LAPORAN PRAKTIKUM

MODUL V
HASH TABLE

Disusun oleh:
FATTAH RIZQY ADHIPRATAMA
NIM: 2311102019

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

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

BAB I

TUJUAN PRAKTIKUM

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

BAB II
DASAR TEORI

Hash Table (Tabel Hash) adalah struktur data yang digunakan untuk menyimpan dan mengelola kumpulan data, di mana setiap elemen dalam kumpulan data memiliki kunci (key) yang unik dan nilainya (value) terkait. Konsep utama di balik tabel hash adalah penggunaan fungsi hash untuk mengonversi kunci menjadi alamat atau indeks di dalam tabel, sehingga memungkinkan pencarian dan pengambilan data dengan efisien.

Pada dasarnya, tabel hash berfungsi sebagai penyimpanan asosiatif, di mana data disimpan dalam bentuk pasangan kunci-nilai. Proses ini melibatkan penggunaan fungsi hash untuk menghasilkan indeks unik dari kunci, dan nilai yang sesuai akan disimpan pada indeks tersebut.

Salah satu keuntungan utama dari tabel hash adalah kecepatan akses dan pencarian data yang sangat efisien. Ketika kita ingin mencari nilai berdasarkan kunci, kita hanya perlu melakukan proses hashing pada kunci untuk mengetahui lokasi penyimpanan datanya, sehingga tidak perlu mencari secara berurutan seperti pada struktur data lainnya seperti array atau list.

Namun, perlu diingat bahwa pada situasi tertentu, bisa saja terjadi tabrakan hash (hash collision), yaitu ketika dua kunci berbeda menghasilkan indeks yang sama. Untuk menangani tabrakan ini, metode penyelesaian tabrakan (collision resolution) digunakan, seperti linear probing, chaining, atau double hashing. Hash table digunakan dalam berbagai aplikasi, termasuk dalam basis data, kamus (dictionary), dan kebanyakan algoritma yang membutuhkan pencarian atau pengelompokan data dengan cepat dan efisien.

Fungsi utamanya pada data adalah mempercepat proses akses data. Hal ini berkaitan dengan peningkatan data dalam jumlah besar yang diproses oleh jaringan

data global dan lokal. Hash table adalah solusi untuk membuat proses akses data lebih cepat dan memastikan bahwa data dapat dipertukarkan dengan aman.

Di dalam banyak bidang, hash table dikembangkan dan digunakan karena menawarkan kelebihan dalam efisiensi waktu operasi, mulai dari pengarsipan hingga pencarian data. Contohnya adalah bidang jaringan komputer yang mengembangkannya menjadi hybrid open hash table, yang kemudian dipakai untuk memproses jaringan komputer.

Untuk membuat hash table, sepotong memori perlu diblokir dengan cara yang sama seperti saat membuat array. Anda perlu membuat indeks yang didasarkan pada kunci dengan menggunakan fungsi hash karena indeks yang dihasilkan harus sesuai dengan potongan memori.

Ada dua pemeriksaan yang dibutuhkan saat menempatkan data baru pada hash table, yaitu nilai hash dari kunci dan bagaimana nilainya dibandingkan dengan objek lain. Pemeriksaan ini diperlukan saat membuatnya dengan Python karena saat data dimasukkan, kunci akan di-hash dan di-mask agar diubah menjadi larik atau indeks yang efisien.

BAB III

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);

    // 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:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 5>
me\Modul 5\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunner
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 5>
```

Deskripsi program

Program ini merupakan implementasi dari hash table. Hash table adalah struktur data yang digunakan untuk menyimpan key-value pairs di mana setiap nilai diakses dengan menggunakan kuncinya. Pada program tersebut terdapat beberapa komponen utama yaitu seperti Fungsi Hash “hash_func”, Struktur Node “node”, Kelas Hash Table “HashTable”, dan Fungsi utama “main”.

2. Guided 2

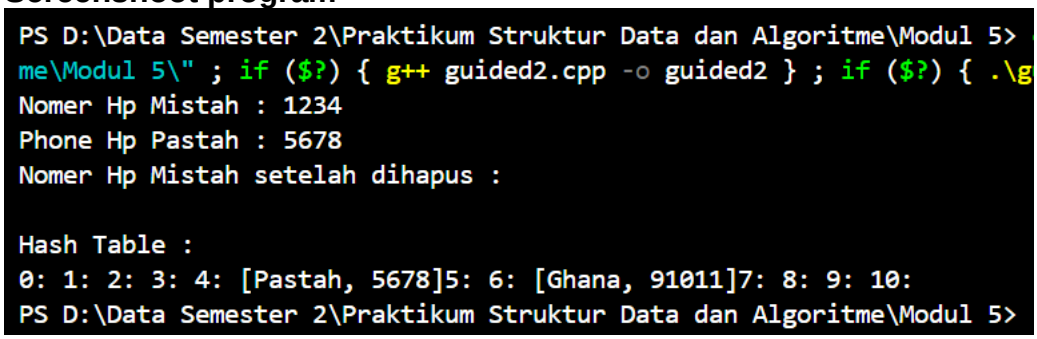
Source Code

