

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 5
“HASH TABLE”**



**DISUSUN OLEH :
BINTANG YUDHISTIRA
2311102052**

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

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

A. Dasar Teori

Hash Table (Tabel Hash) adalah struktur data yang digunakan untuk menyimpan dan mengelola kumpulan data, dimana setiap elemen dalam kumpulan data memiliki kunci (key) yang unik dan nilainya (value) terkait. Konsep utama di balik tabel hash adalah penggunaan fungsi hash untuk mengonversi kunci menjadi alamat atau indeks di dalam tabel, sehingga memungkinkan pencarian dan pengambilan data dengan efisien.

Pada dasarnya, tabel hash berfungsi sebagai penyimpanan asosiatif, di mana data disimpan dalam bentuk pasangan kunci-nilai. Proses ini melibatkan penggunaan fungsi hash untuk menghasilkan indeks unik dari kunci, dan nilai yang sesuai akan disimpan pada indeks tersebut..

Salah satu keuntungan utama dari tabel hash adalah kecepatan akses dan pencarian data yang sangat efisien. Ketika kita ingin mencari nilai berdasarkan kunci, kita hanya perlu melakukan proses hashing pada kunci untuk mengetahui lokasi penyimpanan datanya, sehingga tidak perlu mencari secara berurutan seperti pada struktur data lainnya seperti array atau list.

Namun, perlu diingat bahwa pada situasi tertentu, bisa saja terjadi tabrakan hash (hash collision), yaitu ketika dua kunci berbeda menghasilkan indeks yang sama. Untuk menangani tabrakan ini, metode penyelesaian tabrakan (collision resolution) digunakan, seperti linear probing, chaining, atau double hashing. Hash table digunakan dalam berbagai aplikasi, termasuk dalam basis data, kamus (dictionary), dan kebanyakan algoritma yang membutuhkan pencarian atau pengelompokan data dengan cepat dan efisien.

FUNGSI HASH TABLE

Fungsi utama dari Hash Table (Tabel Hash) adalah untuk menyediakan struktur data yang memungkinkan pencarian, penyisipan, dan penghapusan data dengan efisien. Berikut adalah beberapa fungsi utama dari Hash Table :

1. Pencarian (Search)

Hash Table memungkinkan pencarian data berdasarkan kunci (key) dengan cepat. Saat kita ingin mencari nilai berdasarkan kunci, fungsi hash akan mengonversi kunci menjadi

indeks, dan nilai yang sesuai dapat diakses langsung dari indeks tersebut, menghasilkan waktu akses yang konstan ($O(1)$).

2. Penyisipan (Insertion)

Hash Table memungkinkan penambahan data baru dengan kunci dan nilainya. Saat kita ingin menyisipkan data baru, fungsi hash akan mengonversi kunci menjadi indeks, dan nilai tersebut akan disimpan pada indeks yang sesuai. Proses ini dapat dilakukan dengan waktu yang konstan ($O(1)$) pada kebanyakan kasus, membuatnya sangat efisien.

3. Penghapusan (Deletion)

Hash Table memungkinkan penghapusan data berdasarkan kunci. Saat kita ingin menghapus data, hash table akan menggunakan fungsi hash untuk menemukan indeks data yang sesuai dengan kunci dan menghapusnya dari tabel. Seperti pencarian dan penyisipan, operasi penghapusan juga berjalan dengan waktu yang konstan ($O(1)$) dalam kebanyakan kasus.

4. Asosiasi Kunci-Nilai (Key-Value Association)

Hash Table menyimpan data dalam bentuk pasangan kunci-nilai (key-value). Ini memungkinkan kita untuk mengaitkan kunci dengan nilai tertentu sehingga kita dapat dengan mudah mengakses nilai tersebut ketika diberikan kunci.

5. Kecepatan Akses

Salah satu keunggulan utama dari Hash Table adalah kecepatan aksesnya. Dengan menggunakan fungsi hash, proses mencari dan mengakses data menjadi sangat cepat karena kita dapat langsung menuju lokasi data tanpa perlu mencari secara berurutan.

