LAPORAN PRAKTIKUM

MODUL V
HASH TABLE

Disusun oleh:
Avriel Fitria Rachma Suryantari
NIM: 2311102036

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 Hash Code.
2. Mahasiswa mampu menerapkan Hash Code kedalam pemrograman.

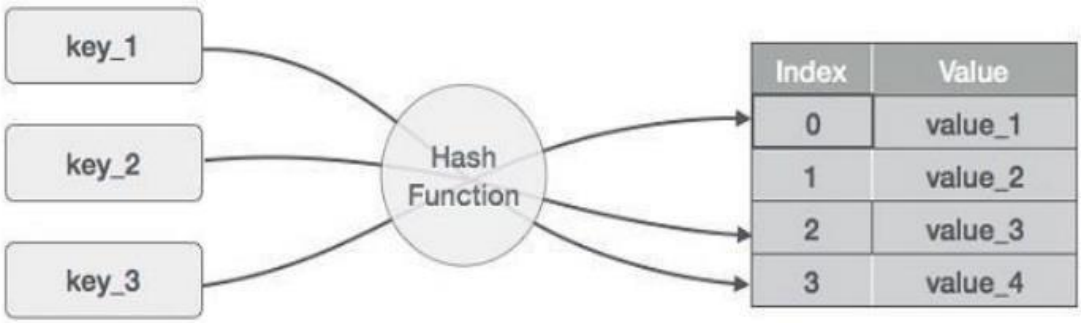
BAB II

DASAR TEORI

Tabel Hash adalah struktur data yang digunakan untuk menyimpan pasangan kunci/nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vector) dan fungsi hash. Dengan menggunakan fungsi hash yangbaik hashing dapat berjalan dengan baik. Hashing adalah teknik untukmengubah rentang nilai kunci menjadi rentang indeks array.

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.

Sistem hash table bekerja dengan cara mengambil input kunci dan memetakkannya ke nilai indeks array menggunakan fungsi hash. Kemudian, data disimpan pada posisi indeks array yang dihasilkan oleh fungsi hash. Ketika data perlu dicari, input kunci dijadikan sebagai parameter untuk fungsihash, 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.



Fungsi hash membuat pemetaan antara kunci dan nilai, hal ini dilakukanmelalui penggunaan rumus matematika yang dikenal sebagai fungsi hash. Hasil dari fungsi hash disebut sebagai nilai hash atau hash. Nilai hash adalahrepresentasi dari string karakter asli tetapi biasanya lebih kecil dari aslinya.

Contoh: Pertimbangkan sebuah array sebagai Peta di mana kuncinya adalah indeks dan nilainya adalah value pada indeks itu. Jadi untuk array A jika kita memiliki indeks i yang akan diperlakukan sebagai kunci maka kita dapat menemukan nilainya hanya dengan mencari value pada A[i].

Tipe fungsi Hash :

a. Division Method.
b. Mid Square Method.
c. Folding Method.
d. Multiplication Method.

Untuk mencapai mekanisme hashing yang baik,penting untuk memiliki fungsi hash yangbaik dengan persyaratan dasar sebagai berikut:

a. Dapat dihitung secara efisien.
b. Harus mendistribusikan kunci secara beragam (setiap posisi tabel memiliki kemungkinan yang sama untuk masing-masing.)
c. Harus meminimalkan tabrakan.

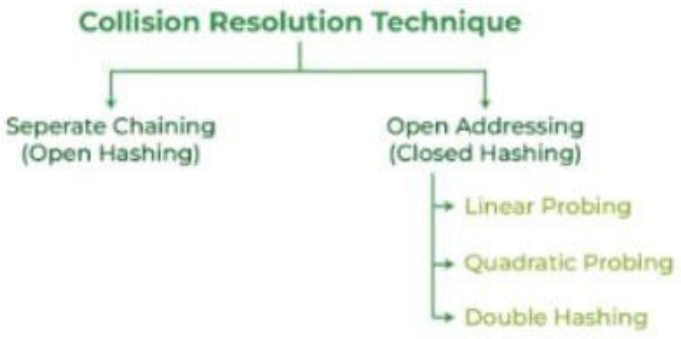
d. Harus memiliki factor muatan rendah (jumlah item dalam tabel dibagi dengan ukuran tabel)

