

**LAPORAN PRAKTIKUM STRUKTUR DATA
DAN ALGORITMA MODUL 5
“Hash Table”**



DISUSUN OLEH:
Farrell Edric Kelvianto
2311102079
S1 IF-11-B

DOSEN:

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

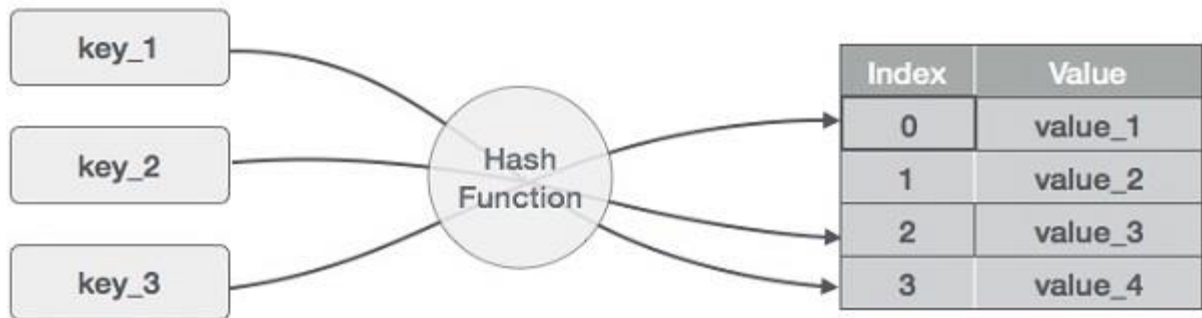
**PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2024**

A. Dasar Teori

a. Pengertian Hash Tabel

Hash Table adalah struktur data yang berfungsi untuk menyimpan dan mengakses dengan cara kerjanya mengonversi kunci untuk dapat mengakses dari dalam tabel hashnya itu sendiri. Hash tabel itu terdiri dari dua komponen utama array, vektor dan fungsi hash. Array menyimpan data dalam slot yang biasa disebut dengan bucket. Dari setiap bucket ini dapat menampung beberapa item data, ini sama halnya seperti array biasa tetapi disini digunakan untuk lebih kompleks lagi dan jauh lebih dinamis dibandingkan array biasa.

Sistem hash tabel ini bekerja dengan cara mengambil dari kunci tadi dan dimasukan kedalam indeks array menggunakan fungsi hash. Lalu, ketika data akan dicari, kuncinya itu sebagai indikasi parameter untuk menjadi fungsi hashnya, dan posisi indeks array untuk mencari sebuah data tersebut. Didalam kasus hash collision, yang dimana dua atau lebih dari data yang dimiliki nilai hash yang sama, hash tabel itu menyimpan data tersebut dalam slot yang sama atau biasa disebut chaining. Sebagai berikut gambaran dari Hashing.



b. Operasi Basic Hash Tabel

Sebagai berikut operasi dari hash tabel:

1. Search – Berfungsi untuk mencari dari data atau elemen dari tabel hash tersebut.
2. Insert – ini berfungsi menambahkan data kedalam tabel hash tersebut dan dapat diakses menggunakan search yang telah dibuat.
3. Delete – berfungsi untuk menghapus data dalam setiap tabelnya.
4. Update – berfungsi untuk memperbarui data dari hash tabel itu sendiri.

c. Resolusi Collision

Resolusi Collision ini adalah tabrakan dalam tabel hash tabrakan terjadi ini ketika dua kunci atau lebih dari nilai hash yang sama. Contohnya ketika kita memiliki tabel hash memiliki indeks 0 – 6 , lalu kita masukan nilai: 3, 5, 6 dan 10 di tabelnya. Kemudian fungsi hash akan memetakan nilai 6 dan 10 ke indeks yang sama sehingga menyebabkan tabrakan. Tabrakan ini juga terjadi karena setiap tabel ini memiliki batas tertentu ketika kita membuat data lebih dari batas tersebut maka akan terjadi tabrakan dan penggunaan fungsi hash yang buruk.

