

LAPORAN PRAKTIKUM

MODUL V HASH TABLE



Disusun oleh:
Muhammad Rifki Fadhilah
NIM: 2311102032

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng.

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

BAB V

TUJUAN PRAKTIKUM

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

BAB II

DASAR TEORI

Struktur data Hash Table adalah cara untuk menyimpan dan mengelola data dengan efisien dan cepat. Ini bekerja dengan menggunakan kunci unik untuk setiap data, yang digunakan untuk mengakses atau mengubah data tersebut. Hash Table mengatur data dalam bentuk pasangan kunci-nilai, menggunakan fungsi hash untuk mengonversi kunci menjadi indeks dalam array. Hal ini memungkinkan akses yang cepat ke data, karena indeks dihitung secara langsung. Hash Table cocok digunakan untuk pencarian, penyisipan, penghapusan, dan pembaruan data dalam waktu konstan, selama tidak ada konflik dalam fungsi hash.

Manfaat dari Struktur Data Hash Table antara lain:

1. **Pencarian Cepat:** Memungkinkan pencarian data dengan cepat berdasarkan kunci, berguna dalam basis data, kamus, dan cache.
2. **Penyimpanan Efisien:** Hash Table dapat menyimpan data secara efisien, dengan pengambilan dan penyisipan data dalam waktu konstan, asalkan tidak ada konflik yang signifikan.
3. **Implementasi Struktur Data Lain:** Hash Table dapat digunakan untuk mengimplementasikan struktur data lain, seperti set dan map.

Teknik Hash Table mencakup cara atau strategi dalam mengimplementasikan fungsi hash dan menangani konflik hash. Beberapa teknik yang umum digunakan antara lain:

1. **Chaining:** Menggunakan struktur data seperti linked list atau array di setiap slot tabel hash. Ketika terjadi konflik, data baru disisipkan dalam linked list atau array yang terkait dengan slot tersebut. Chaining memungkinkan penyimpanan beberapa nilai dengan indeks yang sama, dan merupakan metode yang umum dan fleksibel.
2. **Closed Hashing:**

- a. Linear Probing: Mencari posisi yang kosong di bawah tempat terjadinya konflik. Jika masih penuh, terus mencari ke bawah hingga menemukan tempat kosong. Jika tidak ada tempat yang kosong, Hash Table dianggap penuh.
- b. Quadratic Probing: Mirip dengan linear probing, namun menggunakan lompatan quadratic (12, 22, 32, 42, ...) untuk mencari posisi kosong.
- c. Double Hashing: Menggunakan fungsi hash kedua untuk menentukan posisi kembali saat terjadi konflik.

BAB IV

GUIDED

1. 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;

    }

}

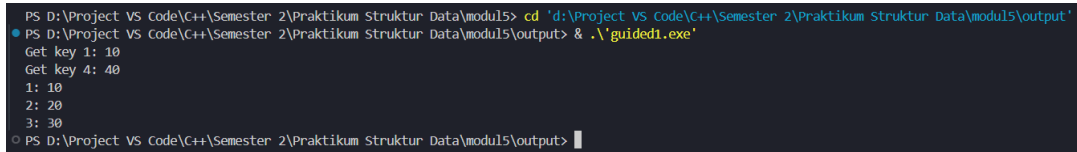
// Traversal
void traverse()
{
    for (int i = 0; i < MAX_SIZE; i++)
    {
        Node *current = table[i];
        while (current != nullptr)
        {
            cout << current->key << ": " << current->value<<
endl;

            current = current->next;
        }
    }
};

int main()
{
    HashTable ht;
    // Insertion
    ht.insert(1, 10);
    ht.insert(2, 20);
    ht.insert(3, 30);
    ht.insert(4, 40);
    // 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;
}

```


Screenshoot program



