

LAPORAN PRAKTIKUM
STRUKTUR DATA PEMOGRAMAN

MODUL V
“HASH TABLE”



Disusun Oleh:

NAMA : Trie Nabilla Farhah

NIM : 2311102071

Dosen Pengampu:

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

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

2024

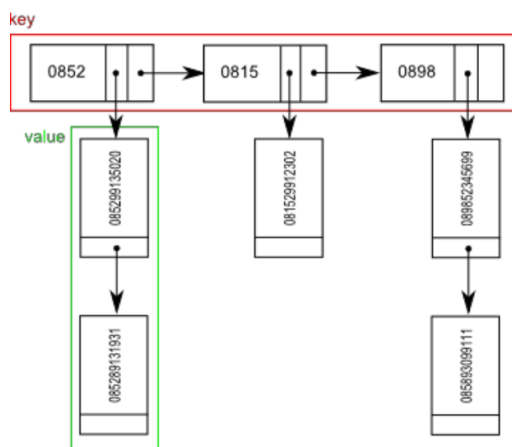
A. DASAR TEORI

a. Pengertian Hash Tabel

Hash table adalah model struktur data yang digunakan untuk menyimpan informasi dan mempermudah proses retrieve. Hash table terdiri atas dua komponen, yaitu key dan value. Key digunakan untuk mengakses data yang disimpan. Key bersifat unik dan biasanya berupa string yang penentuan nilai string tergantung dengan kasusnya. Hash table biasanya direpresentasikan sebagai associative array. Associative array mirip dengan array, tetapi untuk mengakses nomor index-nya array tidak dapat dilakukan secara langsung menggunakan integer, tetapi dengan menggunakan string dengan bantuan hash function yang gunanya untuk mengubah key yang berupa string tadi menjadi nilai integer yang menyatakan index array tempat data disimpan.

Implementasi associative array sebagai multi linked list, disebut juga chain. Pada pembuatan hash table dengan chain, setiap key akan di-generate menggunakan hash function untuk mendapatkan nilai hash value-nya yang digunakan untuk mendapatkan posisi index data yang ingin diakses. Setiap key akan merujuk hanya ke sebuah entry. Entry berupa linked list yang menyimpan data yang memiliki hasil generati hash value yang sama. Dengan hash table, proses pencarian tidak perlu menelusuri seluruh nilai, cukup menelusuri slot yang memiliki key sesuai dengan value yang ingin dicari.

pencarian data yang ingin dicari dilakukan secara sequential pada bucket dengan cara mencocokkan key parameter dengan key yang ada pada data yang dimiliki. Jika ditemukan data yang memiliki key yang sama dengan key parameter, maka untuk method get data tersebut di return sebagai jawaban sedangkan untuk method put, value pada data tersebut akan di-replace dengan data baru yang akan disimpan, dan jika tidak ditemukan maka untuk method get akan me-return nilai null, sedangkan untuk method put artinya data tidak ditemukan sehingga data baru akan ditambahkan pada akhirlist.



b. Fungsi Hash Tabel

Fungsi hash menyimpan nilai asli atau kunci pada alamat yang sama dengan nilai hashnya. Pada pencarian suatu nilai pada tabel hash, yang pertama dilakukan adalah menghitung nilai hash dari kunci atau nilai aslinya, kemudian membandingkan kunci atau nilai asli dengan isi pada memori yang beralamat nomor hashnya. Dengan cara ini, pencarian suatu nilai dapat dilakukan dengan cepat tanpa harus memeriksa seluruh isi tabel satu per satu. Secara teori, kompleksitas waktu ($T(n)$) dari fungsi hash yang ideal adalah $O(1)$. Untuk mencapai itu setiap record membutuhkan suatu kunci yang unik.

c. Operasi Hash Table

1. Insertion

Setiap kali elemen akan dimasukkan, hitung kode hash dari kunci yang diteruskan dan temukan indeks menggunakan kode hash tersebut sebagai indeks dalam array. Gunakan pemeriksaan linier untuk lokasi kosong, jika elemen ditemukan pada kode hash yang dihitung.

