current->value = value;
            return;
        }
        current = current->next;
    }
    Node *node = new Node(key, value);
    node->next = table[index];
    table[index] = node;
}

// Searching
int get(int key)
{
    int index = hash_func(key);
    Node *current = table[index];
    while (current != nullptr)
    {
        if (current->key == key)
        {
            return current->value;
        }
        current = current->next;
    }
    return -1;
}

// Deletion
void remove(int key)
{
    int index = hash_func(key);
    Node *current = table[index];
    Node *prev = nullptr;
    while (current != nullptr)
    {
        if (current->key == key)
        {
            if (prev == nullptr)
            {
                table[index] = current->next;
            }
            else
            {
                prev->next = current->next;
            }
        }
        prev = current;
        current = 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;
    cout << "Get key 5: " << ht.get(5) << endl;
    cout << "Get key 3: " << ht.get(3) << endl;

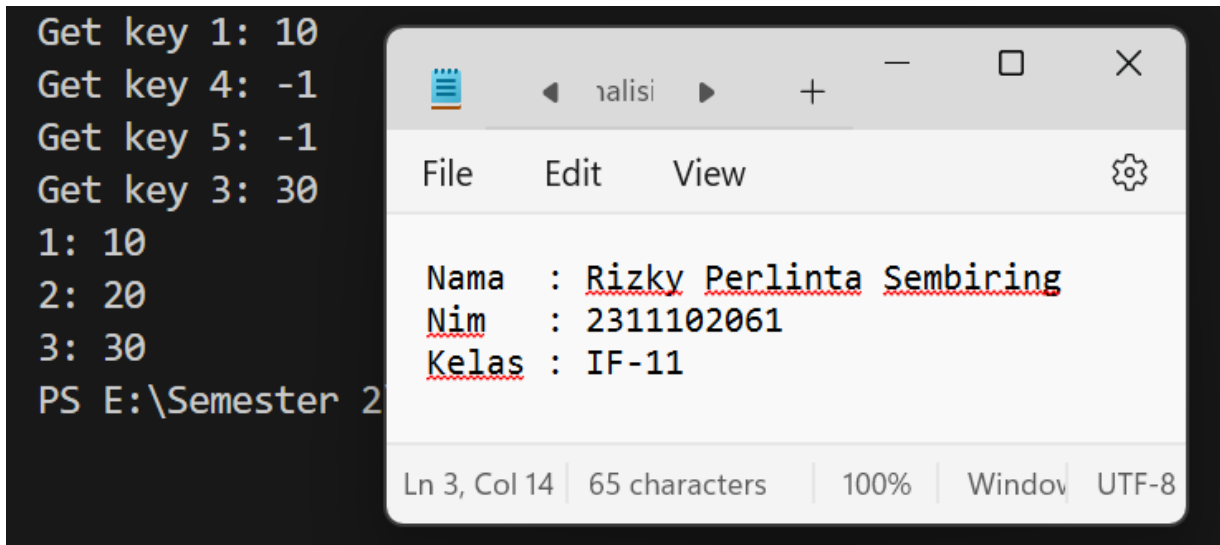
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

Screenshot Output:



The screenshot shows a terminal window on the left and a Notepad++ window on the right. The terminal window displays the following output:

```
Get key 1: 10
Get key 4: -1
Get key 5: -1
Get key 3: 30
1: 10
2: 20
3: 30
PS E:\Semester 2
```

The Notepad++ window shows the following text:

```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11
```

The Notepad++ window also shows the status bar: Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8.

Deskripsi Program: Fungsi Hash: Fungsi sederhana digunakan untuk mengonversi kunci menjadi indeks array dengan menggunakan operasi modulus. Struktur Node: Setiap node menyimpan pasangan kunci-nilai dan referensi ke node berikutnya dalam daftar berantai untuk menangani tabrakan. Kelas HashTable: Kelas utama yang mengatur operasi-operasi pada tabel hash. Insertion: Data disisipkan dengan menambahkan node baru, menangani tabrakan dengan menyisipkan node baru di depan daftar berantai. Searching: Dilakukan dengan mencari kunci yang cocok dalam daftar berantai pada indeks yang dihasilkan. Deletion: Data dihapus dengan menghapus node yang sesuai dengan kunci, menangani tabrakan dengan mengubah tautan node. Traversal: Melalui semua pasangan kunci-nilai dalam tabel hash untuk mencetaknya. Fungsi Main: Digunakan untuk menguji operasi-operasi yang dilakukan pada tabel hash, seperti penambahan, pencarian, dan penghapusan.

Guided 2.