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

```

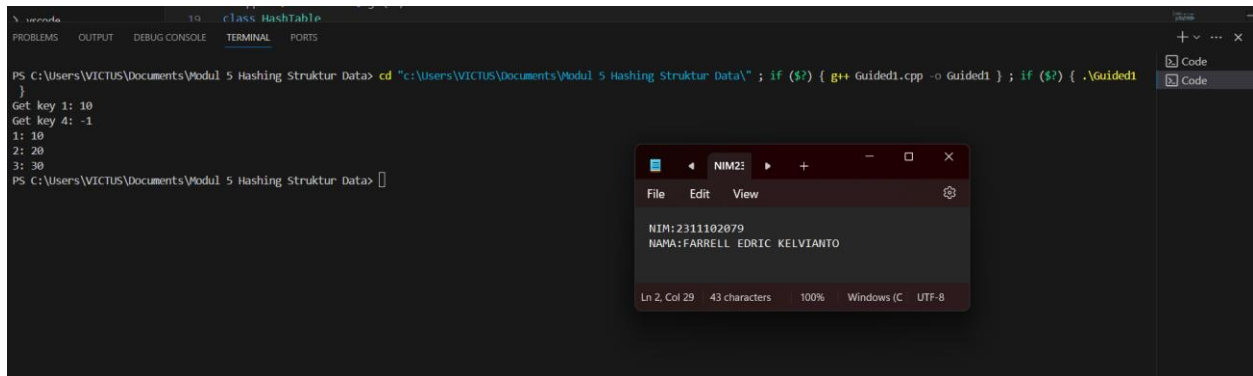
```

        ht.traverse();

        return 0;
    }

```

Screenshot Output



Deskripsi Program

Cara kerja dari program ini adalah pertama terdapat deklarasi struct Node ini mendeklarasikan sebuah kunci dan nilai dari isi node tersebut. lalu beralih kedalam class hashtable terdapat sebuah private tabel yang berfungsi untuk menyimpan data secara private dan ini memerlukan kunci untuk mengaksesnya. Dan untuk mengambil sebuah kuncinya ini menggunakan fungsi hashnya ini. Dan sebagai berikut fungsi dari atau implementasi didalam class tersebut:

1. Constructor Dan Destructor yang berfungsi untuk menginisialisasi tabel hash dengan alokasi memori dinamis untuk array tersebut. Destructor itu untuk membersihkan dari memori yang telah dialokasikan untuk array dan semua yang disimpan didalamnya.
2. Searching ini digunakan untuk mencari nilai yang terkait dengan kunci tertentu yang seperti dijelaskan tadi itu menggunakan kunci di dalam hashnya itu sendiri untuk membuka indeksnya tersebut.
3. Insert ini digunakan untuk memasukan sebuah kunci dan data kedalam tabel hash. Jika tidak ada tabrakan, node baru dibuat dan dimasukan kedalam tabel. Jika ada tabrakan, maka teknik chaining inilah yang digunakan dengan menambahkan node baru di depan linked list.
4. Remove ini digunakan untuk mencari nilai yang akan dihapus sebuah entri dari tabel jika kunci dan nilai nya sesuai maka dia akan menghapus dari nilai di dalam tabel hash tersebut.
5. Traversal ini digunakan untuk menelusuri seluruh dari tabel hash dan mencetak kunci dan nilai dari setiap entrinya.

Dan untuk didalam fungsi utamanya ini sebagai implementasi dari setiap isi dalam classnya tersebut.

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;
        }
    }

