

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL V
“HASH TABLE”**



Disusun Oleh :

Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas : IF 11 B

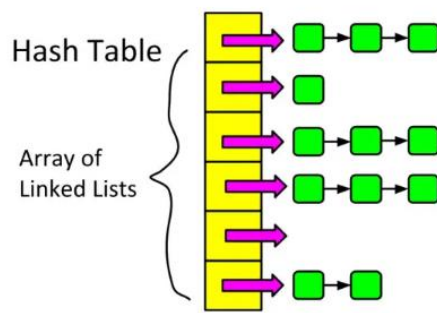
DOSEN:

WAHYU ANDI SAPUTRA, S.Pd. , M.Eng

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

A. Dasar Teori

Tabel hash di C/C++ adalah struktur data yang memetakan kunci ke nilai. Tabel hash menggunakan fungsi hash untuk menghitung indeks kunci. Kita dapat menyimpan nilai di tempat yang tepat berdasarkan indeks tabel hash. Keuntungan menggunakan tabel hash adalah waktu akses yang sangat cepat. Biasanya, kompleksitas waktu (kompleksitas waktu diamortisasi) adalah $O(1)$ untuk waktu akses yang konstan. Jika dua kunci berbeda memperoleh indeks yang sama, struktur data (bucket) yang berbeda harus digunakan untuk memperhitungkan konflik ini. Jika kita memilih fungsi hash yang sangat bagus, kemungkinan tabrakan dapat diabaikan.



Cara kerja table hash :

- Kunci dan Nilai: Setiap entri dalam tabel hash terdiri dari kunci dan nilai. Kuncinya bertindak seperti pengidentifikasi unik, dan nilainya adalah data yang Anda simpan.
- Fungsi Hash: Saat menambahkan entri ke tabel hash, fungsi khusus yang disebut fungsi hash digunakan. Fungsi ini mengambil kunci sebagai masukan dan mengubahnya menjadi angka yang disebut kode hash.
- Array dan Bucket: Tabel hash menggunakan slot atau array bucket untuk menyimpan pasangan nilai kunci. Kode hash digunakan sebagai indeks untuk menentukan keranjang spesifik tempat pasangan nilai kunci disimpan.

Keunggulan table hash :

- Pencarian cepat: Keuntungan utama tabel hash adalah waktu pencarian yang cepat. Kode hash memberikan akses cepat ke nilai yang diinginkan.

terkait dengan suatu kunci. Ini jauh lebih cepat daripada mencari daftar atau array di mana Anda harus membandingkan kunci setiap elemen.

- Efisiensi penyisipan dan penghapusan: Penyisipan dan penghapusan elemen dalam tabel hash juga rata-rata efisien, karena mereka juga menggunakan fungsi hash akses langsung.

Operasi Tabel Hash :

- Sisipkan: Masukkan data baru ke dalam tabel hash dengan memanggil fungsi hash untuk menentukan lokasi bucket yang tepat dan menambahkan data ke bucket.
- Hapus: Hapus data dari tabel hash dengan mencari data menggunakan fungsi hash dan menghapus data dari keranjang yang sesuai.
- Pencarian: Temukan data dalam tabel hash dengan memasukkan kunci ke dalam fungsi hash untuk menentukan lokasi keranjang, lalu mencari data di keranjang yang sesuai.
- Update: Update data pada tabel hash dengan mencari data menggunakan fungsi hash dan mengupdate data yang ditemukan.
- Transversal: Telusuri seluruh tabel hash dan proses semua data dalam tabel.

Collision 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 utama yang sering digunakan untuk menyelesaikan masalah tabrakan:

1. Chaining: Metode: Chaining menyimpan semua data yang terkait dengan slot hash yang sama dalam struktur data tertaut yang disebut rantai. Setiap elemen dalam rantai berisi kunci dan nilainya.
 - Keuntungan : Rantai mudah diterapkan dan dapat menangani tabrakan dalam jumlah besar.

- Kerugian : Menggunakan rantai dapat meningkatkan waktu pencarian, terutama jika rantainya panjang. Selain itu, menyimpan struktur data tertaut memerlukan konsumsi memori tambahan.
2. Open addressing: Metode: Dengan open addressing, slot hash yang sudah terisi ditandai sebagai ``penuh" dan slot lain dicari untuk menyimpan data baru. Ada beberapa strategi open addressing, termasuk:
- Pemeriksaan linier: Temukan slot kosong berikutnya dalam urutan .
 - Pemeriksaan sekunder: Temukan slot kosong menggunakan fungsi hash sekunder. Hash Ganda: Periksa slot kosong menggunakan dua fungsi hash.
- Keuntungan : Open addressing umumnya lebih cepat daripada rantai untuk pengambilan data.
 - Kekurangan: Open addressing dapat mengakibatkan cluster dimana beberapa slot hash berturut-turut terisi. Hal ini dapat memperlambat pencarian dan meningkatkan penggunaan memori.