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;
        }
        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 Visual Studio IDE with a C++ project named "Prak 5 Hash Table". The Explorer pane on the left shows the project structure with files: guided1.cpp, guided1.exe, guided2.cpp, and guided2.exe. The main editor displays the code for guided1.cpp, which includes a simple hash function and a linked list structure. The terminal window at the bottom shows the execution of the program, which prompts for a key and displays the corresponding value from the hash table. A small window titled "Nama : Bintang Yudhistira" and "NIM : 2311102052" is also visible in the foreground.

```
1 #include <iostream>
2 using namespace std;
3 const int MAX_SIZE = 10;
4 // Fungsi hash sederhana
5 int hash_func(int key)
6 {
7     return key % MAX_SIZE;
8 }
9 // Struktur data untuk setiap node
10 struct Node
11 {
12     int key;
13     int value;
14     Node *next;
15     Node(int key, int value) : key(key), value(value),
16                               next(nullptr) {}
17 }
```

```
{ g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semeste
```

Deskripsi Program :

Tabel hash dalam kode C++ tersebut adalah struktur data yang memungkinkan penyimpanan pasangan kunci-nilai dengan efisiensi tinggi. Fungsi hash sederhana digunakan untuk menentukan indeks dalam tabel, dan setiap node dalam tabel memiliki kunci, nilai, dan pointer ke node berikutnya. Kelas HashTable menyediakan operasi dasar seperti penyisipan, pencarian, penghapusan, dan penelusuran. Dalam fungsi main, objek HashTable dibuat dan operasi-operasi tersebut diterapkan untuk menunjukkan cara menggunakan tabel hash secara praktis. Dengan menggunakan tabel hash, program dapat mengelola data dengan cepat dan efisien, terutama saat memiliki jumlah data yang besar.

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
<< "]\n";
            }
        }
        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 C++ IDE with the following components:

- EXPLORER:** Shows the project structure with files `guided1.cpp`, `guided1.exe`, `guided2.cpp`, and `guided2.exe`.
- TERMINAL:** Displays the program's output:


```

2: 20
3: 30
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 5 Hash Table> .\guided2
{ g++ guided2.cpp -o guided2 } ; if ($?) { .\guided2 }
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\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 5 Hash Table>

```
- Code Window:** Shows a small window with the text:


```

Nama : Bintang Yudhistira
NIM : 2311102052

```

Deskripsi Program :

Kode dimulai dengan beberapa inklusi, definisi konstanta, dan deklarasi variabel global. Kemudian, ada definisi dari dua kelas: `HashNode`, yang merepresentasikan node dalam tabel hash, dan `HashMap`, yang mewakili struktur data `HashMap` itu sendiri. Kelas `HashMap` memiliki metode untuk melakukan operasi dasar pada `HashMap`, seperti `insert` (untuk menambahkan entri), `remove` (untuk menghapus entri), dan `searchByName` (untuk

mencari nomor telepon berdasarkan nama). Selain itu, ada juga metode print untuk mencetak isi seluruh tabel hash. Di dalam fungsi main, objek employee_map dari kelas HashMap dibuat, dan beberapa entri ditambahkan ke dalamnya dengan menggunakan metode insert. Selanjutnya, nomor telepon dari beberapa nama dicari dan dicetak menggunakan metode searchByName. Salah satu entri dihapus menggunakan metode remove, dan tabel hash dicetak menggunakan metode print.

C. Unguided

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;

