

LAPORAN PRAKTIKUM

Modul V HASH TABLE



Disusun oleh:

Muhammad Irsyad : **2211102048**

Dosen

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

**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2023**

BAB I

Tujuan Pembelajaran

1. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
2. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

BAB II

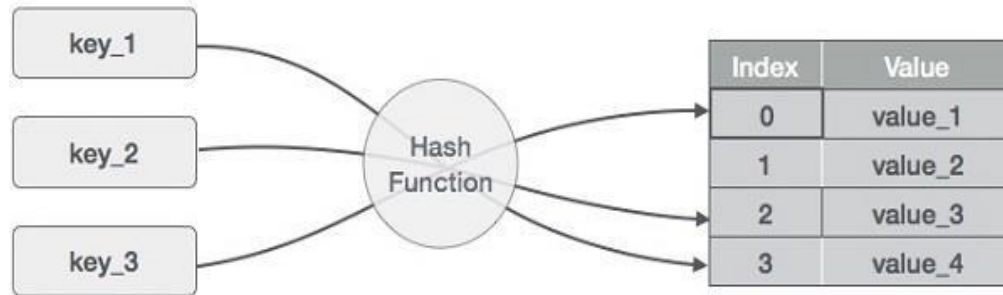
Dasar Teori

1. Pengertian Hash Table

Tabel Hash adalah struktur data yang digunakan untuk menyimpan pasangan kunci/nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vector) dan fungsi hash. Dengan menggunakan fungsi hash yang baik hashing dapat berjalan dengan baik. **Hashing** adalah teknik untuk mengubah rentang nilai kunci menjadi rentang indeks array.

Array menyimpan data dalam slot-slot yang disebut bucket. Setiap bucket dapat menampung satu atau beberapa item data. Fungsi hash digunakan untuk menghasilkan nilai unik dari setiap item data, yang digunakan sebagai indeks array. Dengan cara ini, hash table memungkinkan pencarian data dalam waktu yang konstan ($O(1)$) dalam kasus terbaik

Sistem hash table bekerja dengan cara mengambil input kunci dan memetakannya ke nilai indeks array menggunakan fungsi hash. Kemudian, data disimpan pada posisi indeks array yang dihasilkan oleh fungsi hash. Ketika data perlu dicari, input kunci dijadikan sebagai parameter untuk fungsi hash, dan posisi indeks array yang dihasilkan digunakan untuk mencari data. Dalam kasus hash collision, di mana dua atau lebih data memiliki nilai hash yang sama, hash table menyimpan data tersebut dalam slot yang sama dengan Teknik yang disebut chaining.



Gambar 1. Struktur Data Hash

2. 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.

Contoh: Pertimbangkan sebuah array sebagai Peta di mana kuncinya adalah indeks dan nilainya adalah value pada indeks itu. Jadi untuk array A jika kita memiliki indeks i yang akan diperlakukan sebagai kunci maka kita dapat menemukan nilainya hanya dengan mencari value pada $A[i]$.

Tipe fungsi hash :

- a. Division Method.
- b. Mid Square Method.
- c. Folding Method.
- d. Multiplication Method.

Untuk mencapai mekanisme hashing yang baik, penting untuk memiliki fungsi hash yang baik dengan persyaratan dasar sebagai berikut:

- a. Dapat dihitung secara efisien.
- b. Harus mendistribusikan kunci secara seragam (Setiap posisi tabel memiliki kemungkinan yang sama untuk masing-masing.)
- c. Harus meminimalkan tabrakan.
- d. Harus memiliki faktor muatan rendah (jumlah item dalam tabel dibagi dengan ukuran tabel).