```

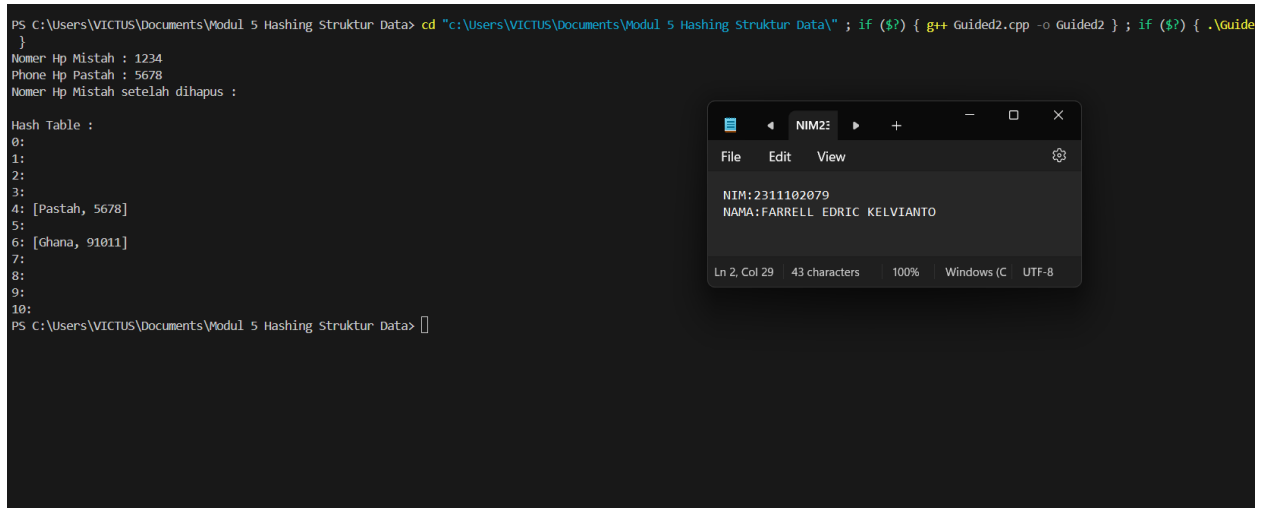


```

    }
};
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 C:\Users\VICTUS\Documents\Modul 5 Hashing Struktur Data> cd "c:\Users\VICTUS\Documents\Modul 5 Hashing Struktur Data\" ; if ($?) { g++ Guided2.cpp -o Guided2 } ; if ($?) { .\Guided2 }
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 C:\Users\VICTUS\Documents\Modul 5 Hashing Struktur Data>

```

Inset window content:

```

NIM:2311102079
NAMA:FARRELL EDRIC KELVIANITO

```

Status bar: Ln 2, Col 29 | 43 characters | 100% | Windows (C) | UTF-8

Deskripsi Program

Untuk guided 2 ini menggunakan vector yang memungkinkan mempersingkat dari sebuah program dan lebih simpel dalam menggunakannya. Cara kerja program ini ditahap awal terdapat libraries yang berisikan string, vector dan iostream. Konstanta tabel ini difungsikan

untuk menentukan ukuran dari hash tabelnya itu sendiri. Didalam simpulnya terdapat menyatakan nama dan nomor telepon yang digunakan untuk di bagian fungsi classnya tersebut. Class HashMap ini terdiri dari sebuah beberapa perintah sebagai berikut:

1. Atribut ini memiliki vector yang berukuran dari hash tabelnya itu sendiri.
2. Fungsi Hash itu sendiri atau (hashFunch) untuk memberikan kunci yaitu “nama” yang berfungsi untuk mengembalikan hasil dari modulo Table-Sizenya .
3. Insert ini berfungsi untuk memasukan nama dan nomor telepon yang akan menjadi data baru dan dimasukan kedalam tabel hash. Nilai hash akan dihitung untuk nama yang diberikan. Lalu pencarian dilakukan di vektor yang sesuai dengan hashnya tersebut.
4. Remove ini berfungsi untuk menghapus setiap simpul nama yang telah diinputkan. Lalu setelah itu vector akan mencari nama yang sesuai ketika nama ini sesuai maka akan dihapus simpulnya / nama yang akan dihapus.
5. Search ini difungsikan untuk mencari sebuah nama menggunakan vector, jika vectornya mencari sesuai dengan nama yang dicari maka ditemukan dan untuk menemukannya ini menggunakan perhitungan nama yang diinputkan oleh pengguna. Jika kosong maka akan dikembalikan.
6. Fungsi mainnya ini berfungsi untuk merepresentasikan yang telah dibuat oleh class atau implementasinya.