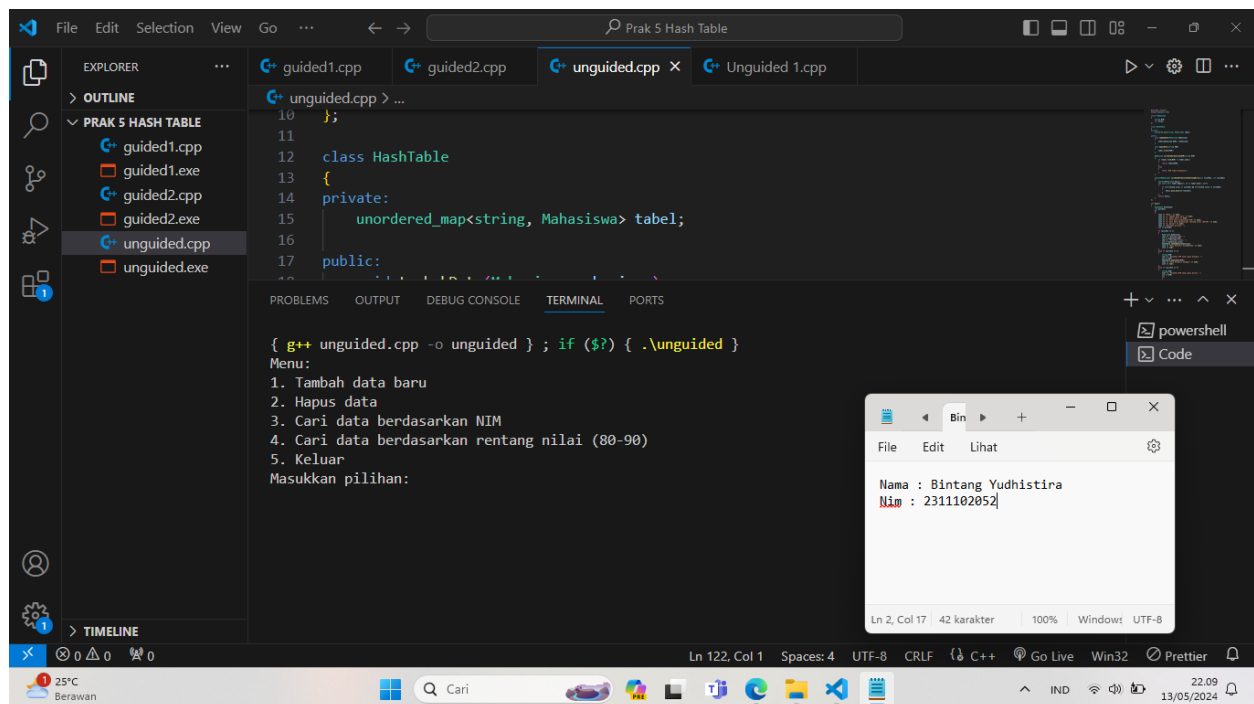
        vector<Mahasiswa> hasil =
hashTable.cariDataBerdasarkanRentangNilai(nilaiMin, nilaiMax);
        for (Mahasiswa mahasiswa : hasil)
        {
            cout << "NIM: " << mahasiswa.NIM << ", Nilai: " <<
mahasiswa.nilai << endl;
        }
        cout << endl;
    }
} while (pilihan != 5);
return 0;
}

```

Screenshot output

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)

• Tampilan Menu



- Tampilan Setelah ditambahkan Data baru

```
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 5 Hash Table> cd "c:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 5 Hash Table\" ; if ($?) { g++ unguided.cpp -o unguided } ; if ($?) { .\unguided }
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: 2311102052
Masukkan nilai: 80
Data berhasil ditambahkan