2. Deletion

Setiap kali sebuah elemen ingin dihapus, hitung kode hash dari kunci yang diteruskan dan temukan indeks menggunakan kode hash tersebut sebagai indeks dalam array. Gunakan pemeriksaan linier untuk memajukan elemen jika elemen tidak ditemukan pada kode hash yang dihitung. Saat ditemukan, simpan item tiruan di sana untuk menjaga kinerja tabel hash tetap utuh.

3. Searching

Setiap kali sebuah elemen ingin dicari, hitung kode hash dari kunci yang diteruskan dan temukan elemen tersebut menggunakan kode hash tersebut sebagai indeks dalam array. Gunakan pemeriksaan linier untuk memajukan elemen jika elemen tidak ditemukan pada kode hash yang dihitung.

4. Update

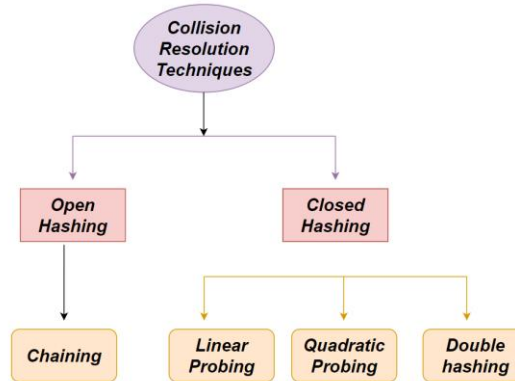
Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

5. Traversal

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

d. Collision Resolution

Tabrakan terjadi ketika dua kunci atau lebih diberi nilai hash yang sama. Misalnya, jika Anda memiliki tabel hash yang memiliki indeks dari 0–6, fungsi hashnya akan menjadi $H(i)\%6$. Misalkan kita harus memasukkan nilai: 2, 6, 24, dan 15 di tabel hash ini. Kemudian fungsi hash akan memetakan nilai 6 dan 24 ke indeks yang sama sehingga menyebabkan tabrakan. Tabrakan juga dapat terjadi karena terbatasnya ukuran tabel hash atau penggunaan fungsi hash yang buruk.



1. Open Hashing (Chaining)

Open hashing atau lebih dikenal dengan chaining adalah salah satu pendekatan paling sederhana untuk menghindari tabrakan dalam tabel hash. Dalam hashing terbuka, setiap slot tabel hash, juga dikenal sebagai bucket, berisi daftar elemen tertaut yang melakukan hash ke slot yang sama. Istilah chaining digunakan untuk menggambarkan proses karena digunakan linked list yang seperti rangkaian elemen.

2. Closed Hashing

Seperti hashing terbuka, hashing tertutup juga merupakan teknik yang digunakan untuk resolusi tabrakan dalam tabel hash. Tidak seperti hashing terbuka, di mana tabrakan diselesaikan dengan merangkai elemen dalam rantai terpisah, hashing tertutup bertujuan untuk menyimpan semua elemen langsung di dalam tabel hash itu sendiri tanpa menggunakan struktur data tambahan seperti daftar tertaut. Dalam hashing tertutup, ketika tabrakan terjadi, dan slot hash sudah terisi, algoritme memeriksa slot berikutnya yang tersedia dalam tabel hash hingga slot kosong ditemukan.

a. Linier Probing

Dalam linear probing, jika terjadi tabrakan dan sebuah slot sudah terisi, algoritma secara linear mencari slot berikutnya yang tersedia secara berurutan. Ia memeriksa slot berikutnya dan terus memeriksa slot berikutnya hingga menemukan slot kosong.

b. Quadrating Probing

Dalam pemeriksaan kuadrat, alih-alih memeriksa slot berikutnya secara linier, algoritme menggunakan fungsi kuadrat untuk menentukan posisi probe berikutnya. Urutan probing mengikuti pola kuadrat sampai ditemukan slot kosong.

c. Double Hashing

menggunakan dua fungsi hash untuk menentukan urutan probe. Ketika tabrakan terjadi, fungsi hash kedua diterapkan untuk menghitung offset, yang kemudian digunakan untuk menemukan slot berikutnya untuk diselidiki.