```
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul5> cd 'd:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul5\output'
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul5\output> & .\'guided1.exe\'
Get key 1: 10
Get key 4: 40
1: 10
2: 20
3: 30
PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul5\output> |
```

Deskripsi program

Program diatas adalah implementasi dari hash table menggunakan chaining untuk menangani tabrakan hash. Dalam program ini, hash table dibangun menggunakan array of linked list. Setiap elemen array merupakan pointer ke node pertama dari linked list di indeks tersebut. Jika terjadi tabrakan hash (hash collision), yaitu dua key yang berbeda memiliki hash yang sama, maka solusinya adalah dengan chaining, yaitu menyimpan data dengan hash yang sama pada linked list yang sama.

Fungsi `hash_func` digunakan untuk menghasilkan indeks array berdasarkan key. `insert` digunakan untuk menyisipkan pasangan key-value ke dalam hash table. `get` digunakan untuk mencari nilai value berdasarkan key. `remove` digunakan untuk menghapus pasangan key-value dari hash table. `traverse` digunakan untuk menampilkan seluruh pasangan key-value dalam hash table. Program ini mengimplementasikan operasi dasar dari hash table, yaitu penyisipan, pencarian, dan penghapusan data.

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;
}

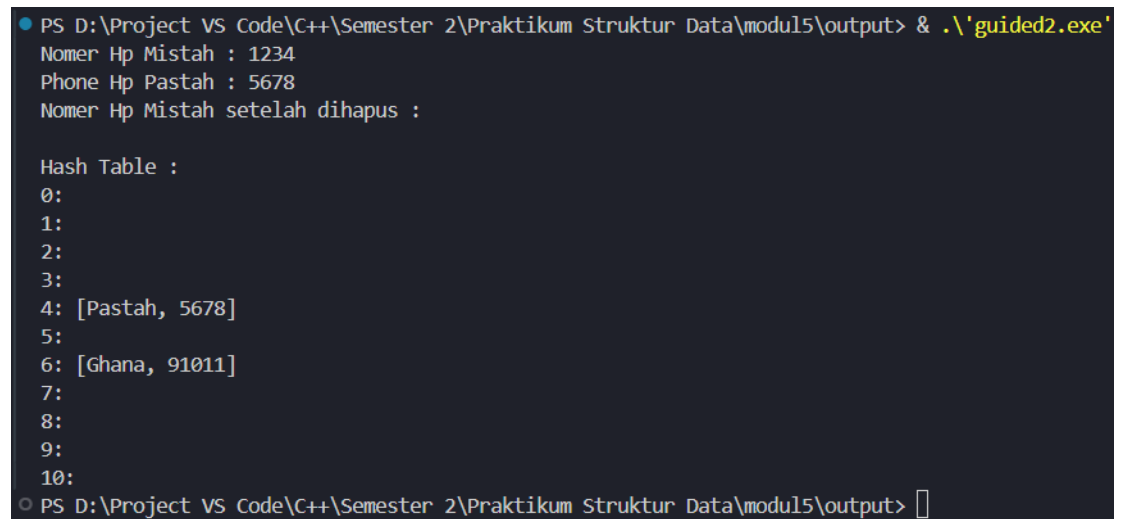
```

```

        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



```

PS D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul5\output> & .\'guided2.exe'
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 D:\Project VS Code\C++\Semester 2\Praktikum Struktur Data\modul5\output>

```

Deskripsi program

Program ini adalah implementasi sederhana dari struktur data hash table dalam bahasa C++. Hash table adalah struktur data yang digunakan untuk menyimpan data dengan cara yang efisien dan memungkinkan pencarian cepat berdasarkan kunci. Dalam program ini, hash table digunakan untuk menyimpan informasi nama karyawan beserta nomor telepon mereka.

Program ini terdiri dari dua kelas utama, yaitu HashNode dan HashMap. Kelas HashNode merepresentasikan node dalam hash table, yang menyimpan nama dan nomor telepon karyawan. Kelas HashMap digunakan untuk mengelola hash table, dengan menyediakan fungsi-fungsi seperti insert untuk menambahkan data, remove untuk menghapus data, searchByName untuk mencari nomor telepon berdasarkan nama, dan print untuk mencetak isi hash table.

Dalam main function, program membuat sebuah instance dari HashMap yang digunakan untuk menyimpan data karyawan. Program kemudian menambahkan beberapa data karyawan ke dalam hash table, mencari dan menghapus data karyawan, serta mencetak isi hash table ke layar.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