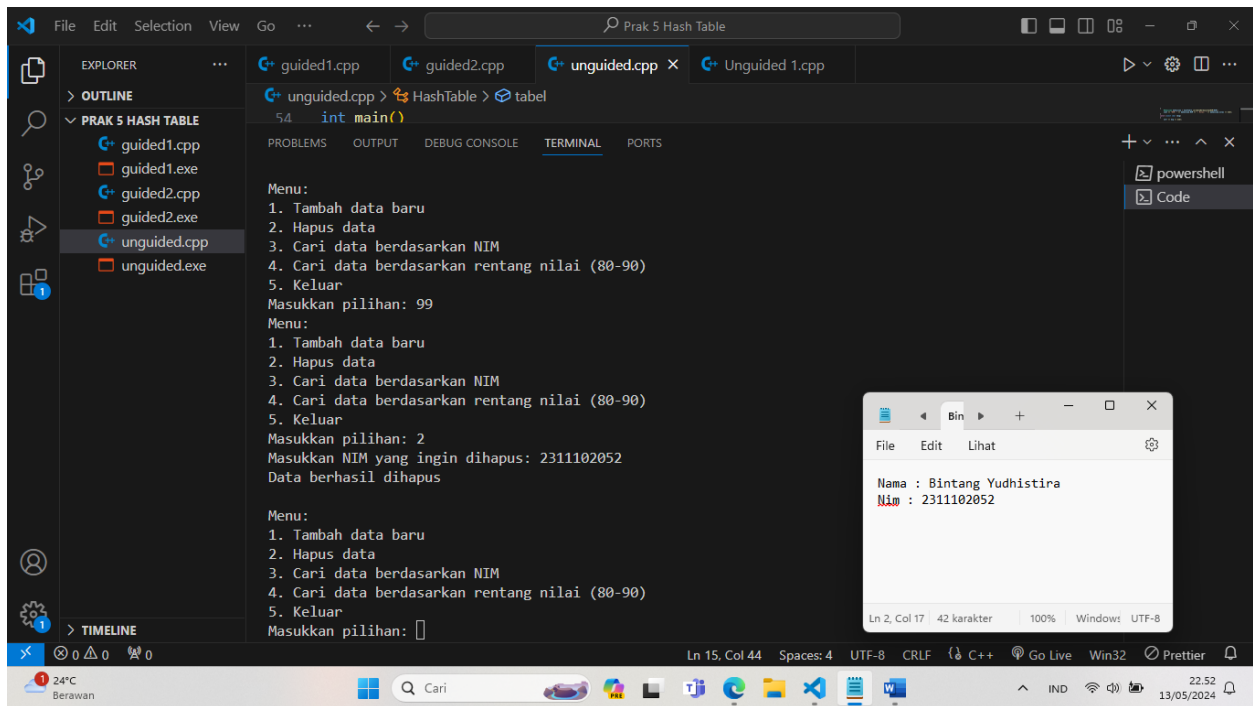
Menu:
1. Tambah data baru
```

```
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: 2311102051
Masukkan nilai: 70
Data berhasil ditambahkan

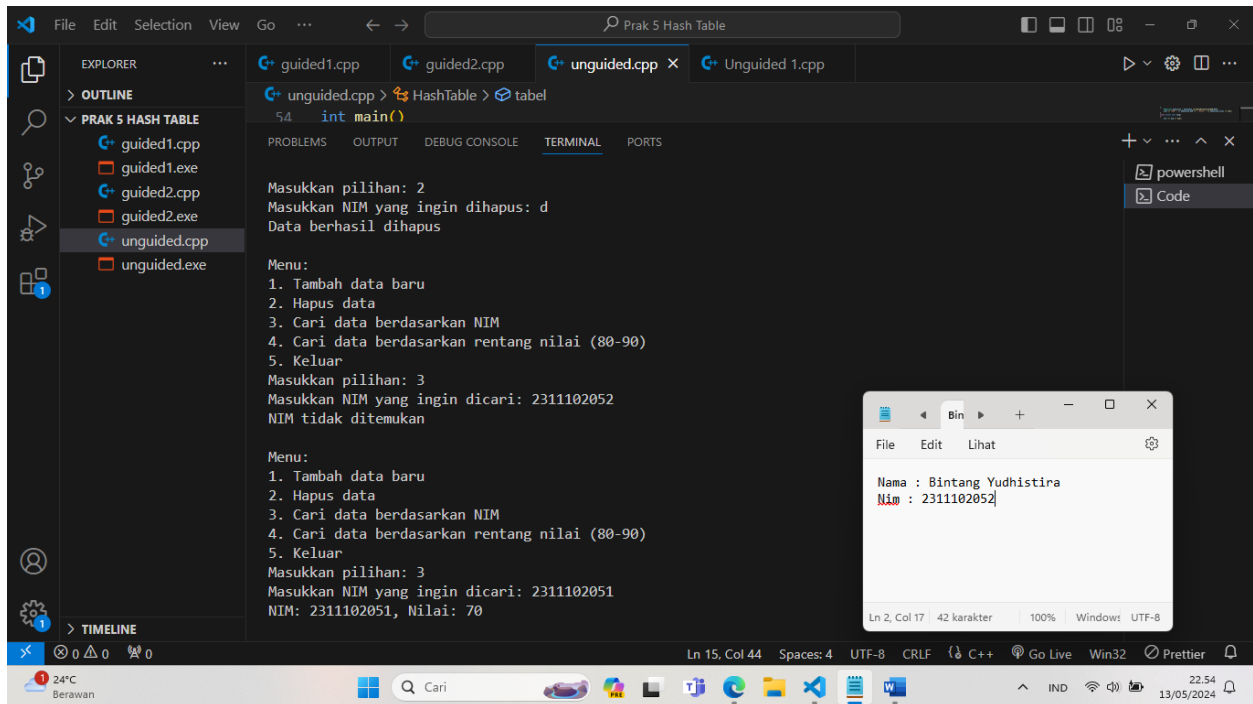
Menu:
1. Tambah data baru
2. Hapus data
3. Cari data berdasarkan NIM
4. Cari data berdasarkan rentang nilai (80-90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102052
Masukkan nilai: 90
Data berhasil ditambahkan