B. Tugas

Unguided 1

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>

using namespace std;

class Mahasiswa
{
public:
    string nama;
    long long int NIM;
    int nilai;

    Mahasiswa(string nama, long long int NIM, int nilai) : nama(nama),
NIM(NIM), nilai(nilai) {}
};

class HashNode
{
public:
```

```

vector<Mahasiswa> mahasiswa;

void insert(Mahasiswa mahasiswa)
{
    mahasiswa.push_back(mahasiswa);
}

void remove(long long int NIM)
{
    for (auto it = mahasiswa.begin(); it != mahasiswa.end(); ++it)
    {
        if (it->NIM == NIM)
        {
            mahasiswa.erase(it);
            return;
        }
    }
}

Mahasiswa *pencarianmhs(long long int NIM)
{
    for (auto &mahasiswa : mahasiswa)
    {
        if (mahasiswa.NIM == NIM)
        {
            return &mahasiswa;
        }
    }
    return nullptr;
}

vector<Mahasiswa *> rentanNilai(int minNilai, int maxNilai)
{
    vector<Mahasiswa *> result;
    for (auto &mahasiswa : mahasiswa)
    {
        if (mahasiswa.nilai >= minNilai && mahasiswa.nilai <= maxNilai)
        {
            result.push_back(&mahasiswa);
        }
    }
    return result;
}
};

```

```

class HashMap
{
private:
    static const int TABLE_SIZE = 20;
    HashNode *table[TABLE_SIZE];

public:
    HashMap()
    {
        for (int i = 0; i < TABLE_SIZE; ++i)
        {
            table[i] = new HashNode();
        }
    }

    int hashFunc(long long int NIM)
    {
        return NIM % TABLE_SIZE;
    }

    void insert(Mahasiswa mahasiswa)
    {
        int hash_val = hashFunc(mahasiswa.NIM);
        table[hash_val]->insert(mahasiswa);
    }

    void remove(long long int NIM)
    {
        int hash_val = hashFunc(NIM);
        table[hash_val]->remove(NIM);
    }

    Mahasiswa *pencarianmhs(long long int NIM)
    {
        int hash_val = hashFunc(NIM);
        return table[hash_val]->pencarianmhs(NIM);
    }

    vector<Mahasiswa *> rentanNilai(int minNilai, int maxNilai)
    {
        vector<Mahasiswa *> result;
        for (int i = 0; i < TABLE_SIZE; ++i)
        {
            vector<Mahasiswa *> temp = table[i]->rentanNilai(minNilai,
maxNilai);

```

```

        result.insert(result.end(), temp.begin(), temp.end());
    }
    return result;
}

void display() {

    cout << "                Data Mahasiswa                " << endl;

    for (int i = 0; i < TABLE_SIZE; ++i) {
        if (!table[i]->mahasiswas.empty()) {
            cout << "Slot " << setw(3) << i << " | ";
            for (auto& mahasiswa : table[i]->mahasiswas) {
                cout << "Nama: " << setw(15) << mahasiswa.nama << " | "
                    << "NIM: " << setw(10) << mahasiswa.NIM << " | "
                    << "Nilai: " << setw(3) << mahasiswa.nilai << " | ";
            }
            cout << endl;
        }
    }