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;
}

```

Output

The screenshot shows a VS Code terminal window with the following output:

```

PS D:\Semester2\Struktur Data\Praktikum5> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-npbdjiam.llr' '--stdout=Microsoft-MIEngine-Out-c3psogmf.eti' '--stderr=Microsoft-MIEngine-Error-h2iktbcx.lrw' '--pid=Microsoft-MIEngine-Pid-5kokpnwn.sc2' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS D:\Semester2\Struktur Data\Praktikum5>

```

A debugger window is also open, displaying the following information:

Na	
Nama	: Trie Nabilla Farhah
NIM	: 2311102071

The debugger window also shows the current cursor position: Ln 2, Col 17 | 44 characters | 100% | Window | UTF-8.

Deskripsi Program

Program ini menunjukkan cara menggunakan hash table dalam bahasa pemrograman C++. Hash table adalah struktur data yang digunakan untuk menyimpan data dalam format pasangan kunci-nilai. Kelas HashTable, yang memiliki kemampuan dasar untuk menambahkan, mencari, dan menghapus data, digunakan dalam program untuk menggambarkan tabel hash. Berikut ini adalah penjelasan tentang cara program itu bekerja:

- **Penginisialisasi HashTable:** Ketika objek HashTable dibuat dalam fungsi main(), konstruktor HashTable() akan dipanggil. Ini memberikan memori untuk hash table dan mengatur setiap elemen tabel dengan nilai nullptr.
- **Penambahan Data (Insertion):** Pada metode insert(int key, int value), cara menghitung nilai hash dari kunci (key) adalah dengan menggunakan fungsi hash sederhana hash func(key). Setelah itu, elemen baru dibuat dengan mengaitkan kunci-nilai dan dimasukkan ke dalam tabel pada posisi yang tepat. Apabila terdapat elemen yang memiliki kunci yang sama, maka nilainya akan diubah.
- **Pencarian Data (Searching):** Dalam metode get(int key), sistem akan mencari nilai hash dari kunci yang diinputkan dan mencari data di indeks yang sesuai dalam tabel hash. Program akan mencari bagian yang memiliki kunci yang cocok. Jika ditemukan, nilai-nilai yang cocok akan dikembalikan; jika tidak, program akan mengembalikan -1.
- **Penghapusan Data (Deletion):** Dalam metode remove(int key), nilai hash dari kunci akan dihitung dan pencarian dilakukan di indeks yang sesuai dalam tabel hash. Apabila ada elemen dengan kunci yang cocok ditemukan, maka elemen tersebut akan dihapus dari tabel.
- **Traversal:** Fungsi traverse() digunakan untuk menjelajahi semua entri dalam tabel hash dan mencetak pasangan kunci-nilai dari setiap elemen.
- **Pemanggilan Operasi:** Beberapa operasi dasar hash table dipanggil di dalam fungsi main() untuk menunjukkan bagaimana fungsinya bekerja. Beberapa pasangan kunci-nilai dimasukkan ke dalam tabel hash, kemudian dilakukan pencarian untuk beberapa kunci, dan satu elemen dihapus. Output dari operasi-operasi ini akan ditampilkan di konsol.

Program ini memperlihatkan cara hash table bisa dimanfaatkan untuk menyimpan dan mengakses data dengan efisien berdasarkan kunci yang digunakan sebagai referensi untuk mengakses nilai-nilai yang sesuai.

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])
        {

