

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



Disusun Oleh :
MARSHELY AYU ISWANTO
2311102073

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

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

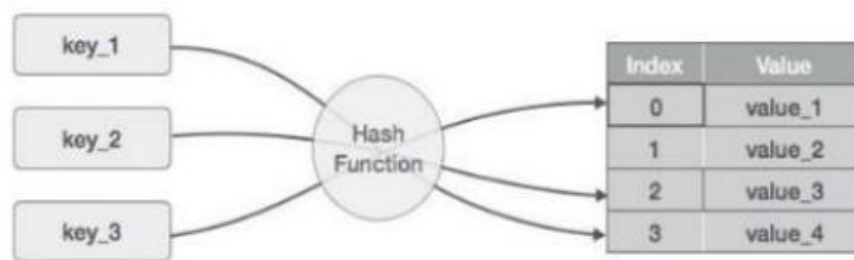
A. Dasar Teori

a. Pengertian Hash Table

Hash Table adalah struktur data yang memungkinkan untuk menyimpan pasangan kunci-nilai, dimana setiap kunci-nilai tersebut dikelompokkan ke sebuah nilai menggunakan fungsi hash. Table hash digunakan dalam pemrograman untuk operasi dasar seperti penambahan, penghapusan, dan pencarian. Untuk Cara kerja hash table melibatkan penggunaan fungsi hash untuk mengubah kunci menjadi indeks dalam tabel array. Hash table biasanya terdiri dari dua komponen utama: array (atau vektor) dan fungsi hash.

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.

Hash table bekerja dengan menggunakan fungsi hash untuk merubah kunci menjadi indeks dalam array, penyimpanan dan pencarian data yang efisien dengan kompleksitas waktu yang konstan, asalkan fungsi hash nya baik. 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 table yang efektif sangat penting dalam memastikan kinerja yang optimal dari hash table, terutama dalam hal efisiensi pencarian dan pengelolaan data. 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

Insertion dalam hash table merupakan proses memasukkan elemen data baru ke dalam struktur hash table. Elemen data baru yang akan disisipkan diolah menggunakan hash table. Memanggil fungsi hash untuk menentukan posisi bucket yang tepat, dan kemudian menambahkan data ke bucket tersebut.

2. Deletion

Deletion dalam hash table merupakan proses penghapusan elemen data dari struktur hash table. elemen data yang akan dihapus diolah menggunakan fungsi hash untuk menentukan indeks di mana elemen data tersebut disimpan, kemudian menghapusnya dari bucket yang sesuai.

3. Searching

Searching (Pencarian) dalam hash table merupakan proses menemukan elemen data dengan menggunakan kunci tertentu dalam struktur hash table. 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

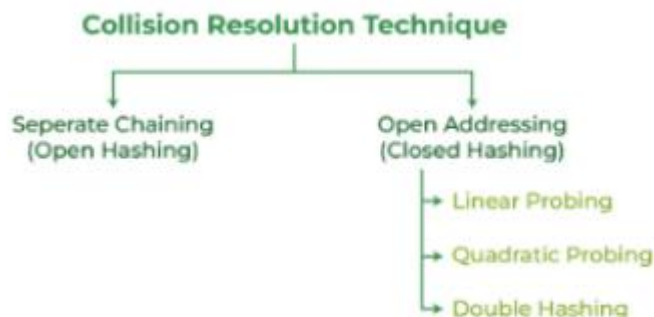
Update (Pembaruan) dalam hash table merupakan proses mengubah nilai elemen data yang sudah ada dalam struktur hash table. Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

5. Traversal

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

d. Collision Resolution

Collision resolution merupakan proses menangani situasi di mana dua atau lebih kunci menghasilkan nilai hash yang sama atau indeks yang sama dalam hash table. 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

- **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.

- **Quadratic Probing**

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

- **Double hashing**

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

B. 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);

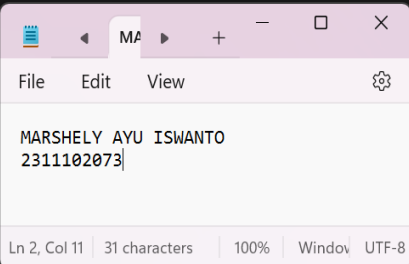
    // Traversal
    ht.traverse();

    return 0;
}

```

Screenshots Output

```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ guided1_modul5.cpp -o guided1_modul5 } ; if ($?) { .\guided1_modul5 }
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS E:\TugasAlproSmt2_6Mar2024>
```



Deskripsi Program:

Program tersebut merupakan program c++ yaitu penerapan sederhana dari hash table untuk penyimpanan dan pencarian data berdasarkan kunci. Input dari kode program tersebut adalah operasi-operasi yang ingin dilakukan pada hash table, yaitu operasi insert, get, dan remove. Input tersebut diwakili oleh pemanggilan fungsi-fungsi yang sesuai pada objek HashTable yang dibuat di dalam fungsi main(). Output dari kode program tersebut adalah untuk operasi pertama yaitu mencari nilai dengan kunci 1 yang hasilnya adalah 10, Lalu untuk operasi kedua yaitu mencari nilai dengan kunci 4, namun kunci 4 tidak ada dalam tabel hash jadi hasil pencariannya adalah -1 karena jika kunci yang dicari tidak ditemukan, nilai yang dikembalikan adalah -1, selanjutnya setelah operasi insert nilai 10,20,30 dimasukkan ke dalam hash tabel kemudian hasil tranversal mencetak semua pasangan kunci-nilai yang tersimpan dalam tabel hash yaitu pasangan (1, 10), (2, 20), dan (3, 30).

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;
}