Operasi HashTable

1. Insertion: Memasukkan data baru ke dalam hash table dengan memanggil fungsihash untuk menentukan posisi bucket yang tepat, dan kemudian menambahkan data ke bucket tersebut.
2. Deletion: Menghapus data dari hash table dengan mencari data menggunakan fungsi hash, dan kemudian menghapusnya dari bucket yang sesuai.
3. Searching: 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: 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 tabel

Collusion Resolution

Keterbatasan tabel hash adalah jika dua angka dimasukkan ke dalam fungsi hash menghasilkan nilai yang sama. Hal ini disebut dengan collision. 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) Close 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..

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()
```

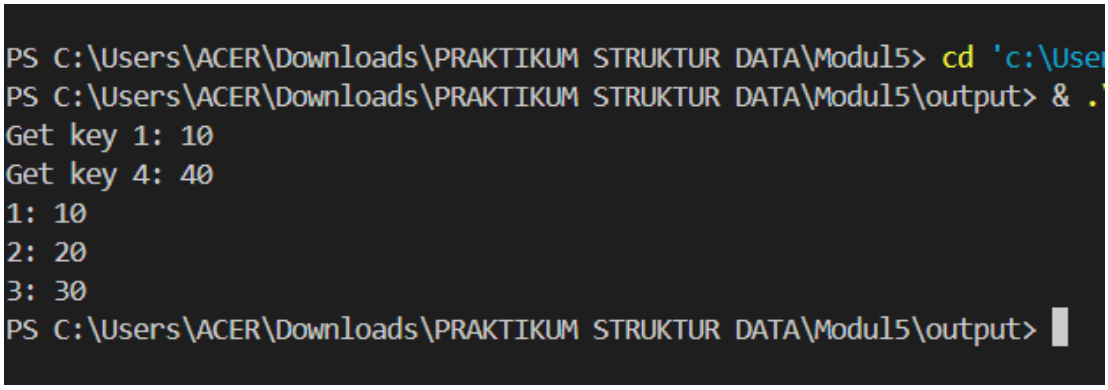
```
    {
        for (int i = 0; i < MAX_SIZE; i++)
        {
            Node *current = table[i];
            while (current != nullptr)
            {
                Node *temp = current;
                current = current->next;
                delete temp;
            }
        }
        delete[] 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);
    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



Deskripsi program

Program di atas merupakan implementasi sederhana dari struktur data hash table dalam bahasa C++. Hash table digunakan untuk menyimpan data dengan menggunakan fungsi hash untuk menentukan lokasi penyimpanan data dalam array. Setiap elemen dalam array disebut sebagai "node" yang berisi pasangan kunci-nilai. Program ini memungkinkan operasi dasar seperti penambahan data (insertion), pencarian data (searching), penghapusan data (deletion), dan penelusuran semua data yang tersimpan dalam hash table (traversal). Setiap node dalam hash table dihubungkan menggunakan linked list untuk menangani kasus-kasus tabrakan hash (collision).