}

};

int main()
{
    HashMap hashTable;
    int pilih;
    while (true)
    {
        cout << "\nMenu Pilihan \n\n";
        cout << "1. Tambah Data Mahasiswa \n";
        cout << "2. Menghapus Data Mahasiswa \n";
        cout << "3. Mencari Mahasiswa \n";
        cout << "4. Mencari Rentan Nilai \n";
        cout << "5. Display \n";
        cout << "0. Exit \n";
        cout << "Pilih: ";
        cin >> pilih;

        switch (pilih)
        {
            case 1:
            {

```

```

        string nama;
        long long int NIM;
        int nilai;
        cout << "Masukkan Nama Mahasiswa: ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan NIM Mahasiswa: ";
        cin >> NIM;
        cout << "Masukkan Nilai Mahasiswa: ";
        cin >> nilai;
        Mahasiswa mahasiswa(nama, NIM, nilai);
        hashTable.insert(mahasiswa);
        cout << "Mahasiswa telah ditambahkan! " << endl;
        break;
    }
    case 2:
    {
        long long int NIM;
        cout << "Masukan NIM mahasiswa yang ingin dihapus: ";
        cin >> NIM;
        hashTable.remove(NIM);
        cout << "Data Mahasiswa Telah Dihapus! " << endl;
        break;
    }
    case 3:
    {
        long long int NIM;
        cout << "Masukkan NIM Mahasiswa: ";
        cin >> NIM;
        Mahasiswa *mahasiswa = hashTable.pencarianmhs(NIM);
        if (mahasiswa != nullptr)
        {
            cout << "Data Mahasiswa:" << endl;
            cout << "Nama: " << mahasiswa->nama << endl;
            cout << "NIM: " << mahasiswa->NIM << endl;
            cout << "Nilai: " << mahasiswa->nilai << endl;
        }
        else
        {
            cout << "Mahasiswa tidak ditemukan." << endl;
        }
        break;
    }
    case 4:
    {

```