```

Screenshoots Output

The screenshot shows a terminal window on the left and a separate application window on the right.

Terminal Output:

```

PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
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:\TugasAlproSmt2_6Mar2024>

```

Application Window:

The application window displays a hash table entry for the name "MARSHELY AYU ISWANTO" with the phone number "2311102073". The window title is "MA" and it shows standard file editing controls.

Deskripsi Program

Program diatas merupakan program c++ yaitu penerapan dari hash tabel dengan chaining dan penambahan, pencarian, serta penghapusan. jika dua kunci pada program tersebut menghasilkan indeks yang sama dalam tabel hash, entri baru akan ditambahkan ke daftar yang sudah ada di indeks tersebut. Input dari kode program tersebut adalah operasi-operasi yang ingin dilakukan pada hash table, yaitu operasi insert, searchByName, dan remove. Input tersebut diwakili oleh pemanggilan fungsi-fungsi yang sesuai pada objek HashTable yang dibuat di dalam fungsi main(), membuat tabel hash dengan tiga entri: "Mistah" dengan nomor telepon "1234", "Pastah" dengan nomor telepon "5678", dan "Ghana" dengan nomor telepon "91011". Lalu, program mencari nomor telepon dari "Mistah" dan "Pastah", menghapus entri dari "Mistah", dan mencetak entri yang tersisa di tabel hash. Output yang dicetak tersebut menunjukkan bahwa setelah operasi-operasi ditambahkan, pencarian berhasil menemukan nomor telepon yang sesuai, penghapusan berhasil menghapus entri yang sesuai, dan tabel hash dicetak untuk menunjukkan status setiap slot di dalamnya

C. Unguided

Unguided 1

1. Implementasikan hash table untuk menyimpan data mahasiswa. Setiap mahasiswa memiliki NIM dan nilai. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan nilai. Dengan ketentuan :
 - a. Setiap mahasiswa memiliki NIM dan nilai.
 - b. Program memiliki tampilan pilihan menu berisi poin C.
 - c. Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan rentang nilai (80 – 90).

Source Code

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

using namespace std;

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

// Kelas hash table
class HashTable {
```

```

private:
    static const int hash_size = 10; // Ukuran tabel hash
    vector<list<Mahasiswa>> table; // Tabel hash

    // Fungsi hash untuk menghitung indeks dalam tabel
hash
    int hashFunction(int NIM) {
        return NIM % hash_size;
    }

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

    // Fungsi untuk 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);
    }

    // Fungsi untuk 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;
            }
        }
    }

    // Fungsi untuk mencari data mahasiswa berdasarkan NIM
    void cariByNIM(int NIM) {
        int index = hashFunction(NIM);
        for (const auto &mhs : table[index]) {
            if (mhs.NIM == NIM) {
                cout << "Data ditemukan: NIM = " <<
mhs.NIM << ", Nilai = " << mhs.nilai << endl;
                return;
            }
        }
        cout << "Data tidak ditemukan" << endl;
    }

```

```

        // Fungsi untuk mencari data mahasiswa berdasarkan
        rentang nilai (80 - 90)
        void cariByRange() {
            for (int i = 0; i < hash_size; ++i) {
                for (const auto &mhs : table[i]) {
                    if (mhs.nilai >= 80 && mhs.nilai <= 90) {
                        cout << "NIM = " << mhs.NIM << ",
Nilai = " << mhs.nilai << endl;
                    }
                }
            }
        }

        // Fungsi untuk menampilkan seluruh data mahasiswa
        void tampilkanSemuaData() {
            for (int i = 0; i < hash_size; ++i) {
                for (const auto &mhs : table[i]) {
                    cout << "NIM = " << mhs.NIM << ", Nilai =
" << mhs.nilai << endl;
                }
            }
        };

int main() {
    HashTable hashTable;

    // Menampilkan menu
    cout << "Menu:" << 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. Tampilkan semua data mahasiswa" << endl;
    cout << "0. Keluar" << endl;

    int choice;
    do {
        cout << "Masukkan pilihan Anda: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                int NIM, nilai;
                cout << "Masukkan NIM: ";
                cin >> NIM;
                cout << "Masukkan nilai: ";
                cin >> nilai;
                hashTable.tambahData(NIM, nilai);
                break;
            }
        }
    } while (choice != 0);
}

```

