

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



Disusun Oleh :
DIO GILBRAN PRAMANA
NIM : 2311102062

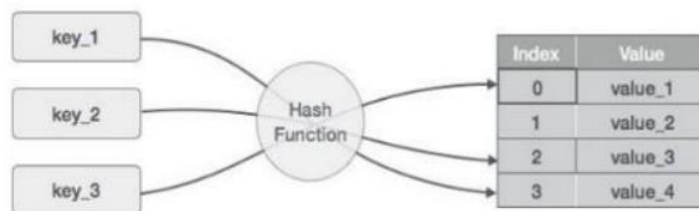
Dosen
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

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

A. 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.



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.

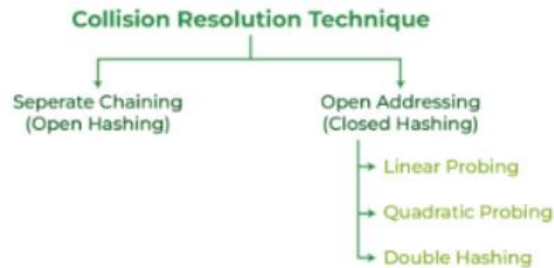
c. Operasi Hash Table

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.

d. 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 :



1. 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 di hash. 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.

2. Closed Hashing

- a) 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.
- b) Quadratic Probing Penanganannya hampir sama dengan metode linear, hanya lompatannya tidak satu-satu, tetapi quadratic (12, 22, 32, 42, ...)
- c) Double Hashing Pada saat terjadi collision, terdapat fungsi hash yang kedua untuk menentukan posisinya kembali.

B. GUIDED

GUIDED 1

```
#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;

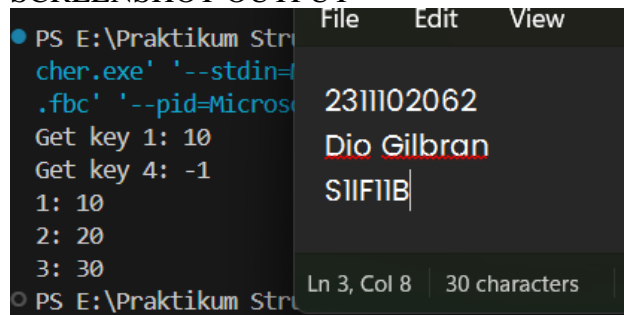
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

SCREENSHOT OUTPUT



```
PS E:\Praktikum Str... File Edit View
cher.exe' '--stdin=
.fbc' '--pid=Micros
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS E:\Praktikum Str...
```

DESKRIPSI

Di dalam main function, program mengilustrasikan penggunaan hash table dengan memasukkan beberapa entri (insertion), mencari nilai berdasarkan kunci (searching), menghapus elemen berdasarkan kunci (deletion), dan menampilkan semua elemen dalam tabel (traversal). Program ini memberikan pengertian praktis tentang penggunaan hash table dalam mengelola data dengan efisien menggunakan teknik hashing dan chaining untuk menangani collision.

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()
    {

```



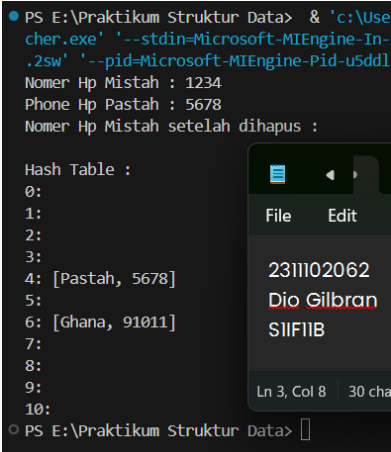
```

        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;
}

```

SCREENSHOT OUTPUT



```

PS E:\Praktikum Struktur Data> & 'c:\Use
cher.exe' '--stdin=Microsoft-MIEngine-In-
.2sw' '--pid=Microsoft-MIEngine-Pid-u5ddl
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:\Praktikum Struktur Data>

```

Context menu details:

- File Edit
- 2311102062
- Dio Gilbran
- SIIFIIB
- Ln 3, Col 8 30 cha

DESKRIPSI

Program yang diberikan adalah implementasi dari hash map menggunakan chaining untuk menangani collision. Hash map ini digunakan untuk menyimpan data pegawai berupa pasangan nama dan nomor telepon. Struktur data yang digunakan terdiri dari dua kelas utama: HashNode untuk merepresentasikan node dalam hash map, dan HashMap sebagai kelas yang mengatur operasi-operasi pada hash map.

