

**LAPORAN PRAKTIKUM STRUKTUR DATA  
DAN ALGORITMA**

**MODUL 5  
“HASH TABLE”**



**Disusun Oleh :**

NAMA : Fadhel Yussie Ramadhan

NIM : 2311102322

**Dosen**

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

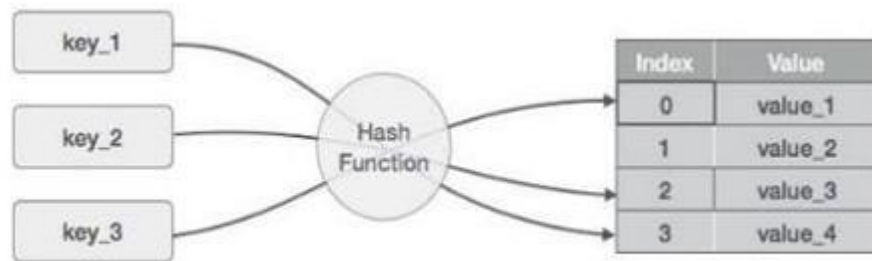
**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## A. TUJUAN PRAKTIKUM

- a. Mahasiswa mampu menjelaskan definisi dan konsep dari Hash Code b. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman B. 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 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.Guided (berisi screenshot source code & output program disertai penjelasannya)



```

#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;

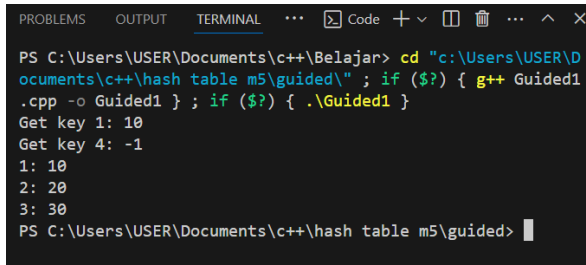
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

## Screenshots Output



```
PS C:\Users\USER\Documents\c++\Belajar> cd "C:\Users\USER\Documents\c++\hash table m5\guided\" ; if ($?) { g++ Guided1.cpp -o Guided1 } ; if ($?) { .\Guided1 }
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\Users\USER\Documents\c++\hash table m5\guided>
```

## Deskripsi:

Program Hash Table dengan teknik Separate Chaining. fungsinya untuk:

1. Memasukkan pasangan kunci-nilai (insert)
2. Mencari nilai berdasarkan kunci (get)
3. Menghapus pasangan kunci-nilai (remove)
4. Mencetak semua pasangan kunci-nilai (traverse)

Hash Table digunakan untuk penyimpanan dan pencarian data berdasarkan kunci dengan waktu rata-rata  $O(1)$ . Separate Chaining menangani kemungkinan error dengan menggunakan linked list.

## Guided 2

```

#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

```

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\USER\Documents\c++\hash table m5\guided>

```

### Deskripsi:

Program Hash Table dengan teknik Separate Chaining untuk menyimpan data karyawan (nama dan nomor telepon).

1. Menggunakan kelas HashNode untuk menyimpan data nama dan nomor telepon.

2. Kelas HashMap memiliki fungsi:

hashFunc (menghitung indeks hash dari nama)

insert (memasukkan data nama dan nomor telepon)

remove (menghapus data berdasarkan nama)

searchByName (mencari nomor telepon berdasarkan nama)

print (mencetak semua data dalam Hash Table)

3. Dalam main, program mendemonstrasikan operasi insert, searchByName, remove, dan print.

Intinya adalah, ini merupakan program struktur data Hash Table menggunakan Separate Chaining untuk menyimpan dan mengakses data karyawan secara efisien berdasarkan nama sebagai kunci.

E. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

```

#include <iostream>

#include <vector>
#include <string>
using namespace std;
const int TABLE_SIZE = 11;
string nim;
int nilai;
class HashNode
{
public:
    string nim;
    int nilai;
    HashNode(string nim, int nilai)
    {
        this->nim = nim;
        this->nilai = nilai;
    }
};
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 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(string 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;
            }
        }
    }
}
int search(string nim)
{
    int hash_val = hashFunc(nim);
    for (auto node : table[hash_val])
    {
        if (node->nim == nim)
        {
            return node->nilai;
        }
    }
    return -1; // return -1 if NIM is not
found
}
void findNimByScore(int minScore, int
maxScore)
{
    bool found = false;
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        for (auto pair : table[i])
        {
            if (pair != nullptr && pair->nilai
>= minScore && pair->nilai <= maxScore)
            {
                cout << pair->nim << "
memiliki nilai " << pair->nilai << endl;
                found = true;
            }
        }
    }
    if (!found)
    {
        cout << "Tidak ada mahasiswa yang
memiliki nilai antara " << minScore << " and " <<
maxScore << endl;
    }
}

```

```

    }
    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 data;
    string NIM;
    int nilai_mhs;
    while (true)
    {
        int menu;
        cout << "\nMenu" << endl;
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Mencari data berdasarkan NIM"
<< endl;
        cout << "4. Mencari data berdasarkan
nilai" << endl;
        cout << "5. Cetak data" << endl;
        cout << "6. Exit" << endl;
        cout << "Masukkan pilihan : ";
        cin >> menu;
        switch (menu)
        {
            case 1:
                cout << "Masukkan NIM : ";
                cin >> NIM;
                cout << "Masukkan nilai : ";
                cin >> nilai_mhs;
                data.insert(NIM, nilai_mhs);
                break;
            case 2:

```

```

        cout << "Masukkan NIM yang akan
dihapus : ";
        cin >> NIM;
        data.remove(NIM);
        cout << "Data berhasil dihapus" <<
endl;
        break;
    case 3:
        cout << "Masukkan NIM yang akan dicari
: ";
        cin >> NIM;
        cout << "NIM " << NIM << " memiliki
nilai " << data.search(NIM) << endl;
        break;
    case 4:
        int a, b;
        cout << "Masukkan rentang nilai
minimal : ";
        cin >> a;
        cout << "Masukkan rentang nilai
maksimal : ";
        cin >> b;
        data.findNimByScore(a, b);
        break;
    case 5:
        data.print();
        break;
    case 6:
        return 0;
    default:
        cout << "Menu tidak tersedia!" <<
endl;
        break;
    }
}
}

```

Screenshots Output

Menu

1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit

Masukkan pilihan : 1

Masukkan NIM : 2311102322

Masukkan nilai : 1

Menu

1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit

Masukkan pilihan : 3

Masukkan NIM yang akan dicari : 2311102322

NIM 2311102322 memiliki nilai 1

Menu

1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit

Masukkan pilihan : 4

Masukkan rentang nilai minimal : 0

Masukkan rentang nilai maksimal : 1

2311102322 memiliki nilai 1

## Menu

1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit

Masukkkan pilihan : 5

0:

1:

2: [2311102322, 1]

3:

4:

5:

6:

7:

8:

9:

10:

## Menu

1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit

Masukkkan pilihan : 2

Masukkan NIM yang akan dihapus : 2311102322

Data berhasil dihapus



```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkkan pilihan : 6
PS C:\Users\USER\Documents\c++\hash table m5\Unguided1> |
```

#### Deskripsi:

Program ini menggunakan Class HashNode sebagai penyimpanan data NIM dan nilai mahasiswa. Class hashMap memetakan Hash Table dengan metode:

- HashFunc merupakan fungsi hash untuk memetakan NIM ke indeks tabel.
- Insert untuk memasukkan data NIM dan nilai ke dalam Hash Table.
- Remove untuk menghapus data berdasarkan NIM.
- Search untuk mencari nilai berdasarkan NIM.
- FindNimByScore untuk mencari NIM berdasarkan rentang nilai.
- Print untuk mencetak semua data di dalam Hash Table.

Program utamanya menyediakan menu pilihan untuk:

- Menambah data NIM dan nilai.
- Menghapus data berdasarkan NIM.
- Mencari nilai berdasarkan NIM.
- Mencari NIM berdasarkan rentang nilai.

#### Referensi:

- LEARNING MANAGEMENT SYSTEM, Modul 5 tipe data praktikum algoritma struktur data IF 11 B
- WHATSAPP GROUP, PRAKTIKUM STRUKDAT IF 11 B, Modul 5 Hash Table praktikum algoritma struktur data IF 11 B