3. 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:

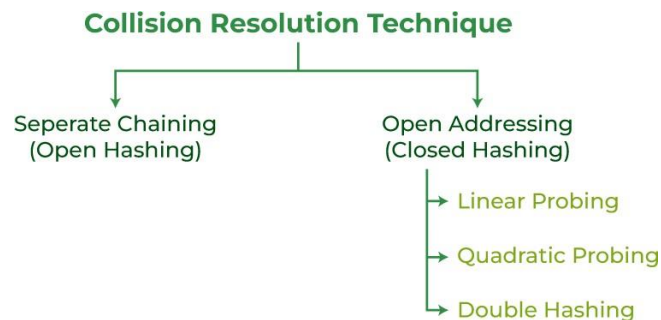
Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

5. Traversal:

Melalui seluruh hash table untuk memproses semua data yang ada dalam tabel

4. 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 :



Gambar 2. Teknik Collision

A. 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 dihash. 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.

B. Closed Hashing

1. 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.

2. Quadratic Probing

Penanganannya hampir sama dengan metode linear, hanya lompatannya tidak satu-satu, tetapi quadratic (12, 22, 32, 42, ...)

3. Double Hashing

Pada saat terjadi collision, terdapat fungsi hash yang kedua untuk menentukan posisinya kembali.

BAB III

LATIHAN KELAS – 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;
        }
    }
}

```

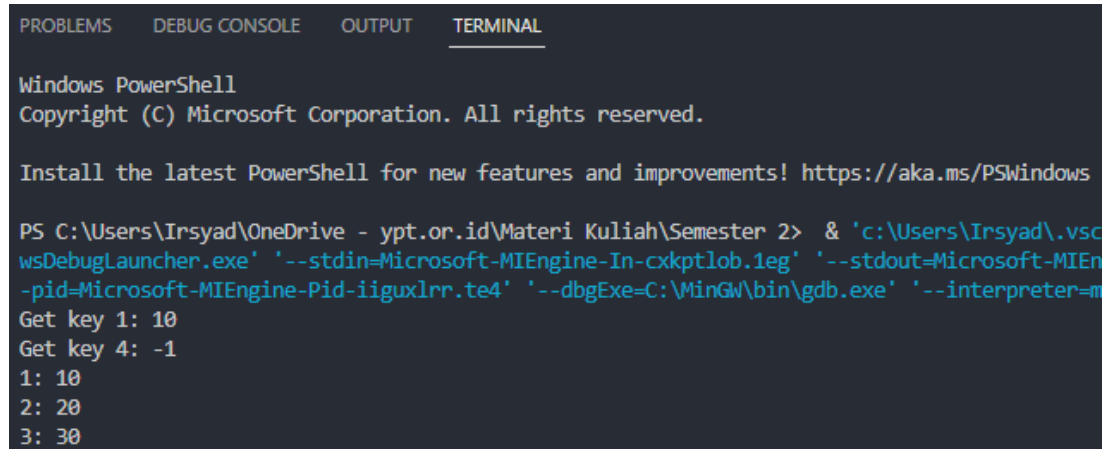


```

        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;
}

```

Screenshot Program