C. UNGUIDED

UNGUIDED 1

```
#include <iostream>
#include <iomanip>
using namespace std;
const int MAX_SIZE = 10;

struct mahasiswa
{
    string nama;
    long long NIM;
    int nilai;
    mahasiswa *next;
    mahasiswa(string nama, long long NIM, int nilai) : nama(nama),
NIM(NIM), nilai(nilai), next(nullptr) {}
};

class HashTable
{
private:
    mahasiswa **table;
    int
    hash_func(long long key)
    {
        return key % MAX_SIZE;
    }
public:
    HashTable()
    {
        table = new mahasiswa *[MAX_SIZE]();
    }

    ~HashTable()
    {
        for (int i = 0; i < MAX_SIZE; ++i)
        {
            mahasiswa *current = table[i];
```

```

        while (current != nullptr)
        {
            mahasiswa *temp = current;
            current = current->next;
            delete temp;
        }
    }
    delete[] table;
}

void insert(string nama, long long NIM, int nilai)
{
    int index = hash_func(NIM);
    mahasiswa *new_mahasiswa = new mahasiswa(nama, NIM,
nilai);
    new_mahasiswa->next = table[index];
    table[index] = new_mahasiswa;
}

void remove(long long NIM)
{
    int index = hash_func(NIM);
    mahasiswa *current = table[index];
    mahasiswa *prev = nullptr;
    while (current != nullptr)
    {
        if (current->NIM == NIM)
        {
            if (prev == nullptr)
            {
                table[index] = current->next;
            }
            else
            {
                prev->next = current->next;
            }
            delete current;
            cout << "Mahasiswa dengan NIM " << NIM << " telah
dihapus." << endl;
            return;
        }
        prev = current;
        current = current->next;
    }
    cout << "Mahasiswa dengan NIM " << NIM << "
tidak ditemukan." << endl;
}

```

```

mahasiswa *cari_NIM(long long NIM)
{
    long long index = hash_func(NIM);
    mahasiswa *current = table[index];
    while (current != nullptr)
    {
        if (current->NIM == NIM)
        {
            return current;
        }
        current = current->next;
    }
    return nullptr;
}

void cari_nilai(int awalnya, int akhirnya)
{
    cout << "~~~~~" << endl;
    cout << "| Nama Mahasiswa | NIM | Nilai | " << endl;
    cout << "~~~~~" << endl;
    for (int i = 0; i < MAX_SIZE; ++i)
    {
        mahasiswa *current = table[i];
        while (current != nullptr)
        {
            if (current->nilai >= awalnya && current->nilai <=
akhirnya)
            {
                string namaSingkat = current->nama.substr(0,
30);
                cout << "| " << setw(30) << left <<
namaSingkat << " | ";
                cout << setw(20) << current->NIM << " | ";
                cout << setw(10) << fixed << setprecision(2)
<< current->nilai << " | " << endl;
            }
            current = current->next;
        }
    }
    cout << "~~~~~" << endl;
}

void traverse()
{
    cout << "~~~~~" << endl;