Source Code:

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
string name;
string phone_number;
class HashNode
{
public:
    string name;
```



```

    string phone_number;
    HashNode(string name, string phone_number)
    {
        this->name = name;
        this->phone_number = phone_number;
    }
};
class HashMap
{
private:
    vector<HashNode *> table[TABLE_SIZE];

public:
    int hashFunc(string key)
    {
        int hash_val = 0;
        for (char c : key)
        {
            hash_val += c;
        }
        return hash_val % TABLE_SIZE;
    }
    void insert(string name, string phone_number)
    {
        int hash_val = hashFunc(name);
        for (auto node : table[hash_val])
        {
            if (node->name == name)
            {
                node->phone_number = phone_number;
                return;
            }
        }
        table[hash_val].push_back(new HashNode(name, phone_number));
    }
    void remove(string name)
    {
        int hash_val = hashFunc(name);
        for (auto it = table[hash_val].begin(); it != table[hash_val].end();
it++)
        {
            if ((*it)->name == name)
            {
                table[hash_val].erase(it);
                return;
            }
        }
    }
    string searchByName(string name)

```

```

{
    int hash_val = hashFunc(name);
    for (auto node : table[hash_val])
    {
        if (node->name == name)
        {
            return node->phone_number;
        }
    }
    return "";
}

void print()
{
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        cout << i << ": ";
        for (auto pair : table[i])
        {
            if (pair != nullptr)
            {
                cout << "[" << pair->name << ", " << pair->phone_number
<< "]"<< endl;
            }
        }
        cout << endl;
    }
}

};

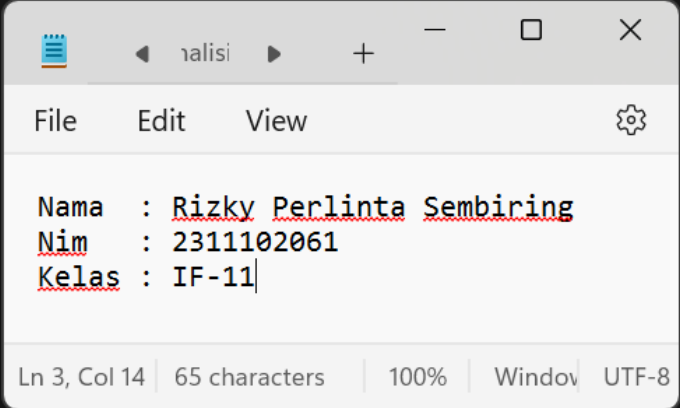
int main()
{
    HashMap employee_map;
    employee_map.insert("Mistah", "1234");
    employee_map.insert("Pastah", "5678");
    employee_map.insert("Ghana", "91011");
    cout << "Nomer Hp Mistah : " << employee_map.searchByName("Mistah") <<
endl;
    cout << "Phone Hp Pastah : " << employee_map.searchByName("Pastah") <<
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;
}

```

Screenshot 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 E:\Semester 2\Pratikum struktur data\guided> 
```

A screenshot of a Notepad window titled 'Notepad' with a menu bar (File, Edit, View) and a status bar (Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8). The text inside the window is: Nama : Rizky Perlinta Sembiring, Nim : 2311102061, and Kelas : IF-11. The text is underlined and has red squiggly lines underneath it, indicating it might be a placeholder or a specific format.

Deskripsi Program: Class HashNode: Merepresentasikan node dalam tabel hash. Setiap node menyimpan sebuah nama dan nomor telepon. Class HashMap: Menyediakan operasi untuk menyisipkan, menghapus, dan mencari data dalam tabel hash. insert(): Menambahkan pasangan nama dan nomor telepon ke dalam tabel hash. Jika terjadi kolisi, data disimpan dalam vektor yang sesuai. remove(): Menghapus data yang terkait dengan nama yang diberikan dari tabel hash. searchByName(): Mencari nomor telepon yang terkait dengan nama yang diberikan dalam tabel hash. print(): Mencetak isi dari setiap slot dalam tabel hash. Fungsi hashFunc(): Menghasilkan nilai hash berdasarkan jumlah karakter dalam nama. Ini menggunakan operasi modulus untuk menentukan indeks dalam tabel. Fungsi main(): Digunakan untuk menguji fungsionalitas tabel hash dengan menyisipkan beberapa entri, mencari nomor telepon dengan nama tertentu, menghapus entri, dan mencetak isi tabel hash.