2. Guided 2

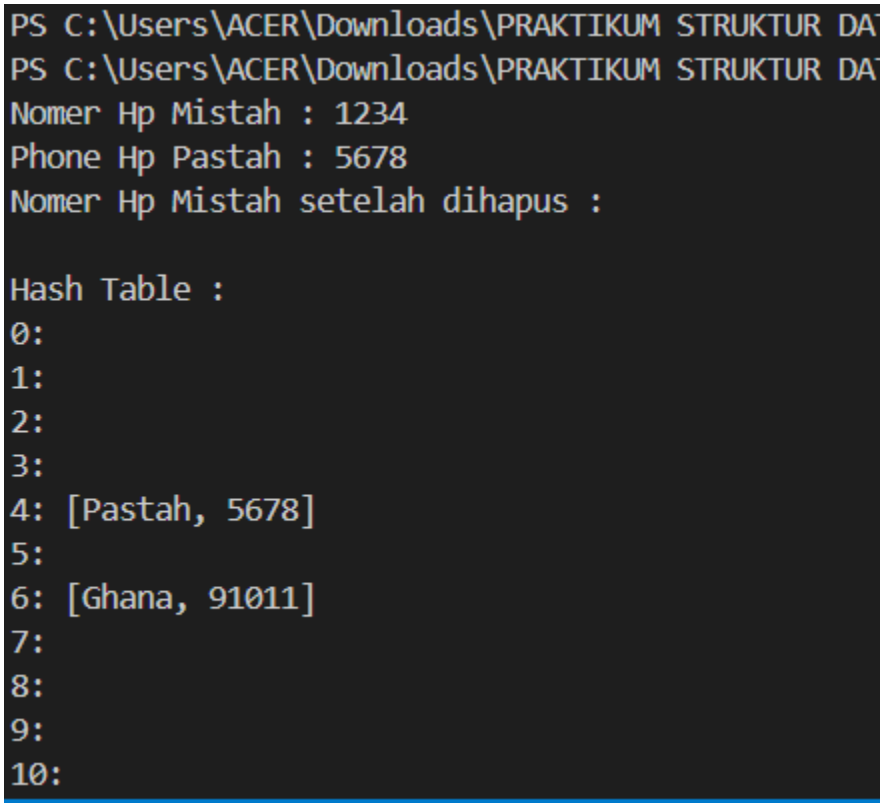
Surce Code

```
#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;
    }
```

Screenshot Program



Deskripsi Program

Program di atas merupakan implementasi sederhana dari hash map dalam bahasa C++, yang digunakan untuk menyimpan data pasangan nama dan nomor telepon. Program menggunakan fungsi hash sederhana untuk menentukan lokasi penyimpanan data dalam array. Setiap elemen dalam array disebut sebagai "node" yang berisi pasangan nama dan nomor telepon. Program ini memungkinkan operasi dasar seperti penambahan data, pencarian data berdasarkan nama, dan penghapusan data. Data-data tersebut kemudian ditampilkan dalam bentuk hash table untuk visualisasi.

LATIHAN KELAS - UNGUIDED

1. Impelementasikan 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 nilai (80-90).

   Source code

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

using namespace std;

struct Mahasiswa
{
    string nama;
    string nim;
    int nilai;
};

class HashTable
{
private:
    vector<vector<Mahasiswa>> data;

    int hash(string nim)
    {
        int sum = 0;
```

```
        for (char c : nim)
        {
            sum += c;
        }
        return sum % data.size();
    }

public:
    HashTable(int size)
    {
        data.resize(size);
    }

    void addData(string nama, string nim, int nilai)
    {
        Mahasiswa mhs = {nama, nim, nilai};
        int index = hash(nim);
        data[index].push_back(mhs);
        cout << "Data Mahasiswa Berhasil Ditambahkan " << endl;
    }

    void removeData(string nim)
    {
        int index = hash(nim);
        vector<Mahasiswa> &bucket = data[index];
        for (int i = 0; i < bucket.size(); i++)
        {
            if (bucket[i].nim == nim)
            {
                cout << "Data Mahasiswa : " << bucket[i].nama <<
" ( " << bucket[i].nim << " ) telah dihapus" << endl;
                bucket.erase(bucket.begin() + i);
                break;
            }
```