```

```

        cout << "| Nama Mahasiswa | NIM | Nilai | " << endl;
        cout << "~~~~~" << endl;
        for (int i = 0; i < MAX_SIZE; ++i)
        {
            mahasiswa *current = table[i];
            while (current != nullptr)
            {
                string namaSingkat = current->nama.substr(0, 30);
                cout << "| " << setw(30) << left << namaSingkat <<
" | ";

                cout << setw(20) << current->NIM << " | ";
                cout << setw(10) << current->nilai << " |" <<
endl;

                current = current->next;
            }
        }
        cout << "~~~~~" << endl;
    }
};

int main()
{
    HashTable ht;
    int pilih, nilai, awalnya, akhirnya;
    string nama;
    long long NIM;
    do
    {
        cout << endl;
        cout << " MENU: " << endl
        << endl;
        cout << "~~~~~" << endl;
        cout << "1. Tambah Data Mahasiswa" << endl;
        cout << "2. Hapus Data Mahasiswa" << endl;
        cout << "3. Cari berdasarkan NIM" << endl;
        cout << "4. Cari berdasarkan Rentang Nilai" << endl;
        cout << "5. Tampilkan Semua Data" << endl;
        cout << "6. Keluar" << endl;
        cout << "~~~~~" << endl
        << endl;
        cout << "Pilih menu : ";
        cin >> pilih;
        cout << endl;
        switch (pilih)
        {
            case 1:
                cout << "Masukan Nama: ";

```

```

        cin.ignore();
        getline(cin, nama);
        cout << "Masukan NIM: ";
        cin >> NIM;
        cout << "Masukan Nilai: ";
        cin >> nilai;
        ht.insert(nama, NIM, nilai);
        cout << "Data berhasil ditambahkan" << endl;
        cout << "~~~~~" << endl
            << endl;
        break;
    case 2:
        cout << "Masukan NIM yang ingin dihapus: ";
        cin >> NIM;
        ht.remove(NIM);
        cout << "~~~~~" << endl
            << endl;
        break;
    case 3:
        cout << "Masukan NIM yang ingin dicari : ";
        cin >> NIM;
        {
            mahasiswa *mahasiswa_ptr = ht.cari_NIM(NIM);
            if (mahasiswa_ptr)
            {
                cout << "Ditemukan mahasiswa dengan NIM " <<
NIM << " bernama " << mahasiswa_ptr->nama << " dan memiliki nilai"
<< mahasiswa_ptr->nilai << endl;
            }
            else
            {
                cout << "mahasiswa dengan NIM " << NIM << "
tidak ditemukan " << endl;
            }
        }
        cout << "~~~~~" << endl
            << endl;
        break;
    case 4:
        cout << "masukan nilai awal: ";
        cin >> awalnya;
        cout << "masukan nilai akhir: ";
        cin >> akhirnya;
        ht.cari_nilai(awalnya, akhirnya);
        break;
    case 5:
        ht.traverse();

```

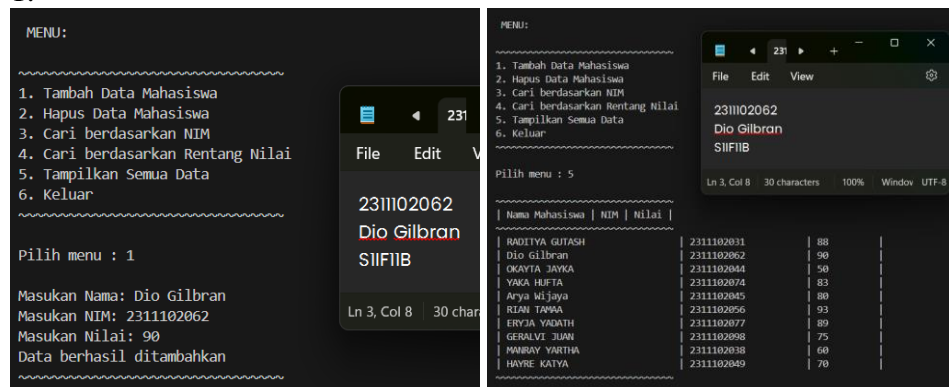
```

        break;
    case 6:
        cout << "Sampai jumpa" << endl;
        cout << "~~~~~" << endl
            << endl;
        break;
    default:
        cout << "Pilihan anda tidak ada" << endl;
        cout << "~~~~~" << endl
            << endl;
    }
} while (pilih != 6);
return 0;
}

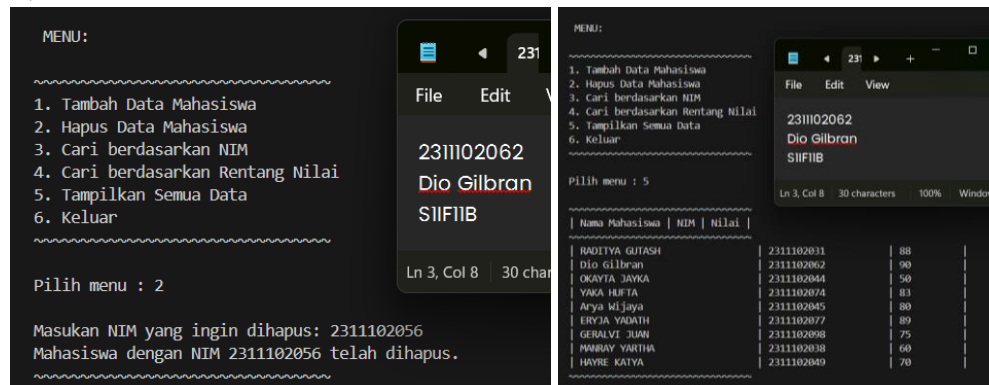
```