```

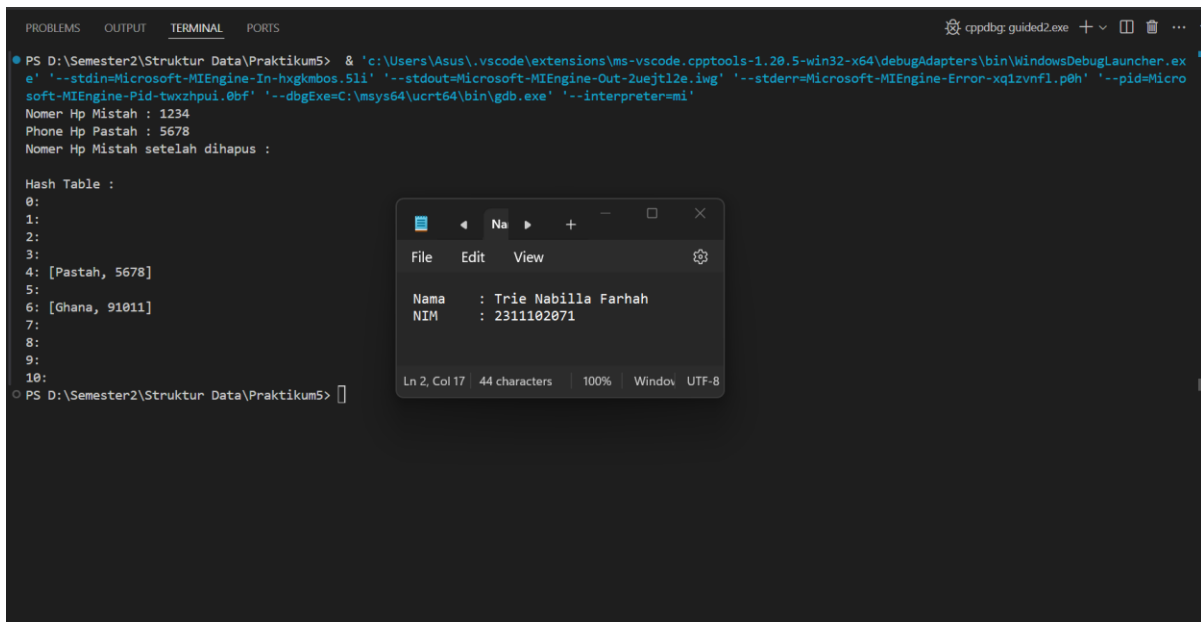
```

        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;
}

```

Output



The screenshot shows a VS Code interface with a terminal window and a small application window. The terminal window displays the following output:

```
PS D:\Semester2\Struktur Data\Praktikum5> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-hxgkmbos.5li' '--stdout=Microsoft-MIEngine-Out-2uejtl2e.iwg' '--stderr=Microsoft-MIEngine-Error-xq1zvnf1.p8h' '--pid=Microsoft-MIEngine-Pid-twxzhpu1.0bf' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
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:\Semester2\Struktur Data\Praktikum5>
```

The small application window, titled "Na", displays the following information:

```
Nama : Trie Nabilla Farhah
NIM : 2311102071
```

The status bar at the bottom of the application window indicates "Ln 2, Col 17", "44 characters", "100%", "Window", and "UTF-8".

Deskripsi Program

Program ini adalah contoh sederhana dari penggunaan struktur data hash map yang menggunakan array dan linked list di setiap elemennya. Berikut ini adalah penjelasan mengenai cara program ini bekerja langkah demi langkah:

- Deklarasi Perpustakaan dan Konstan: Program dimulai dengan mendeklarasikan beberapa berkas header yang diperlukan (iostream, string, dan vector). Variabel TABLE SIZE ditetapkan untuk menentukan dimensi tabel hash.
- Kelas HashNode dinyatakan: Kelas ini menetapkan apa itu node dalam hash map. Setiap simpul memiliki dua rangkaian: nama dan nomor telepon.
- Deklarasi Kelas HashMap: Kelas ini adalah contoh pengimplementasian dari hash map. Di dalam kelas ini, ada array dari vektor pointer tabel yang memiliki ukuran sesuai dengan UKURAN TABEL. Ini adalah tempat sebenarnya di mana elemen-elemen hash map disimpan.
- Fungsi Hashing: hashFunc(string kunci) mengambil suatu string dan menghasilkan sebuah nilai bilangan bulat yang mewakili hasil dari proses hashing dari string tersebut. Hal ini dilakukan dengan menambahkan nilai ASCII dari setiap karakter dalam string, lalu menghitung sisa pembagian hasilnya dengan ukuran TABEL.
- Penambahan Data: metode insert(string name, string phone number) mengambil nama dan nomor telepon karyawan sebagai input. Fungsi ini akan menghasilkan nilai hash dari nama karyawan dengan menggunakan hashFunc() dan kemudian memverifikasi apakah nama itu sudah ada di dalam tabel hash. Jika nomor telepon karyawan sudah ada, maka itu akan diperbarui; jika tidak ada, node baru akan ditambahkan ke dalam vektor di indeks yang sesuai.