```
        }
    }

    Mahasiswa *findDatabyNim(string nim)
    {
        int index = hash(nim);
        vector<Mahasiswa> &bucket = data[index];
        for (int i = 0; i < bucket.size(); i++)
        {
            if (bucket[i].nim == nim)
            {
                cout << "Nama   : " << bucket[i].nama << endl;
                cout << "NIM    : " << bucket[i].nim << endl;
                cout << "Nilai  : " << bucket[i].nilai << endl;
                return &bucket[i];
            }
        }
        return nullptr;
    }

    vector<Mahasiswa   *>   findDataByScore(int   minScore,   int
maxScore)
    {
        vector<Mahasiswa *> result;
        for (vector<Mahasiswa> &bucket : data)
        {
            for (Mahasiswa &mhs : bucket)
            {
                if  (mhs.nilai  >=  minScore  &&  mhs.nilai  <=
maxScore)
                {
                    cout << "Nama   : " << mhs.nama << endl;
                    cout << "NIM    : " << mhs.nim << endl;
                    cout << "Nilai  : " << mhs.nilai << endl;
```

```
                    result.push_back(&mhs);
                }
            }
        }
        return result;
    }

    void printData()
    {
        for (vector<Mahasiswa> &bucket : data)
        {
            for (Mahasiswa &mhs : bucket)
            {
                cout << "Nama   : " << mhs.nama << endl;
                cout << "NIM    : " << mhs.nim << endl;
                cout << "Nilai  : " << mhs.nilai << endl;
                cout << endl;
            }
        }
    }
};

int main()
{
    HashTable mahasiswaTable(100);
    int choice;
    string nama, nim;
    int nilai;

    do
    {
        cout << endl;
        cout << "Data Mahasiswa" << endl;
```

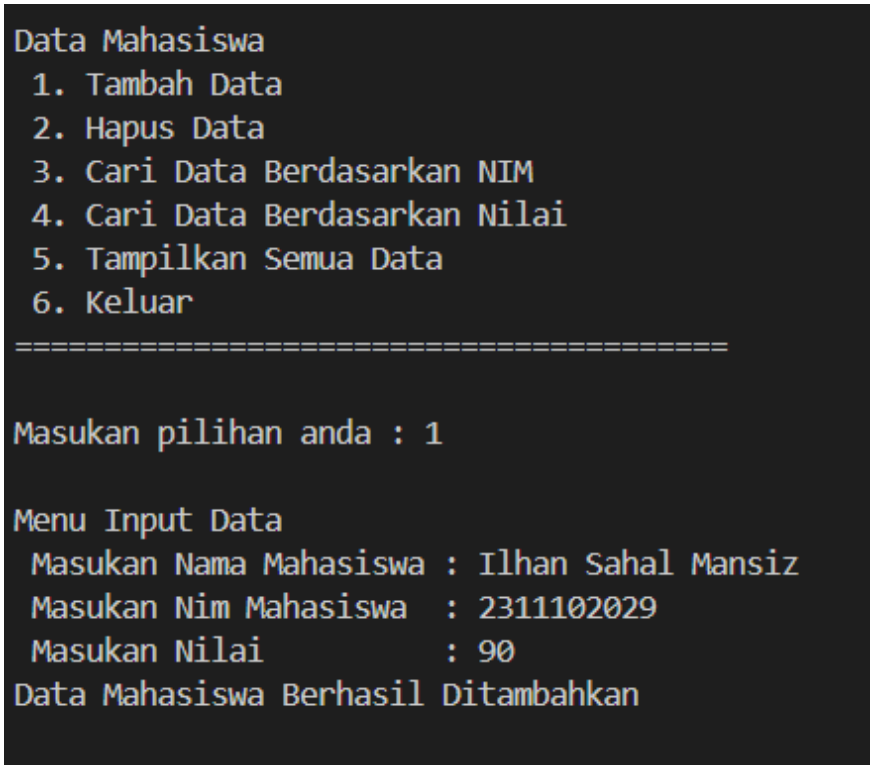
```
        cout << " 1. Tambah Data                                " <<
endl;
        cout << " 2. Hapus Data                                 " <<
endl;
        cout << " 3. Cari Data Berdasarkan NIM                  " <<
endl;
        cout << " 4. Cari Data Berdasarkan Nilai                " <<
endl;
        cout << " 5. Tampilkan Semua Data                       " <<
endl;
        cout << " 6. Keluar                                     " <<
endl;
        cout  <<  "========================================"   <<
endl;
        cout << endl;
        cout << "Masukan pilihan anda : ";
        cin >> choice;
        cin.ignore();
        cout << endl;