```

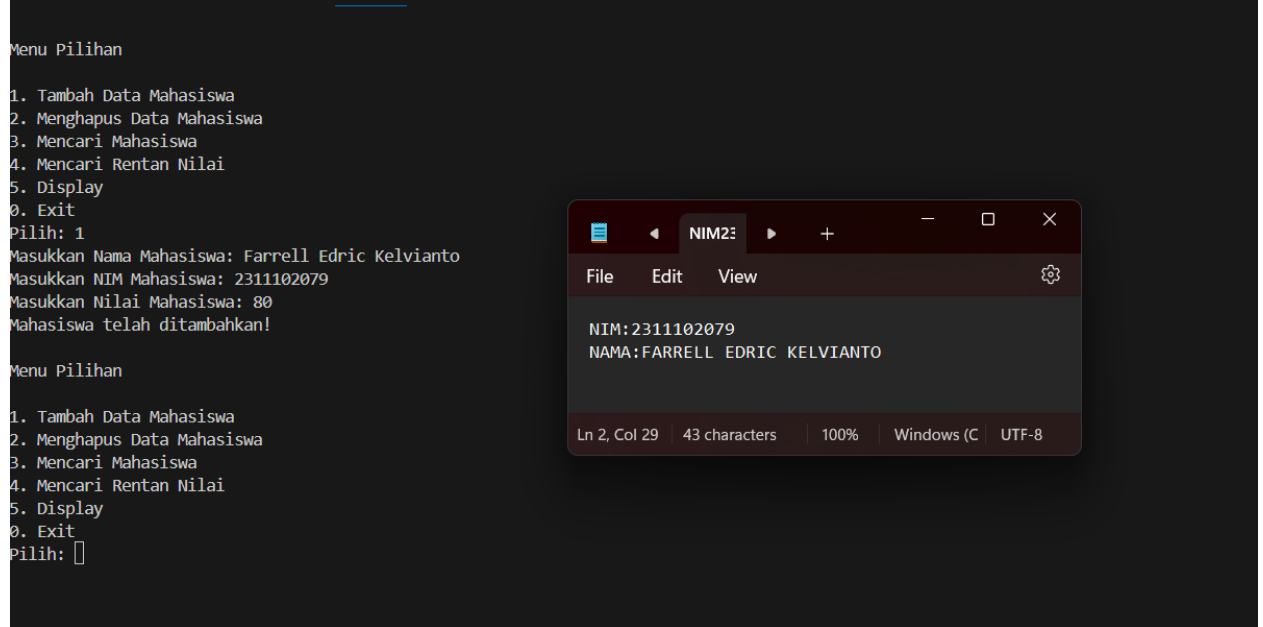
        int minNilai, maxNilai;
        cout << "Masukkan rentang nilai min: ";
        cin >> minNilai;
        cout << "Masukan rentang nilai max: ";
        cin >> maxNilai;
        vector<Mahasiswa *> mahasiswa = hashTable.rentanNilai(minNilai,
maxNilai);
        if (mahasiswa.empty())
        {
            cout << "Tidak ada mahasiswa dengan nilai dalam rentang
tersebut." << endl;
        }
        else
        {
            cout << "Mahasiswa dengan nilai dalam rentang " << minNilai <<
" - " << maxNilai << " adalah:" << endl;
            for (auto mahasiswa : mahasiswa)
            {
                cout << "Nama: " << mahasiswa->nama << ", NIM: " <<
mahasiswa->NIM << ", Nilai: " << mahasiswa->nilai << endl;
            }
        }
        break;
    }
    case 5:
        hashTable.display();
        break;
    case 0:
        return 0;
    default:
        cout << "Pilihan tidak valid." << endl;
        break;
    }
}
return 0;
}

```

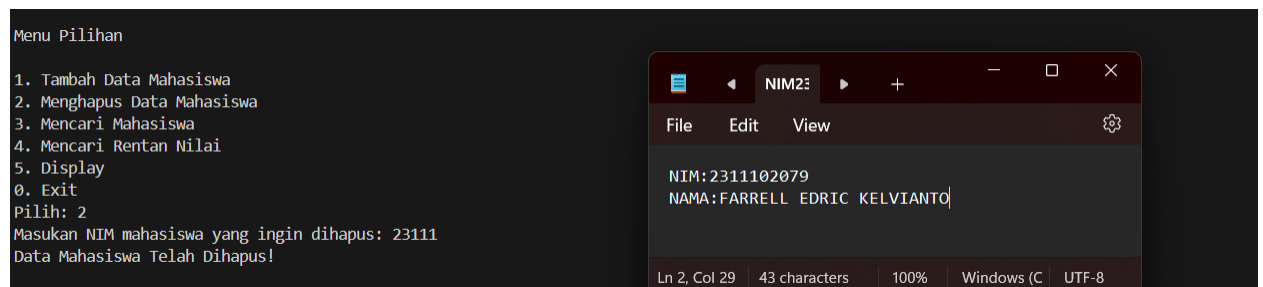
Screenshot Output

Soal No. 1 Membuat menu untuk menambahkan ,mengubah dan melihat nama dan NIM mahasiswa, sebagai berikut outputnya :

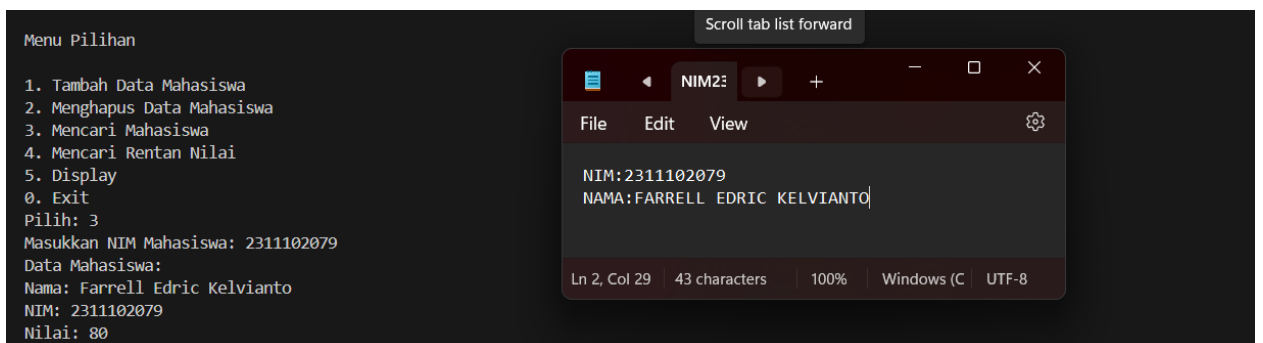
- Tampilan Operasi Insert



- Tampilan Operasi Remove

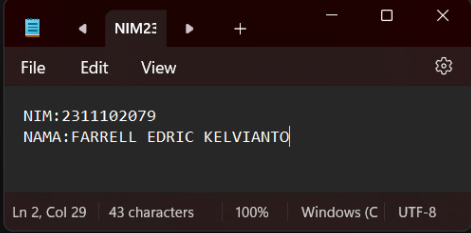


- Tampilan Pencarian Mahasiswa



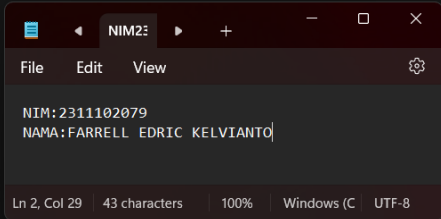
- Tampilan Operasi Rentang Nilai