C. UNGUIDED

A). 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 <unordered_map>
#include <vector>
using namespace std;

struct Mahasiswa
{
    string NIM;
    int nilai;
};

class HashTable
{
private:
    unordered_map<string, Mahasiswa> tabel;

public:
    void tambahData(Mahasiswa mahasiswa)
    {
        tabel[mahasiswa.NIM] = mahasiswa;
    }

    void hapusData(string NIM)
    {
        tabel.erase(NIM);
    }

    Mahasiswa cariDataBerdasarkanNIM(string NIM)
```

```

    {
        if (tabel.find(NIM) != tabel.end())
        {
            return tabel[NIM];
        }
        else
        {
            throw "NIM tidak ditemukan";
        }
    }

    vector<Mahasiswa> cariDataBerdasarkanRentangNilai(int nilaiMin, int
nilaiMax)
    {
        vector<Mahasiswa> hasil;
        for (auto it = tabel.begin(); it != tabel.end(); it++)
        {
            if (it->second.nilai >= nilaiMin && it->second.nilai <=
nilaiMax)
            {
                hasil.push_back(it->second);
            }
        }
        return hasil;
    }
};

int main()
{
    HashTable hashTable;
    int pilihan;
    do
    {
        cout << "Menu:" << endl;
        cout << "1. Tambah data baru" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Cari data berdasarkan NIM" << endl;
        cout << "4. Cari data berdasarkan rentang nilai (80-90)" <<
endl;

        cout << "5. Keluar" << endl;
        cout << "Masukkan pilihan: ";
        cin >> pilihan;

        if (pilihan == 1)
        {
            Mahasiswa mahasiswa;
            cout << "Masukkan NIM: ";
            cin >> mahasiswa.NIM;
            cout << "Masukkan nilai: ";

```

```

        cin >> mahasiswa.nilai;
        hashTable.tambahData(mahasiswa);
        cout << "Data berhasil ditambahkan" << endl;
        cout << endl;
    }
    else if (pilihan == 2)
    {
        string NIM;
        cout << "Masukkan NIM yang ingin dihapus: ";
        cin >> NIM;
        hashTable.hapusData(NIM);
        cout << "Data berhasil dihapus" << endl;
        cout << endl;
    }
    else if (pilihan == 3)
    {
        string NIM;
        cout << "Masukkan NIM yang ingin dicari: ";
        cin >> NIM;
        try
        {
            Mahasiswa mahasiswa =
hashTable.cariDataBerdasarkanNIM(NIM);
            cout << "NIM: " << mahasiswa.NIM << ", Nilai: " <<
mahasiswa.nilai << endl;
        }
        catch (const char *msg)
        {
            cerr << msg << endl;
        }
        cout << endl;
    }

    else if (pilihan == 4)
    {
        int nilaiMin, nilaiMax;
        cout << "Masukkan rentang nilai yang ingin dicari (misal: 80
90): ";
        cin >> nilaiMin >> nilaiMax;

        vector<Mahasiswa> hasil =
hashTable.cariDataBerdasarkanRentangNilai(nilaiMin, nilaiMax);
        for (Mahasiswa mahasiswa : hasil)
        {
            cout << "NIM: " << mahasiswa.NIM << ", Nilai: " <<
mahasiswa.nilai << endl;
        }
        cout << endl;
    }
}

```

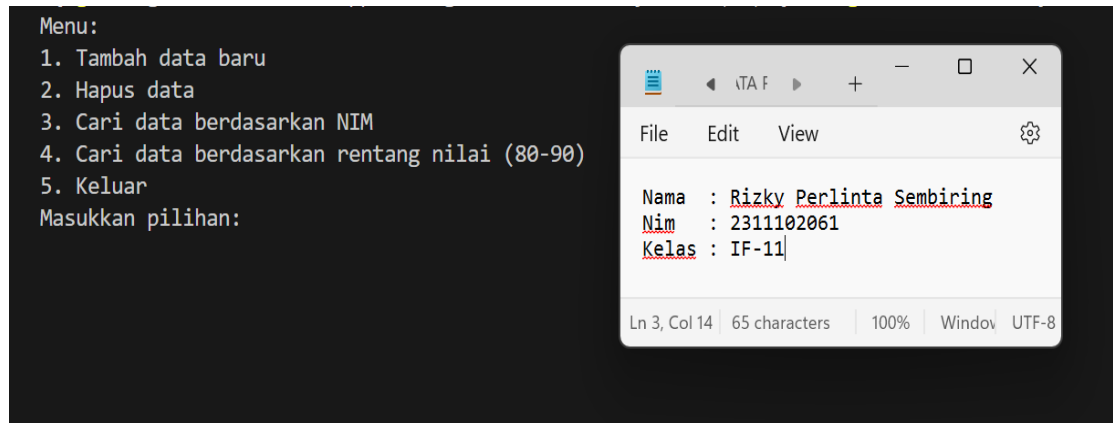
```