Memilih Teknik yang Tepat

Teknik optimal untuk menangani tabrakan bergantung pada kebutuhan aplikasi kita. Secara umum, rantai cocok untuk aplikasi yang jarang terjadi tabrakan, dan open addressing cocok untuk aplikasi yang sering terjadi tabrakan. Faktor lain yang perlu dipertimbangkan mencakup kompleksitas implementasi dan dampak kinerja. Selain kedua teknik di atas, ada beberapa teknik yang kurang umum dalam menangani konflik. Seperti cuckoo hashing dan bucket sort

Penting untuk dicatat bahwa tidak ada solusi sempurna untuk masalah tabrakan. Setiap teknik memiliki kelebihan dan kekurangannya masing-masing, dan pemilihan teknik terbaik bergantung pada kebutuhan spesifik aplikasi Anda.

B. 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;
}

```

Output :

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ praktikum5hashtableG1.cpp -o praktikum5hashtableG1 } ; if ($?) { .\prak
tikum5hashtableG1 }
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS D:\strukdat>

```

```

Nama Fitr
File Edit View
Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
Ln 1, Col 25 | 60 characters | 100% | Window UTF-8

```

Deskripsi program :

Kode di atas adalah implementasi tabel hash menggunakan modulo fungsi hash sederhana `MAX_SIZE(10)`.

- fungsi `hash_func` yang akan digunakan untuk mengubah kunci menjadi nilai hash menggunakan XOR.
- struktur data `node` merepresentasikan node dalam tabel hash. Node ini memiliki tiga atribut: Kunci, Nilai, dan Berikutnya.
- buat kelas `HashTable` untuk mewakili tabel hash. Kelas ini memiliki beberapa metode:
 - konstruktor dan destruktur untuk mengalokasikan dan mengosongkan memori.
 - `insert` : Metode untuk menambahkan rincian kontak ke tabel hash.
 - `get` : Metode untuk mencari rincian kontak berdasarkan kunci.
 - `remove` : Metode untuk menghapus data kontak dari tabel hash.
 - `traverse` : Metode untuk menampilkan isi tabel hash.

Selanjutnya, ada fungsi utama (`main`) yang menjalankan program. Fungsi ini membuat objek `HashTable` dan memasukkan data kontak ke dalamnya.

Program ini dapat melakukan berbagai operasi pada data kontak seperti, mencari data kontak berdasarkan kunci, menghapus data kontak, dan mencetak isi tabel hash. Untuk mengimplementasikan setiap operasi, panggil metode objek HashTable yang sesuai. Kode ini menampilkan beberapa informasi kontak, menghapus kontak dengan kunci 4, dan mencetak isi tabel hash.

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 << "]\n";
            }
        }
        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;
}

```

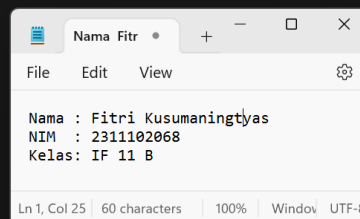
Output :

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prakShashtableG2.cpp -o prakShashtableG2 } ; if ($?) { .\prakShashtableG2 }
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:\strukdat>

```



Deskripsi program :

Kode di atas merupakan implementasi tabel hash untuk menyimpan data kontak.

- kelas HashNode untuk mewakili sebuah node di tabel hash. Node ini memiliki dua atribut: name dan phone_number.
- kelas HashMap dibuat untuk mengimplementasikan tabel hash. Kelas ini memiliki beberapa metode.
 - hashFunc: Fungsi hash yang mengubah nama menjadi nilai hash menggunakan XOR.
 - insert : Metode untuk menambahkan rincian data kontak ke tabel hash.
 - remove : Metode untuk menghapus data kontak dari tabel hash.
 - searchByName: Metode untuk mencari informasi kontak berdasarkan nama.
 - print: Metode untuk mencetak isi tabel hash.
- Selanjutnya, ada fungsi utama (main) yang menjalankan program. Di dalam fungsi ini, buat objek HashMap dan masukkan detail kontak ke dalamnya. Program melakukan beberapa operasi pada data kontak: menemukan data kontak berdasarkan nama, menghapus data kontak, dan mencetak konten tabel hash. Untuk mengimplementasikan setiap operasi, panggil metode yang sesuai dari objek HashMap. Kode ini menampilkan beberapa informasi kontak, menghapus kontak bernama "Mistah", dan mencetak isi tabel hash.

