LAPORAN PRAKTIKUM
STURKTUR DATA DAN ALGORITMA

MODUL V
HASH TABLE

Disusun Oleh :
Muhammad Dani Ayubi
2311102003

Dosen Pengmapu:

Anggi Zafia, S.T., M.Eng

LABORATORIUM MULTIMEDIA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO
2024

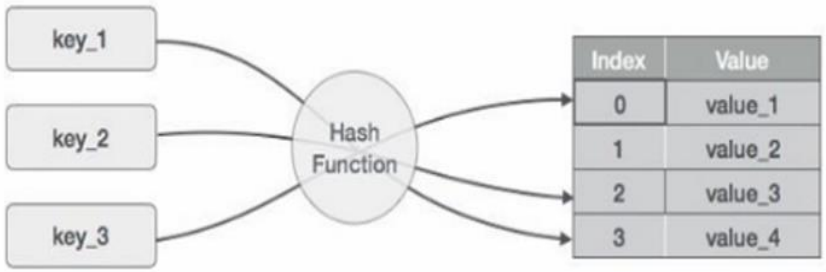
BAB I
TUJUAN PRAKTIKUM

a. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code
b. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman

BAB II
DASAR TEORI

a. Pengertian Hash Table

Hash Table adalah struktur data yang mengorganisir data ke dalam pasangan kunci-nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vektor) dan fungsi hash. 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 memetakkannya 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.



b. Fungsi Hash Table

Fungsi hash membuat pemetaan antara kunci dan nilai, hal ini dilakukan melalui penggunaan rumus matematika yang dikenal sebagai fungsi hash. Hasil dari fungsi hash disebut sebagai nilai hash atau hash. Nilai hash adalah representasi dari string karakter asli tetapi biasanya lebih kecil dari aslinya.

c. Operasi Hash Table

1. Insertion:
   Memasukkan data baru ke dalam hash table dengan memanggil fungsi hash untuk menentukan posisi bucket yang tepat, dan kemudian menambahkan data ke bucket tersebut.
2. Deletion:
   Menghapus data dari hash table dengan mencari data menggunakan fungsi hash, dan kemudian menghapusnya dari bucket yang sesuai.
3. Searching:
   Mencari data dalam hash table dengan memasukkan input kunci ke fungsi hash untuk menentukan posisi bucket, dan kemudian mencari data di dalam bucket yang sesuai.
4. Update:
   Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.
5. Traversal:
   Melalui seluruh hash table untuk memproses semua data yang ada dalam tabel.

BAB III

GUIDED

1. Guided 1

Source code

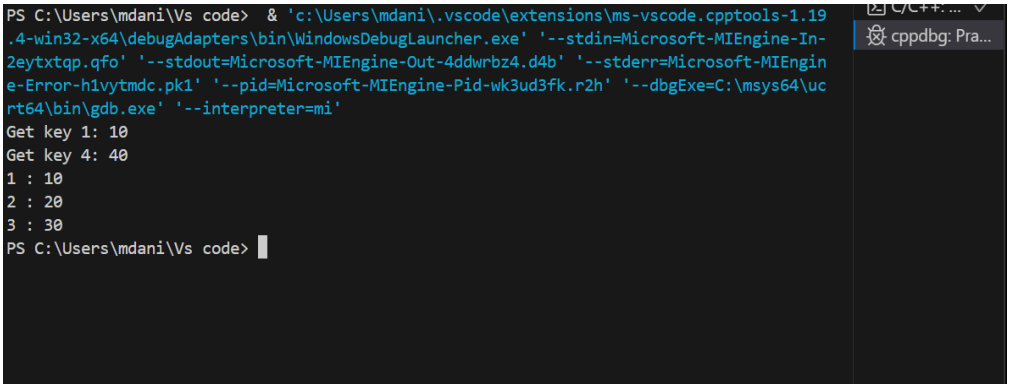
```
#include <iostream>
#include <string> // Untuk dukungan tipe data string

using namespace std;

const int MAX_SIZE = 10;

// Fungsi Hash Untuk Tipe Data int
int hash_func(int key)
{
    return key % MAX_SIZE;
}

// Fungsi Hash Untuk Tipe Data string
int hash_func(const string& key)
{
    int hash_value = 0;
    for (char ch : key)
    {
        hash_value += ch;
    }
    return hash_value % 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 40;
    }

    // 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); // Menghapus kunci yang tidak ada

    // Traversal
    ht.traverse();

    return 0;
}
```

Screenshoot program



Deskripsi program

Program di atas adalah implementasi sederhana dari tabel hash (hash table) menggunakan metode chaining untuk menangani tabrakan (collision). Tabel hash adalah struktur data yang digunakan untuk menyimpan pasangan kunci-nilai (key value) di mana kunci digunakan sebagai indeks untuk mengakses nilai yang terkait. Langkah-langkah yang dilakukan dalam program ini adalah sebagai berikut:

1. Deklarasi konstanta MAX_SIZE yang menentukan ukuran maksimum tabel hash.
2. Definisi fungsi hash_func() yang mengembalikan indeks dalam tabel hash berdasarkan operasi modulo dengan MAX_SIZE.
3. Definisi struktur Node yang memiliki anggota data key, value, dan next yang berfungsi sebagai pointer ke Node berikutnya dalam kasus terjadinya tabrakan.
4. Definisi kelas HashTable dengan atribut table yang merupakan array dari pointer Node.
5. Konstruktor HashTable() yang menginisialisasi tabel hash dengan array kosong.
6. Destruktor ~HashTable() yang menghapus semua node dalam tabel hash dan membebaskan memori yang dialokasikan.
7. Metode insert(int key, int value) untuk menyisipkan pasangan kunci-nilai ke dalam tabel hash dengan menggunakan metode chaining.
8. Metode get(int key) untuk mencari nilai yang terkait dengan kunci tertentu dalam tabel hash.
9. Metode remove(int key) untuk menghapus pasangan kunci-nilai dengan kunci tertentu dari tabel hash.
10. Metode traverse() untuk melintasi seluruh tabel hash dan mencetak pasangan kunci-nilai yang ada.

Fungsi main() sebagai titik masuk program utama yang menguji implementasi tabel hash dengan melakukan operasi penambahan, pencarian, penghapusan, dan traversal pada tabel hash. Dengan menggunakan tabel hash dan metode chaining, program ini memungkinkan penyimpanan efisien dan pencarian cepat nilai-nilai yang terkait dengan kunci tertentu.

2. Guided 2

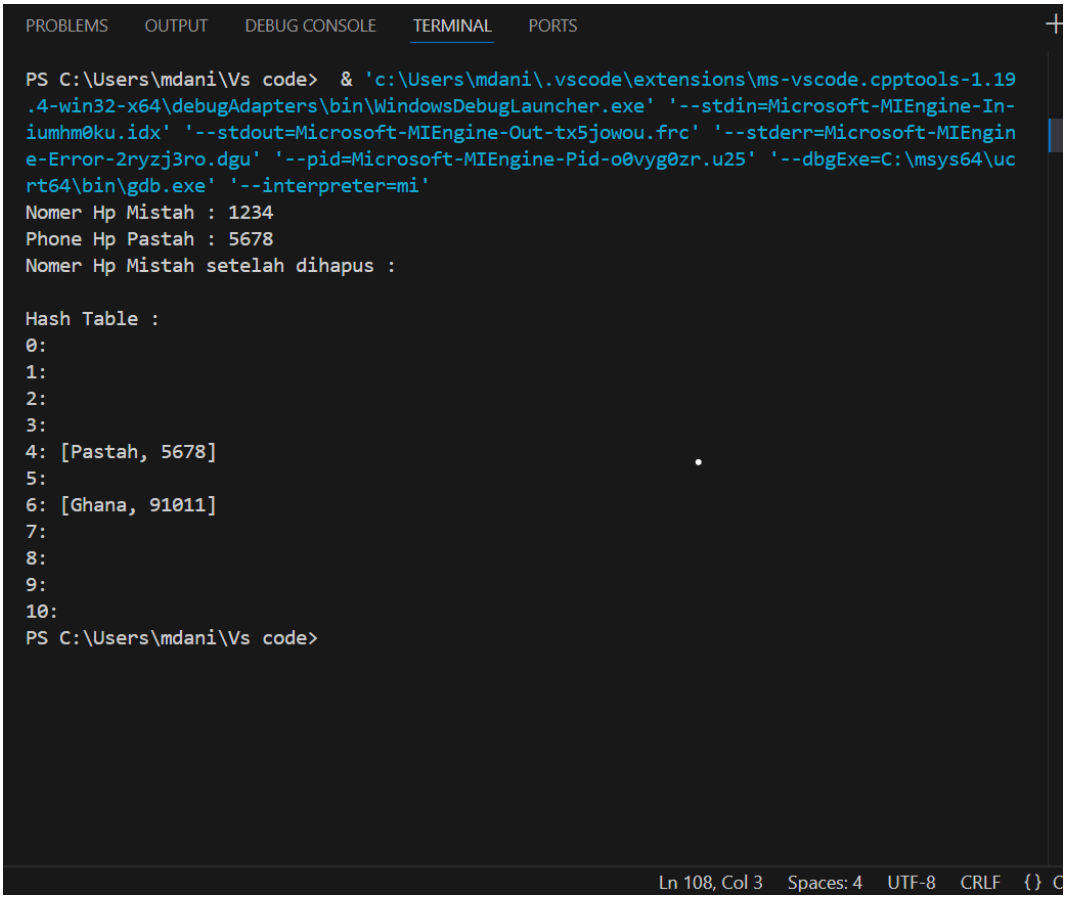
Source code

```
#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 << "]";
                }
```

```
            }
            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;
}
```

Screenshoot program



Deskripsi program

Program di atas merupakan implementasi sederhana dari struktur data HashMap dalam bahasa C++. Program ini menggunakan HashMap untuk menyimpan nomor telepon karyawan berdasarkan nama mereka.