```
#include <iostream>
#include <vector>
#include <list>

using namespace std;

// Struktur data untuk mahasiswa
struct Mahasiswa {
    int nim;
    int nilai;
};

// Ukuran hash table
const int TABLE_SIZE = 11;

// Class untuk hash table
class HashTable {
private:
    vector<list<Mahasiswa>> table;

    // Fungsi hash sederhana
    int hashFunction(int nim) {
        return nim % TABLE_SIZE;
    }

public:
    // Konstruktor
    HashTable() {
        table.resize(TABLE_SIZE);
    }
};
```

```

    }

    // Menambahkan data mahasiswa baru
    void tambahData(int nim, int nilai) {
        Mahasiswa mhs;
        mhs.nim = nim;
        mhs.nilai = nilai;

        int index = hashFunction(nim);
        table[index].push_back(mhs);
    }

    // Menghapus data mahasiswa berdasarkan NIM
    void hapusData(int nim) {
        int index = hashFunction(nim);
        for (auto it = table[index].begin(); it !=
table[index].end(); ++it) {
            if (it->nim == nim) {
                table[index].erase(it);
                break;
            }
        }
    }

    // Mencari data mahasiswa berdasarkan NIM
    void cariByNIM(int nim) {
        int index = hashFunction(nim);
        for (auto it = table[index].begin(); it !=
table[index].end(); ++it) {
            if (it->nim == nim) {
                cout << "Data ditemukan: NIM " << it->nim << ",
Nilai " << it->nilai << endl;
                return;
            }
        }
    }

```



```

    }

    cout << "Data tidak ditemukan" << endl;

}

// Mencari data mahasiswa berdasarkan rentang nilai (80 - 90)
void cariByRange() {
    for (int i = 0; i < TABLE_SIZE; ++i) {
        for (auto it = table[i].begin(); it !=
table[i].end(); ++it) {
            if (it->nilai >= 80 && it->nilai <= 90) {
                cout << "NIM " << it->nim << ", Nilai " <<
it->nilai << endl;
            }
        }
    }
}

};

int main() {
    HashTable hashTable;
    int choice, nim, nilai;

    do {
        cout << "\nMenu:" << endl;
        cout << "1. Tambah Data Mahasiswa" << endl;
        cout << "2. Hapus Data Mahasiswa" << endl;
        cout << "3. Cari Data Mahasiswa berdasarkan NIM" << endl;
        cout << "4. Cari Data Mahasiswa berdasarkan Rentang Nilai
(80 - 90)" << endl;
        cout << "5. Keluar" << endl;
        cout << "Pilih: ";
        cin >> choice;

        switch(choice) {

```

```

        case 1:
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan Nilai: ";
            cin >> nilai;
            hashTable.tambahData(nim, nilai);
            break;
        case 2:
            cout << "Masukkan NIM yang akan dihapus: ";
            cin >> nim;
            hashTable.hapusData(nim);
            break;
        case 3:
            cout << "Masukkan NIM yang akan dicari: ";
            cin >> nim;
            hashTable.cariByNIM(nim);
            break;
        case 4:
            hashTable.cariByRange();
            break;
        case 5:
            cout << "Keluar dari program." << endl;
            break;
        default:
            cout << "Pilihan tidak valid!" << endl;
    }
    } while(choice != 5);

    return 0;
}

```

Screenshoot program

Menu

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

Tambah Data Mahasiswa

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Pilih: 1
Masukkan NIM: 231110201
Masukkan Nilai: 85

Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Pilih: 1
Masukkan NIM: 231110202
Masukkan Nilai: 95

Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Pilih: 1
Masukkan NIM: 231110203
Masukkan Nilai: 90
```