- Menghapus Data: fungsi `remove(string name)` menerima nama karyawan sebagai parameter. Fungsi ini akan menghasilkan nilai hash dari sebuah nama dengan menggunakan `hashFunc()` dan kemudian akan mencari node dengan nama yang sama dalam vektor pada indeks tertentu. Apabila ditemukan, node tersebut akan dihapus dari vektor.
- Pencarian Data: fungsi `searchByName(string name)` menerima nama karyawan sebagai parameter. Fungsi ini menjalankan `hashFunc()` untuk menghasilkan nilai hash dari sebuah nama, setelah itu mencari node dengan nama tersebut di dalam vektor pada indeks yang telah ditentukan. Jika ditemukan, nomor teleponnya akan dikembalikan; jika tidak, akan dikembalikan string kosong.
- Pencetakan Hash Map: Fungsi `print()` menampilkan semua informasi dalam hash map, seperti nama dan nomor telepon tiap karyawan, dengan cara mengunjungi setiap vektor dalam tabel dan menampilkan setiap elemen di dalamnya.
- Fungsi Main: Dalam fungsi utama(), beberapa operasi dasar pada peta hash dipanggil untuk menunjukkan bagaimana fungsinya bekerja. Beberapa pegawai dimasukkan ke dalam hash map, kemudian dilakukan pencarian untuk beberapa nama, dan salah satu pegawai dihapus. Output dari operasi-operasi ini akan ditampilkan di layar konsol.

Program ini menunjukkan cara penggunaan hash map dalam mengaitkan kunci (nama karyawan) dengan nilai (nomor telepon) dengan efektif, terutama dalam hal mencari dan menghapus data.

C. UNGUIDED

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 :

- Setiap mahasiswa memiliki NIM dan nilai.
- Program memiliki tampilan pilihan menu berisi poin C.
- Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan rentang nilai (80 – 90).

Unguided 1

Source Code

```
#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 << "]\n";
            }
        }
    }
}

```

```

        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 << "Masukkkkan 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 x, z;
        cout << "Masukkan rentang nilai min : ";
        cin >> x;
        cout << "Masukkan rentang nilai max : ";
        cin >> z;
        data.findNimByScore(x, z);
        break;
    case 5:
        data.print();
        break;
    case 6:
        return 0;
    default:
        cout << "Menu tidak tersedia!" << endl;
        break;
    }
}
return 0;
}

```

Output

The screenshot shows a C++ IDE with a terminal window displaying the program's output. The output shows a menu with 6 options. The user selects option 1 (Tambah data), enters NIM 2311102071 and nilai 90. The program then displays a window with the entered data: Nama: Trie Nabilla Farhah, NIM: 2311102071. The user then selects option 2 (Hapus data), enters NIM 2311102079, and the program displays a message: 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
Masukkan pilihan : 1
Masukkan NIM : 2311102071
Masukkan nilai : 90

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 : 2311102079
Masukkan nilai : 87

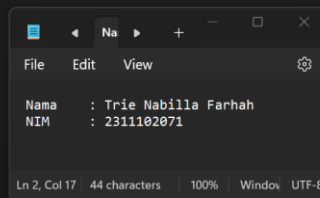
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkan pilihan : 2
Masukkan NIM yang akan dihapus : 2311102079
Data berhasil dihapus

```

```
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkan pilihan : 3
Masukkan NIM yang akan dicari : 2311102071
NIM 2311102071 memiliki nilai 90

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 min : 80
Masukkan rentang nilai max : 90
2311102071 memiliki nilai 90

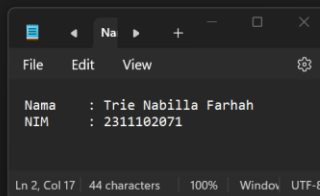
Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkan pilihan : 5
```



```
Masukkan pilihan : 5
0:
1:
2:
3: [2311102071, 90]
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
Masukkan pilihan : 66
Menu tidak tersedia!