C. Unguided

1. 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 << "];"
            }
        }
        cout << endl;
    }
}
};

int main()
{
    HashMap data;
    string NIM;
    int nilai_mhs;
    while (true)
    {
        int menu;
        cout << "\n===== MENU =====" << endl;
        cout << "1. Input data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Mencari data berdasarkan NIM" << endl;
        cout << "4. Mencari data berdasarkan nilai" << endl;
        cout << "5. Tampilkan 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 a, b;
            cout << "Masukkan rentang nilai minimal : ";
            cin >> a;
            cout << "Masukkan rentang nilai maksimal : ";
            cin >> b;
            data.findNimByScore(a, b);
            break;
        case 5:
            data.print();
            break;
        case 6:
            cout << "Anda telah keluar dari program";
            return 0;
        default:
            cout << "Menu tidak tersedia!" << endl;
            break;
    }
}
return 0;
}

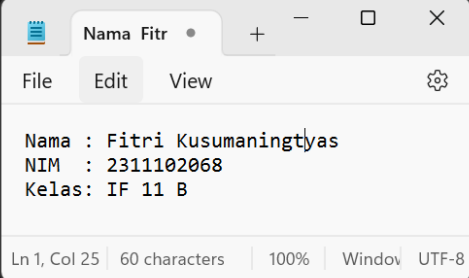
```


Output :

- a. Setiap mahasiswa memiliki nim dan nilai

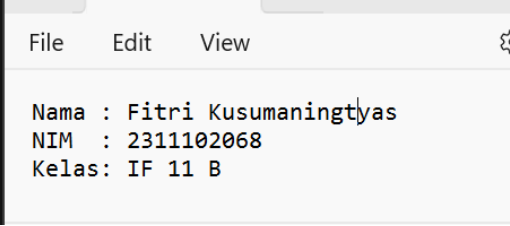
```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prakShashtableug.cpp -o prakShashtableug } ;

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2311102068
Masukkan nilai : 98
```



- b. Program memiliki tampilah pilihan menu berisi poin C

```
===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
```



- c. Implementasikan fungsi untuk menambah data baru, menghapus data, mencari data berdasarkan NIM, dan mencari data berdasarkan rantang nilai (80-90)

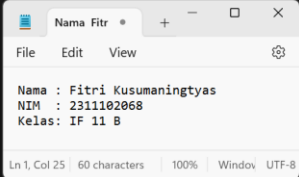
- Input data

```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prakShashtableug.cpp -o prakShashtableug } ; if ($?) { .\prakShashtableug }

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2311102068
Masukkan nilai : 98

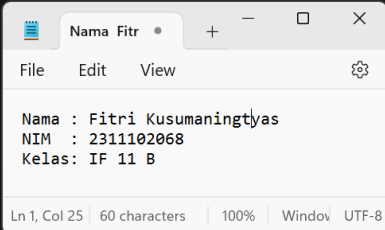
===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2234567821
Masukkan nilai : 82

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2215679903
Masukkan nilai : 89
```

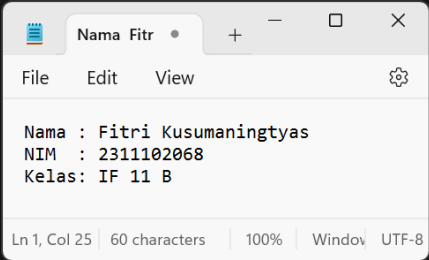


```
===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2300923477
Masukkan nilai : 76

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 1
Masukkan NIM : 2199865432
Masukkan nilai : 85
```



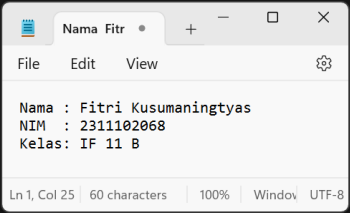
```
===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 5
0: [2300923477, 76]
1: [2199865432, 85]
2:
3: [2234567821, 82]
4:
5:
6:
7: [2215679903, 89]
8:
9: [2311102068, 98]
10:
```



- Delete data

```
===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 2
Masukkan NIM yang akan dihapus : 2234567821
Data berhasil dihapus