```
PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2> & 'c:\Users\Irsyad\.vscode\bin\DebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-cxkptlob.1eg' '--stdout=Microsoft-MIEngine-Out-cxkptlob.1eg' --pid=Microsoft-MIEngine-Pid-iiguxlrr.te4' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mingw64-gdb-pdb.exe'
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
```

Deskripsi Program

Kode program di atas merupakan implementasi sederhana dari tabel hash (hash table) menggunakan metode chaining untuk menangani tabrakan (collision). Berikut adalah deskripsi kode program tersebut:

1. Baris 1-3: Mengimpor pustaka iostream dan menggunakan ruang nama std.
2. Baris 4: Mendefinisikan konstanta MAX_SIZE dengan nilai 10. Konstanta ini menentukan ukuran maksimum tabel hash.
3. Baris 7-14: Mendefinisikan fungsi hash sederhana bernama `hash_func` yang menerima parameter `key` dan mengembalikan nilai hasil modulus `key` dengan `MAX_SIZE`.
4. Baris 17-22: Mendefinisikan struktur data `Node` yang merepresentasikan setiap elemen dalam tabel hash. Setiap `Node` memiliki `key` (kunci) dan `value` (nilai), serta pointer `next` yang menunjuk ke `Node` berikutnya dalam kasus tabrakan (chaining).
5. Baris 25-45: Mendefinisikan kelas `HashTable`. Kelas ini memiliki anggota data berupa pointer ganda `**table` yang akan menampung array dinamis dari pointer `Node`. Konstruktor `HashTable` akan menginisialisasi `table` dengan array baru yang berukuran `MAX_SIZE`. Destruktor `~HashTable` akan membersihkan memori yang dialokasikan untuk `table` dan setiap `Node` yang ada di dalamnya.

6. Baris 48-64: Mendefinisikan fungsi ``insert`` yang digunakan untuk memasukkan elemen baru ke dalam tabel hash. Fungsi ini akan menggunakan fungsi hash untuk mendapatkan indeks yang tepat dalam ``table``. Jika terjadi tabrakan (elemen dengan kunci yang sama), maka elemen baru akan ditambahkan sebagai ``Node`` baru di awal rantai (chaining).
7. Baris 67-86: Mendefinisikan fungsi ``get`` yang digunakan untuk mencari nilai (value) yang terkait dengan kunci (key) yang diberikan. Fungsi ini akan menggunakan fungsi hash untuk mendapatkan indeks yang tepat dalam ``table``. Jika ditemukan elemen dengan kunci yang sesuai, maka nilai (value) yang terkait akan dikembalikan. Jika tidak ada elemen dengan kunci tersebut, maka fungsi ini akan mengembalikan -1.
8. Baris 89-111: Mendefinisikan fungsi ``remove`` yang digunakan untuk menghapus elemen dengan kunci (key) yang diberikan dari tabel hash. Fungsi ini akan menggunakan fungsi hash untuk mendapatkan indeks yang tepat dalam ``table``. Jika ditemukan elemen dengan kunci yang sesuai, maka elemen tersebut akan dihapus dari rantai (chaining) yang sesuai. Fungsi ini juga akan membersihkan memori yang telah dialokasikan untuk elemen yang dihapus.
9. Baris 114-131: Mendefinisikan fungsi ``traverse`` yang digunakan untuk menelusuri (traversal) semua elemen dalam tabel hash dan mencetak kunci (key) dan nilai (value) mereka ke layar.
10. Baris 134-145: Fungsi ``main`` yang merupakan titik masuk utama program. Pada fungsi ``main`` ini, objek ``HashTable`` dengan nama ``ht`` dibuat. Beberapa operasi dijalankan pada ``ht``, termasuk operasi penambahan (``insert``), pencarian (``get``), penghapusan (``remove``), dan penelusuran (``traverse``).

Kode program ini mengilustrasikan implementasi dasar tabel hash dengan metode chaining menggunakan bahasa pemrograman C++.

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;
            }
        }
    }
}

};

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 Program

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2> & 'c:\Users\Irsyad\.vscode\extensions\ms-vscode.cpptools\bin\DebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-zzhmijp4.dq1' '--stdout=Microsoft-MIEngine-Out-
--pid=Microsoft-MIEngine-Pid-lva1dtyp.1vr' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
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\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2>

```

Deskripsi Program

Program di atas merupakan implementasi sederhana dari hash map yang menggunakan chaining untuk menangani tabrakan (collision). Berikut adalah deskripsi singkat dari kode program tersebut:

1. Baris 1-4: Mendefinisikan header file yang akan digunakan dalam program, yaitu `*<i>iostream</i>*` untuk operasi input-output, `*<i>string</i>*` untuk penggunaan tipe data string, dan `*<i>vector</i>*` untuk penggunaan vector. Selain itu, `*using namespace std</i>*` digunakan untuk menghindari penulisan `*std::</i>* pada setiap pemanggilan objek dari namespace `*std</i>*`.