    } while (pilihan != 5);
    return 0;
}

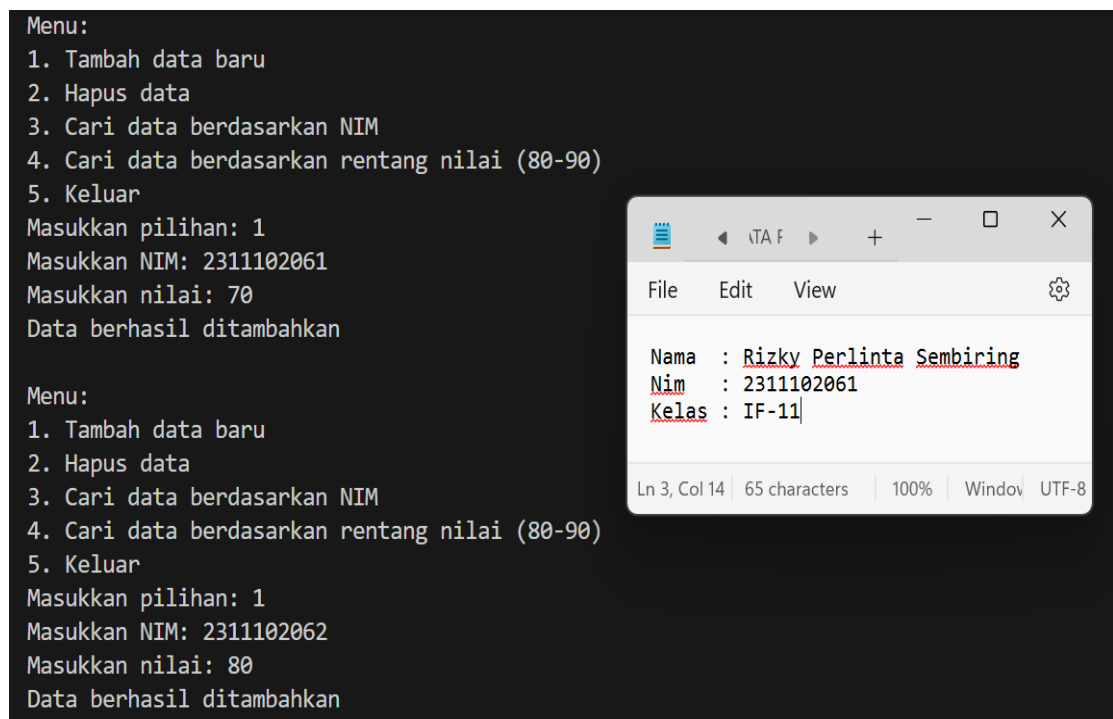
```

Screenshot Output:

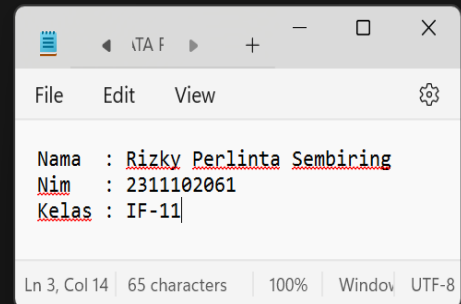
- Tampilan menu awal



- Menambahkan nim dan nilai



```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102063
Masukkan nilai: 90
Data berhasil ditambahkan
```



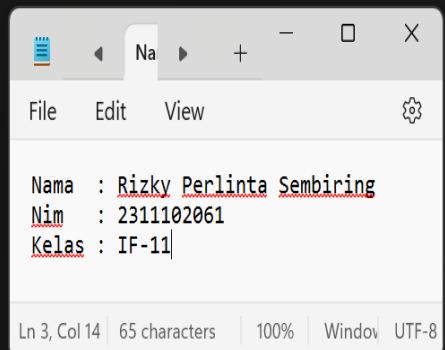
A screenshot of a text editor window. The window has a title bar with a file icon, navigation arrows, and window controls. The menu bar includes File, Edit, View, and a settings icon. The text content is as follows:

```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11|
```

The status bar at the bottom shows "Ln 3, Col 14", "65 characters", "100%", "Window", and "UTF-8".

- Menghapus data