SCREENSHOT OUTPUT

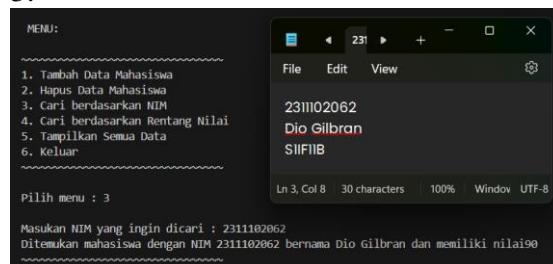
1.



2.



3.



4.

```
MENU:
~~~~~
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar
~~~~~

Pilih menu : 4

masukan nilai awal: 80
masukan nilai akhir: 90
~~~~~
| Nama Mahasiswa | NIM | Nilai |
~~~~~
| RADITYA GUTASH | 2311102031 | 88 |
| Dio Gilbran    | 2311102062 | 90 |
| YAKA HUFTA     | 2311102074 | 83 |
| Arya Wijaya    | 2311102045 | 80 |
| ERYJA YADATH   | 2311102077 | 89 |
~~~~~
```

5.

```
MENU:
~~~~~
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari berdasarkan NIM
4. Cari berdasarkan Rentang Nilai
5. Tampilkan Semua Data
6. Keluar
~~~~~

Pilih menu : 6

Sampai jumpa
~~~~~
```

DESKRIPSI

Program ini mengimplementasikan tabel hash untuk menyimpan, mencari, dan mengelola data mahasiswa dengan atribut nama, NIM, dan nilai. Program menggunakan struktur data linked list untuk menangani tabrakan hash, memastikan setiap entri dengan hash yang sama dapat disimpan dalam tabel. Program ini menyediakan menu interaktif untuk menambah data mahasiswa, menghapus data berdasarkan NIM, mencari mahasiswa berdasarkan NIM atau rentang nilai, dan menampilkan semua data mahasiswa yang tersimpan. Pengguna dapat berinteraksi dengan program melalui antarmuka konsol yang mudah digunakan.

D. KESIMPULAN

Kesimpulan dari praktikum ini adalah bahwa penggunaan hash table dengan teknik chaining efektif dalam mengorganisir dan mengelola data dengan efisien. Praktikum ini mengilustrasikan berbagai operasi dasar pada hash table seperti penyisipan, penghapusan, pencarian, dan traversal, serta bagaimana menangani collision menggunakan linked list. Implementasi hash table yang diuji memberikan wawasan praktis mengenai penerapan konsep hashing dalam struktur data untuk meningkatkan efisiensi pencarian dan pengelolaan data.

E. REFERENSI

Asisten Praktikum. 08 Mei 2024. "Modul 5 Hash Table". 2024, dari Learning Management System. 2024

Aziz, N., & Arief, M. (2020). Implementasi Hash Table Menggunakan C++. <https://www.belajarstatistik.com/blog/2022/03/08/implementasi-hash-dalam-bahasa-c/>