        switch (choice)
        {
        case 1:
            cout << "Menu Input Data" << endl;
            cout << " Masukan Nama Mahasiswa : ";
            getline(cin, nama);
            cout << " Masukan Nim Mahasiswa  : ";
            cin >> nim;
            cout << " Masukan Nilai          : ";
            cin >> nilai;
            mahasiswaTable.addData(nama, nim, nilai);
            cout << endl;
            break;
```

```
        case 2:
            cout << "Hapus Data Mahasiswa" << endl;
            cout << " Masukan NIM Mahasiswa : ";
            cin >> nim;
            mahasiswaTable.removeData(nim);
            cout << endl;
            break;

        case 3:
            cout << "Menu Mencari Data Mahasiswa Berdasarkan NIM"
<< endl;
            cout << " Masukan NIM Mahasiswa : ";
            cin >> nim;
            mahasiswaTable.findDatabyNim(nim);
            cout << endl;
            break;

        case 4:
            cout << "Menu Mencari Data Mahasiswa Berdasarkan
Nilai" << endl;
            cout << " Data mahasiswa dengan nilai 80-90 : " <<
endl;
            mahasiswaTable.findDataByScore(80, 90);
            break;

        case 5:
            cout << "Tampilkan Semua Data" << endl;
            mahasiswaTable.printData();
            break;

        case 6:
            cout << "Anda Telah Keluar Dari Program" << endl;
            break;
```

```
        default:
            break;
        }

    } while (choice != 6);
    return 0;
}
```

Screenshot Program

1. **Menambah Data**



2. **Hapus Data**

```
Data Mahasiswa
 1. Tambah Data
 2. Hapus Data
 3. Cari Data Berdasarkan NIM
 4. Cari Data Berdasarkan Nilai
 5. Tampilkan Semua Data
 6. Keluar
=======================================

Masukan pilihan anda : 2

Hapus Data Mahasiswa
 Masukan NIM Mahasiswa : 22011289
Data Mahasiswa : Ganang ( 22011289 ) telah dihapus
```

3. **Mencari data dengan NIM**

```
 Data Mahasiswa
  1. Tambah Data
  2. Hapus Data
  3. Cari Data Berdasarkan NIM
  4. Cari Data Berdasarkan Nilai
  5. Tampilkan Semua Data
  6. Keluar
 =======================================

 Masukan pilihan anda : 3

 Menu Mencari Data Mahasiswa Berdasarkan NIM
  Masukan NIM Mahasiswa : 2311102029
 Nama   : Ilhan Sahal Mansiz
 NIM    : 2311102029
 Nilai  : 90
```

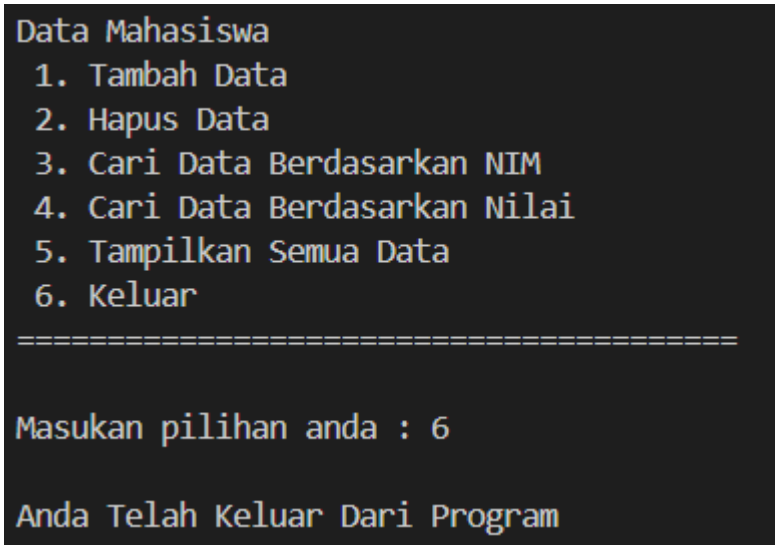
4. **Mencari data berdasarkan nilai 80-90**