```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 2
Masukkan NIM yang ingin dihapus: 2311102063
Data berhasil dihapus
```



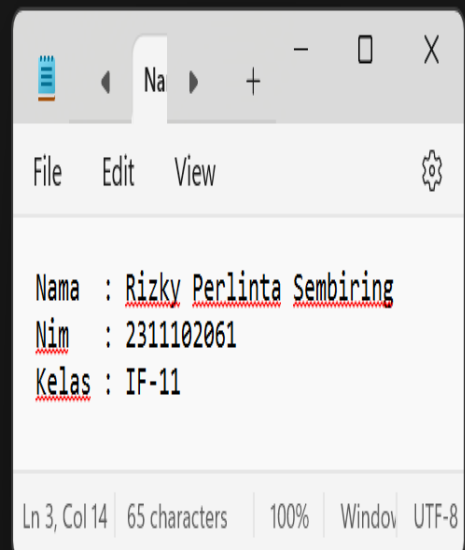
A screenshot of a text editor window. The window has a title bar with a file icon, navigation arrows, and window controls. The menu bar includes File, Edit, View, and a settings icon. The text content is as follows:

```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11|
```

The status bar at the bottom shows "Ln 3, Col 14", "65 characters", "100%", "Window", and "UTF-8".

- Mencari data berdasarkan NIM

```
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 3
Masukkan NIM yang ingin dicari: 2311102061
NIM: 2311102061, Nilai: 70
```



A screenshot of a text editor window. The window has a title bar with a file icon, navigation arrows, and window controls. The menu bar includes File, Edit, View, and a settings icon. The text content is as follows:

```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11|
```

The status bar at the bottom shows "Ln 3, Col 14", "65 characters", "100%", "Window", and "UTF-8".

- Tampilan data ketika mencari nilai di rentang (80-90)

```

Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 4
Masukkan rentang nilai yang ingin dicari (misal: 80 90): 80
85
NIM: 2311102062, Nilai: 80
  
```

Notepad window content:

```

Nama : Rizky Perlenta Sembiring
Nim : 2311102061
Kelas : IF-11
  
```

- Tampilan keluar

```

Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 5
PS E:\Semester 2\Pratikum struktur data\unguided
  
```

Notepad window content:

```

Nama : Rizky Perlenta Sembiring
Nim : 2311102061
Kelas : IF-11
  
```

Deskripsi Program :

- Library yang di-include:

`iostream`: Untuk operasi input-output.

`unordered_map`: Untuk menggunakan struktur data hash map yang tidak terurut. `vector`: Untuk menggunakan struktur data vector.

`using namespace std;`: Menggunakan namespace std secara keseluruhan.

- Struktur Data:

`struct Mahasiswa`: Mendefinisikan struktur data Mahasiswa yang memiliki dua atribut, yaitu NIM (Nomor Induk Mahasiswa) dan nilai.

- Kelas HashTable:

- unordered_map tabel;; Mendefinisikan sebuah hash map dengan kunci berupa string (NIM) dan nilai berupa Mahasiswa.

-void tambahData(Mahasiswa mahasiswa): Menambahkan data mahasiswa ke dalam hash table.

-void hapusData(string NIM): Menghapus data mahasiswa dari hash table berdasarkan NIM.

-Mahasiswa cariDataBerdasarkanNIM(string NIM): Mencari data mahasiswa berdasarkan NIM.

-vector cariDataBerdasarkanRentangNilai(int nilaiMin, int nilaiMax): Mencari data mahasiswa berdasarkan rentang nilai.

- Fungsi main():

Membuat objek HashTable.

Menampilkan menu dan menerima input pilihan dari pengguna.

Melakukan aksi sesuai dengan pilihan pengguna (menambah data baru, menghapus data, mencari data berdasarkan NIM, mencari data berdasarkan rentang nilai, atau keluar dari program).

Program akan terus berjalan sampai pengguna memilih untuk keluar (pilihan 5).

D. Kesimpulan

Uraian di atas dengan jelas menjelaskan apa itu hash table di C++ dan bagaimana penggunaannya dalam program untuk menyimpan pasangan nilai kunci. hash table digunakan karena sangat cepat untuk mengakses dan memproses data. Ada banyak kemungkinan tabrakan saat menghitung indeks menggunakan fungsi hash. Kita harus selalu mencari metode yang akan membantu mencegah tabrakan. Jadi kita harus sangat berhati-hati saat menerapkannya dalam program.

E. Referensi

[1] Asisten Pratikum “Modul 4 HASH TABLE”, Learning Management System, 2024.

[2] yashi goyal (2021, 13 april). “HASH TABLE”.

Diakses Pada 9 Mei 2024, dari

<https://www.educba.com/c-plus-plus-hash-table/>