• Program ini menggunakan library , , dan .
• Program mendefinisikan konstanta TABLE_SIZE dengan nilai 11.
• Program mendefinisikan kelas HashNode yang digunakan sebagai wadah untuk menyimpan pasangan nama dan nomor telepon.
• Program mendefinisikan kelas HashMap yang merupakan implementasi struktur data HashMap.
• Kelas HashMap memiliki metode untuk menghitung hash, memasukkan pasangan nama dan nomor telepon, menghapus pasangan, mencari nomor telepon berdasarkan nama, dan mencetak isi HashMap.
• Di dalam fungsi main(), program membuat objek employee_map dari kelas HashMap dan melakukan beberapa operasi seperti memasukkan pasangan nama dan nomor telepon, mencari nomor telepon

berdasarkan nama, menghapus pasangan, dan mencetak isi HashMap. Program tersebut dapat digunakan untuk menyimpan dan mengelola data nomor telepon karyawan berdasarkan nama mereka.

BAB IV

UNGUIDED

1. Unguided 1

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;
}
```

Screenshoot program

```
PS C:\Users\mdani\Vs code>  & 'c:\Users\mdani\.vscode\extensions\ms-vscode.cpptools-1
.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-
qfaybzb5.avp' '--stdout=Microsoft-MIEngine-Out-ujdfiqxz.ars' '--stderr=Microsoft-MIEn
e-Error-pismcmc5.uny' '--pid=Microsoft-MIEngine-Pid-sj0dxhaf.sst' '--dbgExe=C:\msys64
rt64\bin\gdb.exe' '--interpreter=mi'
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: 2311102001
Masukkan nilai: 82
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: 2311102002
Masukkan nilai: 89
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: 2311102003
Masukkan nilai: 88
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: 3
Masukkan NIM yang ingin dicari: 2311102003
NIM: 2311102003, Nilai: 88

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 90
NIM: 2311102003, Nilai: 88
NIM: 2311102002, Nilai: 89
NIM: 2311102001, Nilai: 82

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: 2311102001
Data berhasil dihapus
```

```
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 90
NIM: 2311102003, Nilai: 88
NIM: 2311102002, Nilai: 89
```

Deskripsi Program

Program di atas merupakan implementasi sederhana dari struktur data HashTable menggunakan unordered_map dalam bahasa C++. Program ini digunakan untuk menyimpan data mahasiswa berdasarkan NIM dan nilai mereka.

• Program ini menggunakan library , , dan .

• Program mendefinisikan struct Mahasiswa yang memiliki dua atribut, yaitu NIM (Nomor Induk Mahasiswa) dan nilai.

• Program mendefinisikan kelas HashTable yang memiliki atribut tabel berupa unordered_map untuk menyimpan data mahasiswa.

• Kelas HashTable memiliki metode tambahData untuk menambahkan data mahasiswa ke tabel menggunakan NIM sebagai kunci.

• Kelas HashTable memiliki metode hapusData untuk menghapus data mahasiswa dari tabel berdasarkan NIM.

• Kelas HashTable memiliki metode cariDataBerdasarkanNIM untuk mencari data mahasiswa berdasarkan NIM. Jika ditemukan, data mahasiswa akan dikembalikan; jika tidak ditemukan, akan dilempar pesan "NIM tidak ditemukan".

• Kelas HashTable memiliki metode cariDataBerdasarkanRentangNilai untuk mencari data mahasiswa berdasarkan rentang nilai. Data mahasiswa dengan nilai di antara nilaiMin dan nilaiMax akan dikumpulkan dalam vector dan dikembalikan sebagai hasil pencarian.

• Di dalam fungsi main(), program membuat objek hashTable dari kelas HashTable dan menampilkan menu interaktif untuk pengguna.

• Pengguna dapat memilih untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, mencari data berdasarkan rentang nilai, atau keluar dari program.

• Setiap pilihan menu akan memanggil metode yang sesuai dari hashTable untuk melakukan operasi yang diminta dan menampilkan hasilnya.

Program tersebut dapat digunakan untuk menyimpan dan mengelola data mahasiswa berdasarkan NIM dan nilai mereka.