Menu
1. Tambah data
2. Hapus data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Cetak data
6. Exit
Masukkan pilihan : 6
PS D:\Semester2\Struktur Data\Praktikum5>
```



Deskripsi Program

Program tersebut adalah implementasi sederhana dari struktur data hash table untuk menyimpan data mahasiswa. Berikut penjelasan cara program tersebut berjalan:

- **Struktur HashNode:** Struktur ini memungkinkan simpul ditunjukkan dalam hash table. Setiap simpul memiliki dua fitur: NIM untuk menyimpan NIM dan nilai untuk menyimpan nilai mahasiswa.
- **Kelas HashMap:** Kelas ini menggunakan hash table. Ini memiliki array table yang terdiri dari array vektor. Setiap elemen vektor menyimpan alamat simpul, atau pointer, yang mengarah ke simpul data. Ini memungkinkan untuk menangani tabrakan hash dengan menambahkan simpul baru ke dalam vektor yang sesuai.
- **Fungsi Hashing Hash:** Fungsi ini mengambil NIM sebagai input dan mengembalikan nilai hash yang dihasilkan dari NIM. Metode hashing yang digunakan cukup sederhana, yaitu menjumlahkan kode ASCII dari setiap karakter dalam NIM dan kemudian mengambil sisa pembagian dengan ukuran tabel hashing.

- Operasi penambahan, penghapusan, dan pencarian: Ketika pengguna memasukkan data baru, fungsi penambahan() akan menghasilkan nilai hash untuk NIM dan memasukkan simpul baru ke dalam vektor yang sesuai. Nilai mahasiswa akan diperbarui jika NIM sudah ada dalam tabel.
- Pencarian Berdasarkan Rentang Nilai findNimByScore(): Fungsi ini menemukan dan mencetak NIM dan nilai siswa yang berada dalam rentang nilai yang ditentukan oleh pengguna. Ini dicapai dengan meninjau setiap vektor dalam tabel hash dan memeriksa setiap simpul untuk memastikan apakah nilai siswa berada dalam rentang yang ditentukan.
- Menu Utama: Pada awal program, pengguna dapat melihat menu yang memungkinkan mereka memilih apa yang ingin mereka lakukan. Mereka dapat menambah, menghapus, mencari data menggunakan NIM, mencari data menggunakan rentang nilai, mencetak semua data, atau keluar dari program.
- Perulangan Menu: Setelah pengguna memilih operasi, program memproses perintah sesuai dengan pilihan pengguna. Setelah operasi selesai, program menampilkan menu utama kembali.
- Keluar dari Program: Saat pengguna memilih opsi keluar (menu == 6), program keluar dari perulangan menu dan fungsi main(), sehingga program selesai dieksekusi.

Oleh karena itu, program ini menggunakan hash table untuk memberi pengguna antarmuka yang interaktif untuk mengelola data siswa. Ini memungkinkan pengguna untuk menambahkan, menghapus, dan mencari data siswa berdasarkan NIM atau rentang nilai dengan cepat dan efektif.

D. KESIMPULAN

Algoritma hash table dapat diimplementasikan untuk mengarsipkan dan mencari suatu data dengan waktu operasi mencapai $O(1)$ karena hash table akan memetakan data sesuai dengan kunci yang telah diberikan. Hash table dapat dikembangkan untuk mencari pola-pola tertentu dalam sebuah string atau kalimat, hash table berfungsi untuk memetakan urutan dari setiap karakter. Bila ada suatu pola yang dicari, sistem pencarian yang digunakan adalah mencari karakter pertama, bila karakter pertama ditemukan baru kemudian dicocokkan sesuai dengan urutan karakter pada pola yang dicari dan begituseterusnya sampai kepada karakter terakhir pada pola.

Dengan demikian hash table mempersingkat waktu pencarian, karena bila ada satu karakter saja yang tidak cocok, maka pencarian langsung berpindah pada bagian selanjutnya, jadi tidak mencari secara sekuensial.

E. REFERENSI

Asisten praktikum, “*Modul 5 Hash Table*”, learning Management System, 2024

Isjhar, K. Kemas, R.S. Gia, S.W. (2019). “*Analisis Performansi Metode Graph Decomposition Index Pada Graph database*”, e-proceeding Of Engineering. 2(3), 1730.

Izhar, R. Nizirwan, A. Agung, M.W. Kundang, K.J. Iwan, S. (2023). “*Komparasi Fungsi Hash Md5 dan Sha256 Dalam Keamanan Gambar dan Teks*”, Jurnal IKRAITH-INFORMATIKA. 7(2), 42.

Jasson, P. (2020). “*Aplikasi Table Hash dalam Pengarsipan dan Pencarian Data*”, Jurnal Teknologi Informasi. 4(5), 55.

Tutorialspoint. *Hash Table Data Structur* . Diakses pada 10 Mei 2024, dari https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm

Iqra, M. *Hash Table Collision Resolution*. Diakses pada 10 Mei 2024, dari <https://www.educative.io/answers/hash-table-collision-resolution>