Menu:
1. Tambah data baru
2. Hapus data
```

- Tampilan Menghapus Data



- Tampilan Mencari Data berdasarkan NIM



-
- The screenshot shows a Windows 11 desktop environment. The primary application is Visual Studio Code, which is open to a C++ file named 'unguided.cpp'. The file contains a menu-driven hash table application. The program prompts the user to enter a choice (1-5), a NIM, and a value, and then displays the results. The output of the program is shown in a Notepad window, displaying the name 'Bintang Yudhistira' and the NIM '2311102052'.
- Visual Studio Code Editor Content:**
- ```

54 int main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 2311102088
Masukkan nilai: 81
Data berhasil ditambahkan

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): 81
90
NIM: 2311102088, Nilai: 81

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:

```
- Notepad Window Content:**
- ```

File Edit Lihat

Nama : Bintang Yudhistira
Nim : 2311102052
  
```
- The status bar at the bottom of the VS Code window shows the current cursor position as 'Ln 15, Col 44' and the file encoding as 'UTF-8'. The Windows taskbar at the bottom displays the system clock as '22:56' on '13/05/2024'.

-
- The screenshot shows a Windows 11 desktop environment. The primary application is Visual Studio Code, which is open to a C++ file named `unguided.cpp`. The file contains a menu-driven hash table implementation in C++. The code includes a `main` function that prompts the user to enter a choice (1-5), a NIM, and a range of values. It then performs operations like adding, deleting, or searching for data in a hash table. The output of the program is visible in the terminal window, showing the user's input and the program's response.
- The PowerShell window is open, showing the output of the program. The output includes the user's input and the program's response, such as "Masukkan pilihan: 1", "Masukkan NIM: 2311102088", "Masukkan nilai: 81", and "Data berhasil ditambahkan".

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<string, Mahasiswa> 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<Mahasiswa> 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

Hash table adalah sebuah struktur data yang sangat efisien digunakan untuk menyimpan data dalam bentuk pasangan kunci-nilai. Keunggulan utama dari hash table adalah kemampuannya untuk menyediakan operasi penyimpanan, pencarian, dan penghapusan data dengan kompleksitas waktu rata-rata yang konstan, yaitu $O(1)$ dalam kasus terbaik, asalkan fungsi hash yang digunakan seimbang dan efisien. Salah satu kelebihan utama dari hash table adalah efisiensi dalam melakukan pencarian data. Dengan fungsi hash yang baik, pencarian data dapat dilakukan dalam waktu konstan, tanpa memperhatikan jumlah data yang tersimpan. Namun, pengelolaan kolisi merupakan salah satu tantangan utama dalam penggunaan hash table. Penanganan kolisi dapat dilakukan melalui berbagai metode seperti chaining dan open addressing. Selain itu, kualitas dari fungsi hash sangat mempengaruhi kinerja dari hash table tersebut. Dengan menggunakan hash table secara bijak, kita dapat mengelola dan memanipulasi data dengan efisien dalam berbagai aplikasi, mulai dari basis data hingga algoritma pencarian.

E. Referensi

[1] Asisten Pratikum “Modul 5 Hash Table”, Learning Management System, 2024.

[2] Educative. Hash Table in C++. 3 Maret 2024.

Diakses Pada 10 Mei 2024, dari

<https://www.educative.io/answers/singly-linked-list-in-cpp>

[3] DosenIT (2019, juli). “Hash Table”.

Diakses pada 10 Mei 2024, dari

[Hash Table: Pengertian, Fungsi dan Cara Membuat - DosenIT.com](https://dosenit.com/Hash-Table-Pengertian-Fungsi-dan-Cara-Membuat)