```
Data Mahasiswa
 1. Tambah Data
 2. Hapus Data
 3. Cari Data Berdasarkan NIM
 4. Cari Data Berdasarkan Nilai
 5. Tampilkan Semua Data
 6. Keluar
=======================================

Masukan pilihan anda : 4

Menu Mencari Data Mahasiswa Berdasarkan Nilai
 Data mahasiswa dengan nilai 80-90 :
Nama   : Ilhan Sahal Mansiz
NIM    : 2311102029
Nilai  : 90
Nama   : Umam
NIM    : 22239049
Nilai  : 81
Nama   : Angga
NIM    : 223110904
Nilai  : 89
```

5. **Tampilkan Semua Data**

```
Data Mahasiswa
 1. Tambah Data
 2. Hapus Data
 3. Cari Data Berdasarkan NIM
 4. Cari Data Berdasarkan Nilai
 5. Tampilkan Semua Data
 6. Keluar
=======================================

Masukan pilihan anda : 5

Tampilkan Semua Data
Nama   : Ilhan Sahal Mansiz
NIM    : 2311102029
Nilai  : 90

Nama   : Umam
NIM    : 22239049
Nilai  : 81

Nama   : Angga
NIM    : 223110904
Nilai  : 89

Nama   : Shasa
NIM    : 233112930
Nilai  : 79
```

6. **Keluar**



Deskripsi Program

Program di atas adalah implementasi sederhana dari struktur data hash table dalam bahasa C++, yang digunakan untuk menyimpan data mahasiswa berupa nama, NIM, dan nilai. Program ini memungkinkan pengguna untuk melakukan beberapa operasi, termasuk penambahan data mahasiswa, penghapusan data berdasarkan NIM, pencarian data berdasarkan NIM, pencarian data berdasarkan rentang nilai, dan penampilan semua data yang tersimpan.

Setiap mahasiswa disimpan dalam vektor yang merupakan bagian dari array dua dimensi, di mana array luar digunakan untuk menampung vektor-vektor yang berisi data mahasiswa, dengan pemetaan dilakukan menggunakan fungsi hash sederhana berdasarkan NIM mahasiswa. Selain itu, program menyediakan antarmuka pengguna yang memungkinkan pengguna untuk memilih operasi yang diinginkan melalui konsol. Ini membuatnya menjadi alat yang berguna untuk mengelola informasi mahasiswa secara efisien.

BAB IV

KESIMPULAN

Modul 5 membahas konsep dan implementasi dari struktur data hash table. Hash table merupakan struktur data yang efisien digunakan untuk menyimpan dan mengakses data dengan cepat berdasarkan kunci tertentu. Dalam modul ini, kita mempelajari tentang konsep dasar hash table, seperti fungsi hash, penanganan tabrakan, dan penggunaan hash table dalam konteks aplikasi nyata. Dengan menggunakan teknik hashing, hash table dapat memberikan kinerja yang sangat baik dalam pencarian, penambahan, dan penghapusan data, bahkan dengan ukuran data yang besar. Dengan demikian, pemahaman tentang hash table menjadi penting dalam pengembangan aplikasi yang membutuhkan operasi cepat dan efisien pada kumpulan data yang besar.

BAB V

REFERENSI

- https://algorit.ma/blog/hash-table-adalah-2022/
- https://www.digitalocean.com/community/tutorials/hash-table-in-c-plus-plus
- https://aozturk.medium.com/simple-hash-map-hash-table-implementation-in-c-931965904250