2. Baris 6-7: Mendefinisikan konstanta ``TABLE_SIZE`` dengan nilai 11. Konstanta ini menentukan ukuran maksimum hash table.
3. Baris 9-16: Mendefinisikan kelas ``HashNode`` yang merepresentasikan setiap node dalam linked list pada setiap slot hash table. Setiap node memiliki atribut ``name`` dan ``phone_number`` untuk menyimpan pasangan nama-nomor telepon.
4. Baris 18-34: Mendefinisikan kelas ``HashMap`` yang merupakan implementasi dari hash map. Kelas ini memiliki atribut ``table`` yang merupakan array vector untuk menyimpan data pada hash table.
 - Metode ``hashFunc(string key)``: Menghitung hash value berdasarkan key (nama) dengan menjumlahkan nilai ASCII dari setiap karakter dalam nama, kemudian mengembalikan hasil modulo dengan ``TABLE_SIZE``.
 - Metode ``insert(string name, string phone_number)``: Menyisipkan pasangan nama-nomor telepon baru ke dalam hash table. Fungsi ini menggunakan metode chaining, yaitu mencari slot hash berdasarkan hash value dari nama, kemudian melakukan pencarian pada linked list pada slot yang sesuai. Jika nama sudah ada dalam linked list, nomor telepon yang terkait akan diperbarui. Jika nama belum ada, node baru akan ditambahkan ke linked list pada slot yang sesuai.
 - Metode ``remove(string name)``: Menghapus pasangan nama-nomor telepon yang terkait dengan nama yang diberikan. Fungsi ini menggunakan metode chaining, yaitu mencari slot hash berdasarkan hash value dari nama, kemudian melakukan pencarian pada linked list pada slot yang sesuai. Jika nama ditemukan, node tersebut akan dihapus dari linked list.
 - Metode ``searchByName(string name)``: Mencari dan mengembalikan nomor telepon yang terkait dengan nama yang diberikan. Fungsi ini menggunakan metode chaining, yaitu mencari slot hash berdasarkan hash value dari nama, kemudian melakukan pencarian pada linked list pada slot yang sesuai. Jika nama ditemukan, nomor telepon yang terkait akan dikembalikan. Jika nama tidak ditemukan, fungsi akan mengembalikan string kosong (`""`).

- Metode ``print()``: Melakukan traversal pada hash table dan mencetak semua pasangan nama-nomor telepon yang ada pada setiap slot. Fungsi ini akan mencetak indeks slot dan semua pasangan nama-nomor telepon yang ada pada linked list pada slot tersebut.

5. Baris 39-54: Fungsi ``main()`` sebagai fungsi utama program.

- Membuat objek ``employee_map`` dari kelas ``HashMap``.
- Menyisipkan pasangan nama-nomor telepon menggunakan metode ``insert()``.
- Mencari nomor telepon berdasarkan nama menggunakan metode ``searchByName()`` dan mencetak hasilnya.
- Menghapus pasangan nama-nomor telepon menggunakan metode ``remove()``.
- Mencetak hash table menggunakan metode ``print()``.

Program ini bertujuan untuk menyimpan pasangan nama-nomor telepon dalam hash table dan memperlihatkan penggunaan operasi seperti penyisipan, pencarian, dan penghapusan data pada hash map.

TUGAS – UNGUIDED

Unguided 1

Source Code

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
const int TABLE_SIZE = 11;
int NIM;
int nilai;
class HashNode
{
public:
    int NIM;
    int nilai;
    HashNode(int NIM, int nilai)
    {
        this->NIM = NIM;
        this->nilai = nilai;
    }
};
class HashMap
{
private:
    vector<HashNode *> table[TABLE_SIZE];
public:
    int hashFunc(long key)
    {
        return key % TABLE_SIZE;
    }
    void insert(int 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(int 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;
            }
        }
    }

    void searchByNIM(int NIM)
    {
        int hash_val = hashFunc(NIM);
        for (auto node : table[hash_val])
        {
            if (node->NIM == NIM)
            {
                cout << "Data dengan NIM " << NIM << " ditemukan:\n";
                cout << "NIM: " << node->NIM << "\n";
                cout << "Nilai: " << node->nilai << "\n";
            }
        }
    }

```