```
#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 ini merupakan implementasi dari struktur data Hash Table. Hash table digunakan untuk menyimpan dan mencari kontak dengan efisien berdasarkan nama. Hash table ini memungkinkan pengguna untuk menambahkan, mencari, dan menghapus informasi tentang pegawai.

LATIHAN KELAS - UNGUIDED

1. **Unguided 1**

   Source code

```cpp
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

// Struktur data untuk mahasiswa
struct Mahasiswa {
    string nim;
    string nama; // Menambahkan variabel nama
    int nilai;
};

// Class HashTable untuk menyimpan data mahasiswa
class HashTable {
private:
    unordered_map<string, Mahasiswa> data;

public:
    // Fungsi untuk menambahkan data mahasiswa
    void tambahData(const Mahasiswa& mhs) {
        data[mhs.nim] = mhs;
    }

    // Fungsi untuk menghapus data mahasiswa berdasarkan NIM
    void hapusData(const string& nim) {
        data.erase(nim);
    }

    // Fungsi untuk mencari data mahasiswa berdasarkan NIM
    Mahasiswa* cariByNIM(const string& nim) {
        if (data.find(nim) != data.end()) {
            return &data[nim];
        }
        return nullptr;
```

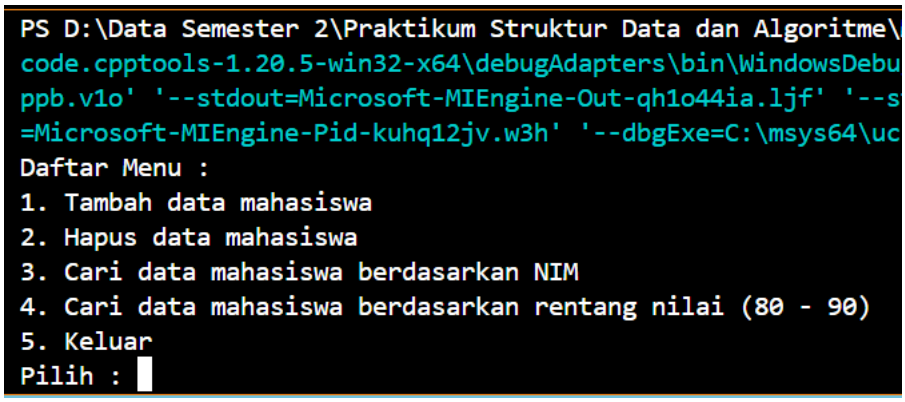
```
    }

    // Fungsi untuk mencari data mahasiswa berdasarkan rentang
nilai (80 - 90)
    vector<Mahasiswa> cariByNilai() {
        vector<Mahasiswa> result;
        for (auto& pair : data) {
            if (pair.second.nilai >= 80 && pair.second.nilai <=
90) {
                result.push_back(pair.second);
            }
        }
        return result;
    }
};

// Fungsi untuk menampilkan menu
void tampilkanMenu() {
    cout << "-=-=-=-=-=Menu-=-=-=-=-=-=-\n";
    cout << "1. Tambah data mahasiswa\n";
    cout << "2. Hapus data mahasiswa\n";
    cout << "3. Cari data mahasiswa berdasarkan NIM\n";
    cout << "4. Cari data mahasiswa berdasarkan rentang nilai (80
- 90)\n";
    cout << "5. Keluar\n";
    cout << "Pilih : ";
}

int main() {
    HashTable hashTable;
    int pilihan;
    string nim;
    string nama; // Menambahkan variabel nama
    int nilai;

    do {
        tampilkanMenu();
        cin >> pilihan;

        switch (pilihan) {
            case 1: {
```

```
                Mahasiswa mhs;
                cout << "Masukkan NIM : ";
                cin >> mhs.nim;
                cout << "Masukkan nama : ";
                cin.ignore(); // Membersihkan buffer sebelum
membaca string
                getline(cin, mhs.nama); // Menggunakan getline
untuk membaca nama dengan spasi
                cout << "Masukkan nilai : ";
                cin >> mhs.nilai;
                hashTable.tambahData(mhs);
                break;
            }
            case 2: {
                cout << "Masukkan NIM mahasiswa yang akan dihapus
: ";
                cin >> nim;
                hashTable.hapusData(nim);
                break;
            }
            case 3: {
                cout << "Masukkan NIM mahasiswa yang akan dicari
: ";
                cin >> nim;
                Mahasiswa* mhs = hashTable.cariByNIM(nim);
                if (mhs != nullptr) {
                    cout << "NIM : " << mhs->nim << ", Nama : "
<< mhs->nama << ", Nilai : " << mhs->nilai << endl;
                } else {
                    cout << "Mahasiswa dengan NIM tersebut tidak
ditemukan!\n";
                }
                break;
            }
            case 4: {
                vector<Mahasiswa> result =
hashTable.cariByNilai();
                if (result.empty()) {
                    cout << "Tidak ada mahasiswa dengan nilai di
rentang (80 - 90)!\n";
                } else {
```

```
                    cout << "Mahasiswa dengan nilai di rentang
(80 - 90) :\n";
                    for (const Mahasiswa& mhs : result) {
                        cout << "NIM : " << mhs.nim << ", Nama :
" << mhs.nama << ", Nilai : " << mhs.nilai << endl;
                    }
                }
                break;
            }
            case 5:
                cout << "Terima kasih!\n";
                break;
            default:
                cout << "Pilihan tidak valid. Silakan pilih
kembali!\n";
        }

    } while (pilihan != 5);

    return 0;
}
```