Mencari Data Mahasiswa berdasarkan NIM

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Pilih: 3
Masukkan NIM yang akan dicari: 231110203
Data ditemukan: NIM 231110203, Nilai 90

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

Hapus Data Mahasiswa

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Pilih: 2
Masukkan NIM yang akan dihapus: 231110203
```

Mencari Data Mahasiswa berdasarkan Rentang Nilai (80-90)

```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Pilih: 4
NIM 231110201, Nilai 85
```

Deskripsi program

Program ini adalah implementasi sederhana dari hash table dalam bahasa C++. Hash table adalah struktur data yang digunakan untuk menyimpan pasangan kunci-nilai di mana kunci unik digunakan untuk mengakses nilai yang terkait.

Pada program ini, kita memiliki struktur data Mahasiswa yang memiliki dua anggota: nim (Nomor Induk Mahasiswa) dan nilai (nilai mahasiswa). Kemudian, ada kelas HashTable yang memiliki hash table sebagai anggota. Ukuran hash table ditentukan oleh konstanta TABLE_SIZE.

Fungsi hashFunction digunakan untuk menghitung indeks hash dari sebuah nilai NIM. Ini dilakukan dengan menggunakan sisa pembagian (modulo) dari NIM dengan TABLE_SIZE.

Metode tambahData digunakan untuk menambahkan data mahasiswa baru ke hash table. Data tersebut di-hash menggunakan fungsi hash, dan kemudian dimasukkan ke dalam list pada indeks yang dihasilkan.

Metode hapusData digunakan untuk menghapus data mahasiswa dari hash table berdasarkan NIM. Ini melakukan pencarian di indeks yang dihasilkan oleh fungsi hash dan menghapus entri dengan NIM yang sesuai.

Metode cariByNIM digunakan untuk mencari data mahasiswa berdasarkan NIM. Ini melakukan pencarian di indeks yang dihasilkan oleh fungsi hash dan mencetak data mahasiswa jika ditemukan.

Metode `cariByRange` digunakan untuk mencari data mahasiswa berdasarkan rentang nilai antara 80 hingga 90. Ini melakukan iterasi melalui seluruh hash table dan mencetak data mahasiswa yang memenuhi kriteria.

Di dalam main, program memberikan pilihan menu kepada pengguna untuk menambah data mahasiswa, menghapus data mahasiswa, mencari data mahasiswa berdasarkan NIM, mencari data mahasiswa berdasarkan rentang nilai, atau keluar dari program. Kemudian, sesuai pilihan pengguna, program akan memanggil metode yang sesuai dari kelas `HashTable`. Program berjalan hingga pengguna memilih untuk keluar.

BAB IV

KESIMPULAN

Struktur data Hash Table adalah metode efisien dan cepat untuk menyimpan dan mengelola data dengan menggunakan kunci unik untuk setiap elemen. Hash Table bekerja dengan prinsip kunci-nilai dan menggunakan fungsi hash untuk mengonversi kunci menjadi indeks dalam array, sehingga memungkinkan akses data dengan cepat. Manfaat Hash Table antara lain pencarian cepat, penyimpanan efisien, dan kemampuan untuk mengimplementasikan struktur data lain seperti set dan map. Teknik Hash Table meliputi chaining untuk penanganan konflik dan closed hashing dengan metode linear probing, quadratic probing, dan double hashing.

BAB V

DAFTAR PUSTAKA

1. Asisten Praktikum. (2024). Modul V : Hash Table
2. Annisa. (2023). Struktur Data Hash Table: Pengertian, Cara Kerja, dan Operasi Hash Table. Diakses pada 14 Mei 2024, dari <https://fikti.umsu.ac.id/struktur-data-hash-table-pengertian-cara-kerja-dan-operasihash-table/>
3. Programiz. (n.d.). Hash Table. Diakses pada 14 Mei 2024, dari <https://www.programiz.com/dsa/hash-table>