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkkkan pilihan : 5
0: [2300923477, 76]
1: [2199865432, 85]
2:
3:
4:
5:
6:
7: [2215679903, 89]
8:
9: [2311102068, 98]
10:
```



- Mencari data berdasarkan NIM

The terminal window shows the following menu and user input:

```

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkan pilihan : 3
Masukkan NIM yang akan dicari : 2311102068
NIM 2311102068 memiliki nilai 98
  
```

The Notepad window, titled 'Nama Fitr', displays the following data:

```

Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
  
```

- Mencari data berdasarkan rentang nilai (80-90)

The terminal window shows the following menu and user input:

```

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkan pilihan : 4
Masukkan rentang nilai minimal : 80
Masukkan rentang nilai maksimal : 90
2199865432 memiliki nilai 85
2215679903 memiliki nilai 89
  
```

The Notepad window, titled 'Nama Fitr', displays the same data as in the previous screenshot:

```

Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
  
```

- Tampilan

The terminal window shows the following menu and user input:

```

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkan pilihan : 5
0: [2300923477, 76]
1: [2199865432, 85]
2:
3:
4:
5:
6:
7: [2215679903, 89]
8:
9: [2311102068, 98]
10:

===== MENU =====
1. Input data
2. Delete data
3. Mencari data berdasarkan NIM
4. Mencari data berdasarkan nilai
5. Tampilkan data
6. Exit
Masukkan pilihan : 6
Anda telah keluar dari program
PS D:\strukdat>
  
```

The Notepad window, titled 'Nama Fitr', displays the same data as in the previous screenshots:

```

Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
  
```

Deskripsi program :

Kode diatas merupakan implementasi tabel hash untuk menyimpan data siswa berdasarkan NIM dan nilai siswa.

- kelas HashNode berfungsi untuk mewakili sebuah node di tabel hash. Node ini memiliki dua atribut: Nim dan Nilai.

- kelas HashMap dibuat untuk merepresentasikan tabel hash. Kelas ini memiliki metode berikut:
 - hashFunc: Fungsi hash yang menggunakan XOR untuk mengubah NIM menjadi nilai hash.
 - Insert : Metode untuk menambahkan data siswa ke tabel hash.
 - Hapus : Metode untuk menghapus data mahasiswa dari tabel hash.
 - Search : Cara pencarian data siswa berdasarkan NIM.
 - findNimByScore: Metode untuk mencari data siswa berdasarkan rentang nilai
 - print: Metode untuk mencetak isi tabel hash.
- fungsi utama (main) yang menjalankan program. Di dalam fungsi ini, kita membuat objek HashMap dan memulai perulangan while yang menampilkan menu opsi kepada pengguna. Untuk setiap opsi menu, lakukan tindakan berikut untuk opsi tersebut. Terdapat pilihan menu Memasukkan data, menghapus data, mengambil data berdasarkan NIM atau nilai, menampilkan data, atau keluar dari program. Untuk mengimplementasikan setiap operasi, panggil metode yang sesuai dari objek HashMap. Kode ini akan terus berjalan hingga pengguna keluar dari program.

D. Kesimpulan

Hash Table (juga dikenal sebagai hash map atau dictionary) adalah struktur data dasar yang ditandai dengan pencarian, penyisipan, dan penghapusan key-value yang efisien dalam data. Hash Table menggunakan fungsi hash untuk memetakan kunci unik ke indeks bucket dalam array internal. Hal ini memungkinkan pengoperasian dengan waktu rata-rata yang konstan dalam kondisi ideal (ketika Hash Table tidak kelebihan beban dan fungsi hash mendistribusikan kunci secara seragam di seluruh bucket).

E. Referensi jurnal

Tapia-Fernández, S., García-García, D., & García-Hernandez, P. (2022). Key Concepts, Weakness and Benchmark on Hash Table Data Structures. *Algorithms*, 15(3), 100.

Goodrich, M. T., Tamassia, R., & Mount, D. M. (2011). *Data structures and algorithms in C++*. John Wiley & Sons.

Barnat, J., Ročkai, P., Štill, V., & Weiser, J. (2015, August). Fast, dynamically-sized concurrent hash table. In *International SPIN Workshop on Model Checking of Software* (pp. 49-65). Cham: Springer International Publishing.

Ročkai, P. (2019, October). Model Checking in a Development Workflow: A Study on a Concurrent C++ Hash Table. In *International Symposium on Formal Methods* (pp. 46-60). Cham: Springer International Publishing.

Barreiro, J., Fraley, R., & Musser, D. (1995). Hash Tables for the Standard Template Library. *Rensselaer Polytechnic Institute and Hewlett Packard Laboratories, document*, (X3J16/94), 0218.