Screenshoot program

Tampilan Awal



Menambahkan data mahasiswa

```
Daftar 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 : 2311102019
Masukkan nama : Fattah
Masukkan nilai : 88
```

Mencari data mahasiswa berdasarkan NIM

```
Daftar 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 mahasiswa yang akan dicari : 2311102019
NIM : 2311102019, Nama : Fattah, Nilai : 88
```

Mencari data mahasiswa berdasarkan rentang nilai (80-90)

```
Daftar 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
Mahasiswa dengan nilai di rentang (80 - 90) :
NIM : 2311102019, Nama : Fattah, Nilai : 88
```

Menghapus data mahasiswa

```
Daftar 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 mahasiswa yang akan dihapus : 2311102019
```

Deskripsi program

Program ini merupakan implementasi sistem basis data sederhana menggunakan hash table untuk mengelola informasi mahasiswa. Hash table digunakan untuk menyimpan data mahasiswa dengan efisien. Dalam program ini, terdapat struktur data “Mahasiswa” yang memiliki atribut NIM, nama, dan nilai. Selain itu, terdapat kelas HashTable yang menyediakan berbagai operasi seperti penambahan, penghapusan, dan pencarian data mahasiswa berdasarkan NIM atau rentang nilai.

BAB IV

KESIMPULAN

Hash table adalah struktur data yang sangat berguna untuk menyimpan dan mencari data dengan efisien. Praktikum ini telah memberikan pemahaman dasar tentang konsep dan implementasi hash table dalam C++. Berikut adalah beberapa poin yang dapat disimpulkan :

1. Fungsi Hash Table :
   - Hash table menyimpan data dalam tabel yang dibagi menjadi beberapa bucket.
   - Setiap key dipetakan ke bucket tertentu menggunakan fungsi hash.
   - Fungsi hash harus menghasilkan distribusi key yang merata ke seluruh bucket untuk meminimalkan tabrakan.
2. Implementasi Hash Table :
   - Hash table dapat diimplementasikan menggunakan array, linked list, atau struktur data lainnya.
   - Fungsi hash table sederhana digunakan untuk memetakan key ke bucket.
3. Keuntungan dan Kekurangan Hash Table :
   a. Keuntungan :
      - Pencarian data yang sangat cepat, terutama untuk operasi search.
      - Penggunaan memori yang efisien, terutama jika fungsi hash menghasilkan distribusi key yang merata.
   b. Kekurangan :
      - Kinerja pencarian bisa lambat jika terjadi banyak tabrakan.
      - Memerlukan fungsi hash table yang dirancang dengan baik untuk meminimalkan tabrakan.

DAFTAR PUSTAKA

[1] Sutiono, 2023. Hash Table: Pengertian, Fungsi dan Cara Membuat. Diakses pada tanggal 12 Mei 2024, dari https://dosenit.com/kuliah-it/hash-table

[2] Algoritma, 2022. Pengertian Hash Table dan Cara Penggunaannya. Diakses pada tanggal 12 Mei 2024, dari https://algorit.ma/blog/hash-table-adalah-2022/