```
Menu Pilihan
1. Tambah Data Mahasiswa
2. Menghapus Data Mahasiswa
3. Mencari Mahasiswa
4. Mencari Rentan Nilai
5. Display
0. Exit
Pilih: 4
Masukkan rentang nilai min: 70
Masukan rentang nilai max: 90
Mahasiswa dengan nilai dalam rentang 70 - 90 adalah:
Nama: Jared , NIM: 23111, Nilai: 90
Nama: Farrell Edric Kelvianto, NIM: 2311102079, Nilai: 80
```



- Tampilan Operasi Tampil Data.

```
Data Mahasiswa
Slot 19 | Nama: Farrell Edric Kelvianto | NIM: 2311102079 | Nilai: 80 |
Menu Pilihan
1. Tambah Data Mahasiswa
2. Menghapus Data Mahasiswa
3. Mencari Mahasiswa
4. Mencari Rentan Nilai
5. Display
0. Exit
Pilih: 
```



Deskripsi Program

Cara kerja program ini adalah

- classMahasiswa digunakan untuk membuat variabel yang digunakan untuk menginisialisasi setiap program – program nya ini.
- Class hash node ini berguna untuk membuat semua perintah yang akan dijelaskan dibawah ini dan menggunakan vektor karena untuk mempersingkat dari program tersebut.
- Insert () berfungsi untuk memetakan data ke dalam hash table
- Display () berfungsi untuk menampilkan hasil data dari mahasiswa.
- Remove () berfungsi untuk menghapus mahasiswa menggunakan NIM untuk menghapus dari hash tabel tersebut.
- rentanNilai () ini difungsikan untuk mencari rentan nilai dari mahasiswa yang telah diinputkan.
- mencariMahasiswa () ini berfungsi untuk mencari dari data mahasiswa tersebut dalam indeks hash tabel ketika dia tidak ada maka data mahasiswa tersebut tidak ada.

Kesimpulan

Kesimpulannya dari hash tabel ini adalah sangatlah dinamis daripada array atau linked list kemarin dan disini menggunakan vektor sehingga sangatlah singkat sekali dalam pemrogramannya. Dan untuk membuat hash tabel itu memerlukan sebuah key yang berguna untuk membuka dari file keynya tersebut yang dapat dilakukan dengan cara menjadikannya ASCII setelah dijadikannya, kita dapat mengaksesnya kembali menggunakan penjumlahan ASCIInya tadi dan implementasinya ini adalah sebagian juga dari stack.

Referensi

Asisten Praktikum. “Modul 5 Hash Tabel”. Learning Management System. 2024