```

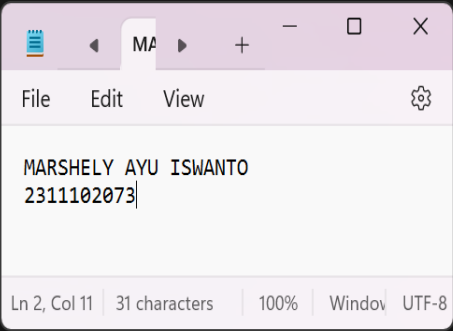
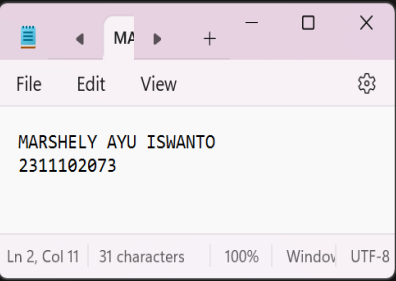
        }
        case 2: {
            int NIM;
            cout << "Masukkan NIM yang ingin dihapus:
";
            cin >> NIM;
            hashTable.hapusData(NIM);
            break;
        }
        case 3: {
            int NIM;
            cout << "Masukkan NIM yang ingin dicari:
";
            cin >> NIM;
            hashTable.cariByNIM(NIM);
            break;
        }
        case 4: {
            cout << "Mahasiswa dengan nilai antara 80
dan 90:" << endl;
            hashTable.cariByRange();
            break;
        }
        case 5: {
            cout << "Semua data mahasiswa:" << endl;
            hashTable.tampilkanSemuaData();
            break;
        }
        case 0:
            cout << "Program selesai." << endl;
            break;
        default:
            cout << "Pilihan tidak valid." << endl;
    }
} while (choice != 0);

return 0;
}

```

Screenshots Output

```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ unguided1_modul5.cpp -o unguided1_modul5 } ; if ($?) { .\unguided1_modul5 }
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. Tampilkan semua data mahasiswa
0. Keluar
Masukkan pilihan Anda: 1
Masukkan NIM: 073
Masukkan nilai: 90
Masukkan pilihan Anda: 1
Masukkan NIM: 074
Masukkan nilai: 80
Masukkan pilihan Anda: 1
Masukkan NIM: 075
Masukkan nilai: 75
Masukkan pilihan Anda: 1
Masukkan NIM: 076
Masukkan nilai: 87
Masukkan pilihan Anda: 5
Semua data mahasiswa:
NIM = 73, Nilai = 90
NIM = 74, Nilai = 80
NIM = 75, Nilai = 75
NIM = 76, Nilai = 87
Masukkan pilihan Anda: 2
Masukkan NIM yang ingin dihapus: 073
Masukkan pilihan Anda: 5
Semua data mahasiswa:
NIM = 74, Nilai = 80
NIM = 75, Nilai = 75
NIM = 76, Nilai = 87
Masukkan pilihan Anda: 3
Masukkan NIM yang ingin dicari: 075
Data ditemukan: NIM = 75, Nilai = 75
Masukkan pilihan Anda: 4
Mahasiswa dengan nilai antara 80 dan 90:
NIM = 74, Nilai = 80
NIM = 76, Nilai = 87
Masukkan pilihan Anda: 5
Semua data mahasiswa:
NIM = 74, Nilai = 80
NIM = 75, Nilai = 75
NIM = 76, Nilai = 87
Masukkan pilihan Anda: 0
Program selesai.
PS E:\TugasAlproSmt2_6Mar2024>
* History restored
PS E:\TugasAlproSmt2_6Mar2024>
```



Deskripsi Program :

Program diatas merupakan program c++ yaitu program dari struktur data hash tabel dalam bahasa c++. Program ini untuk melakukan beberapa operasi seperti menambah, menghapus, mencari berdasarkan NIM, mencari berdasarkan rentang nilai dari 80-90, dan menampilkan seluruh data mahasiswa yang telah dimasukkan. Input dari kode program tersebut adalah pilihan operasi yang disediakan, untuk output dari kode program tersebut adalah tergantung pada pilihan yang dipilih oleh pengguna lalu menampilkan data mahasiswa.

Kesimpulan

Hash tabel merupakan struktur data yang efisien untuk menyimpan dan mengakses data. Operasi dasar seperti penambahan, penghapusan, dan pencarian data memiliki kompleksitas waktu rata-rata $O(1)$.

D. Referensi

Bhojasia, M. (n.d.). *C++ Program to Implement Hash Table*. From www.sanfoundry.com: <https://www.sanfoundry.com/cpp-program-implement-hash-tables/>

Harendra. (2024, Mei 1). *hashing-data-structure*. From www.geeksforgeeks.org: <https://www.geeksforgeeks.org/hashing-data-structure/>

Hash Table. (n.d.). From www.programiz.com: <https://www.programiz.com/dsa/hash-table>