```

        }
        else
        {
            cout << "Data dengan NIM " << NIM << " tidak
ditemukan.\n";
        }
    }
}

void cariDataByRentangNilai(int minNilai, int maxNilai)
{
    cout << "Data dengan rentang nilai " << minNilai << " - " <<
maxNilai << ":\n";
    for (auto pair : table)
    {
        for (auto node : pair)
        {
            if (node->nilai >= minNilai && node->nilai <=
maxNilai)
            {
                cout << "NIM: " << node->NIM << ", Nilai: " <<
node->nilai << "\n";
            }
        }
    }
}

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 employee_map;
    int choice;
    int NIM;
    int nilai, minNilai, maxNilai;

    while (true)
    {
        cout << "Menu:\n";
        cout << "1. Tambah Data\n";
        cout << "2. Hapus Data\n";
        cout << "3. Cari Data berdasarkan NIM\n";
        cout << "4. Cari Data berdasarkan Rentang Nilai (80-90)\n";
        cout << "5. Tampilkan Data\n";
        cout << "6. Keluar\n";
        cout << "Pilih: ";
        cin >> choice;

        switch (choice)
        {
            case 1:
                cout << "Masukkan NIM: ";
                cin >> NIM;
                cout << "Masukkan Nilai: ";

```

```

        cin >> nilai;
        employee_map.insert(NIM, nilai);
        break;
    case 2:
        cout << "Masukkan NIM: ";
        cin >> NIM;
        employee_map.remove(NIM);
        break;
    case 3:
        cout << "Masukkan NIM: ";
        cin >> NIM;
        employee_map.searchByNIM(NIM);
        break;
    case 4:
        employee_map.cariDataByRentangNilai(80, 90);
        break;
    case 5:
        employee_map.print();
    case 6:
        cout << "Terima kasih. Program selesai.\n";
        return 0;
    default:
        cout << "Pilihan tidak valid. Silakan coba lagi.\n";
    }
    cout << endl;
}
}

```

Screenshot Program

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Irsyad\OneDrive - ypt.or.id\Materi Kuliah\Semester 2> & 'c:\Users\Irsyad\.vscode\extensions\ms-vscode.cpptools-1.19.1\bin\DebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-2laxuudd.vzm' '--stdout=Microsoft-MIEngine-Out-exuwrqyd.zgg' '--stderr=Microsoft-MIEngine-Pid-adaxiffj.mhi' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 1
Masukkan NIM: 2048
Masukkan Nilai: 90

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 1
Masukkan NIM: 2049
Masukkan Nilai: 80

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 1
Masukkan NIM: 2054
```

```
Masukkan Nilai: 85

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 1
Masukkan NIM: 2057
Masukkan Nilai: 70

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 2
Masukkan NIM: 2049
Data telah dihapus!

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 3
Masukkan NIM: 2048
Data dengan NIM 2048 ditemukan:
NIM: 2048
Nilai: 90

Menu:
1. Tambah Data
```

```

2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 4
Data dengan rentang nilai 80 - 90:
NIM: 2048, Nilai: 90
NIM: 2054, Nilai: 85

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 5
0: [2057, 70]
1:
2: [2048, 90]
3:
4:
5:
6:
7:
8: [2054, 85]
9:
10:

Menu:
1. Tambah Data
2. Hapus Data
3. Cari Data berdasarkan NIM
4. Cari Data berdasarkan Rentang Nilai (80-90)
5. Tampilkan Data
6. Keluar
Pilih: 6
Terima kasih. Program selesai.

```

Deskripsi Program

Kode program di atas adalah implementasi sederhana dari struktur data hash map (tabel hash) yang digunakan untuk menyimpan data mahasiswa. Berikut adalah deskripsi kode program tersebut:

1. Baris 1-4: Mengimpor pustaka `iostream`, `string`, dan `vector`, serta menggunakan ruang nama `std`.
2. Baris 6-13: Mendefinisikan kelas **HashNode** yang merepresentasikan setiap pasangan NIM dan nilai dalam tabel hash. Setiap **HashNode** memiliki anggota data **NIM** dan **nilai**.
3. Baris 15-30: Mendefinisikan kelas **HashMap** yang merupakan implementasi tabel hash. Kelas ini memiliki anggota data berupa array dari vektor **table** dengan ukuran **TABLE_SIZE**. Metode **hashFunc** digunakan untuk menghasilkan nilai hash berdasarkan kunci (NIM). Metode **insert** digunakan

untuk memasukkan pasangan NIM dan nilai baru ke dalam tabel hash. Jika terdapat pasangan dengan NIM yang sama, nilai akan diperbarui. Metode **remove** digunakan untuk menghapus pasangan dengan NIM yang diberikan dari tabel hash. Metode **searchByNIM** digunakan untuk mencari data berdasarkan NIM. Metode **cariDataByRentangNilai** digunakan untuk mencari data dengan nilai dalam rentang tertentu. Metode **print** digunakan untuk mencetak isi tabel hash.

4. Baris 33-80: Fungsi **main** yang merupakan titik masuk utama program. Pada fungsi **main** ini, objek **employee_map** dari kelas **HashMap** dibuat. Terdapat pula sebuah loop **while (true)** yang akan terus berjalan hingga pengguna memilih untuk keluar.

Dalam loop, pengguna akan diberikan beberapa pilihan melalui menu yang muncul. Pilihan yang tersedia antara lain adalah:

- Menambahkan data mahasiswa dengan memasukkan NIM dan nilai.
- Menghapus data mahasiswa berdasarkan NIM.
- Mencari data mahasiswa berdasarkan NIM.
- Mencari data mahasiswa berdasarkan rentang nilai.
- Menampilkan seluruh data mahasiswa.
- Keluar dari program.

Setiap pilihan akan memanggil metode yang sesuai dari objek **employee_map** untuk melakukan operasi yang diminta.

Kode program ini mengilustrasikan implementasi dasar dari hash map yang menggunakan tabel hash dengan larik vektor untuk menangani tabrakan (collision). Setiap elemen dalam tabel hash adalah vektor yang berisi objek **HashNode** yang menyimpan pasangan NIM dan nilai mahasiswa.

BAB IV

KESIMPULAN

Ketiga kode program di atas merupakan contoh implementasi hash table dengan pendekatan yang sedikit berbeda. Guided 1 menggunakan array dan linked list untuk menangani tabrakan (collision), Guided 2 dan Unguided 1 menggunakan vektor. Fungsi hash yang digunakan juga berbeda-beda, mulai dari sederhana menggunakan operasi modulus hingga memanfaatkan total nilai ASCII karakter dalam kunci (key).

Meskipun memiliki perbedaan dalam struktur data dan fungsi hash, ketiga kode program tersebut memiliki tujuan yang sama, yaitu menyediakan struktur data yang efisien untuk penyimpanan dan pengaksesan data dengan kunci (key) dalam tabel hash. Masing-masing kode program dapat melakukan operasi dasar pada hash table, seperti penambahan data, pencarian data, penghapusan data, dan penelusuran data.

Guided 1 dapat digunakan untuk menyimpan dan mengakses data dengan kunci (key) bilangan bulat. Guided 2 dan Unguided 1 kode program lebih fleksibel dalam hal kunci (key), Guided 2 menggunakan string dan yang Unguided 1 menggunakan bilangan bulat (int).

Dengan mempelajari ketiga kode program ini, kita dapat memahami berbagai pendekatan yang dapat digunakan dalam implementasi hash table. Terlepas dari perbedaan teknis, penggunaan hash table dapat memberikan efisiensi dalam operasi penyimpanan dan pencarian data, serta memungkinkan akses yang cepat dengan kompleksitas waktu yang tetap, tergantung pada implementasinya.