

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL V  
HASH TABLE**



**Disusun oleh:**  
**Alfin Ilham Berlianto**  
**2311102047**

**Dosen pengampu:**  
Wahyu Andi Saputra, S.Pd., M.Eng.

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

## MODUL 5

### HASH TABLE

#### A. Tujuan

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

#### B. Dasar Teori

##### Hash Table

Apasih itu hash table? Hash table merupakan struktur data yang secara asosiatif menyimpan data. Dalam hal ini, data disimpan dalam format array, di mana setiap nilai data memiliki nilai indeks uniknya sendiri. Akses data akan menjadi sangat cepat jika Anda mengetahui indeks dari data yang diinginkan.

Dengan demikian, hash table menjadi struktur data di mana operasi penyisipan dan pencarian data terjadi sangat cepat terlepas dari ukuran data tersebut. Hash table menggunakan array sebagai media penyimpanan dan tekniknya untuk menghasilkan indeks suatu elemen yang dimasukkan atau ditempatkan.

##### Fungsi Hash Table

Fungsi utamanya pada data adalah mempercepat proses akses data. Hal ini berkaitan dengan peningkatan data dalam jumlah besar yang diproses oleh jaringan data global dan lokal. Hash table adalah solusi untuk membuat proses akses data lebih cepat dan memastikan bahwa data dapat dipertukarkan dengan aman.

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

##### Teknik-Teknik Hash Table

###### 1. Hashing

Hashing merupakan sebuah proses mengganti kunci yang diberikan atau string karakter menjadi nilai lain. Penggunaan hashing paling populer adalah pada hash table. Hash table menyimpan pasangan kunci dan nilai dalam daftar yang dapat diakses melalui indeksinya. Karena pasangan kunci dan nilai tidak terbatas, maka fungsinya akan memetakan kunci ke ukuran tabel dan kemudian nilainya menjadi indeks untuk elemen tertentu.

## 2. Linear Probing

Linear probing merupakan skema dalam pemrograman komputer untuk menyelesaikan collision pada hash table. Dalam skema ini, setiap sel dari hash table menyimpan satu pasangan kunci-nilai. Saat fungsi hash menyebabkan collision dengan memetakan kunci baru ke sel hash table yang sudah ditempati oleh kunci lain, maka linear probing akan mencari tabel untuk lokasi bebas terdekat dan menyisipkan kunci baru.

Pencarian dilakukan dengan cara yang sama, yaitu dengan mencari tabel secara berurutan, mulai dari posisi yang diberikan oleh fungsi hash, hingga menemukan sel dengan kunci yang cocok atau sel kosong. Hash table adalah struktur data non trivial yang paling umum digunakan. Linear probing dapat memberikan kinerja tinggi karena lokasi referensi yang baik, namun lebih sensitif terhadap kualitas fungsi hash daripada beberapa skema resolusi collision lainnya.

## C. 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;
}

```

```

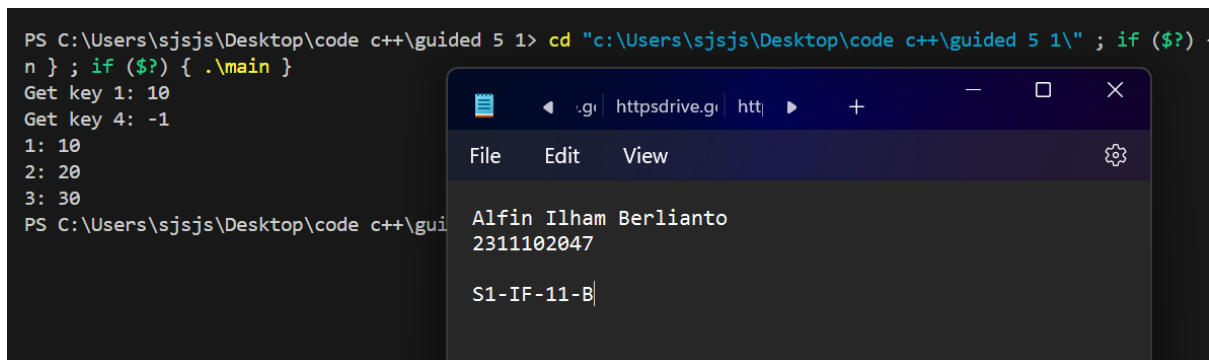
// Deletion
ht.remove(4);

// Traversal
ht.traverse();

return 0;
}

```

## SCREENSHOUT OUTPUT



The screenshot shows a Windows command prompt window with the following output:

```

PS C:\Users\sjsjs\Desktop\code c++\guided 5 1> cd "c:\Users\sjsjs\Desktop\code c++\guided 5 1\" ; if ($?) {
n } ; if ($?) { .\main }
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\Users\sjsjs\Desktop\code c++\gui

```

Overlaid on the command prompt is a web browser window displaying a form with the following text:

Alfin Ilham Berlianto  
2311102047  
S1-IF-11-B

## DESKRIPSI PROGRAM

Dalam program ini membuat program hash table, yaitu fungsi hash table itu sendiri ialah digunakan untuk menyimpan nilai yang berpasangan. Setiap elemen dalam hash table diakses menggunakan key atau kunci yang ditentukan. Di dalam program ini terdapat method Insert, Get, Remove dan Traversal. Yang mana insert ini untuk menambah data, Get untuk mengambil data, remove untuk menghapus data, dan traversal untuk menampilkan data secara keseluruhan.

## 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()

```

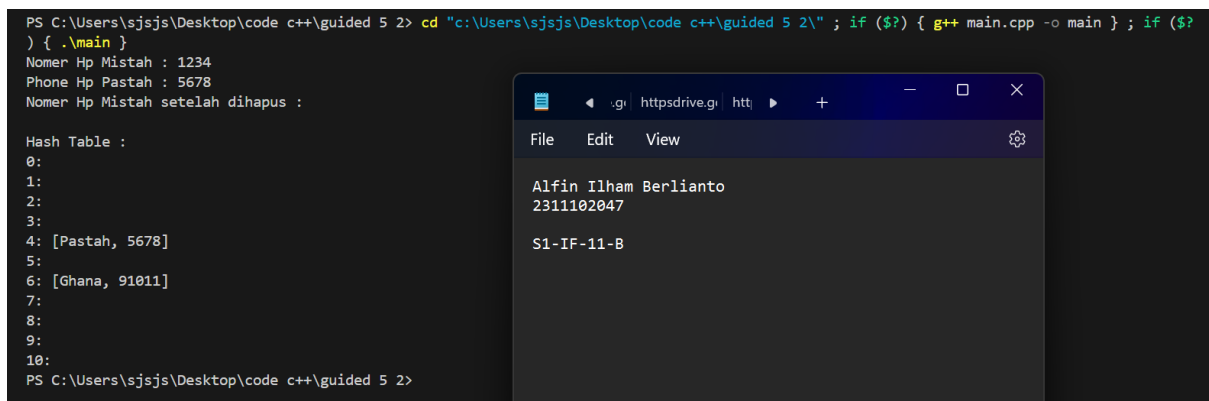


```

{
    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;
}

```

## SCREENSHOT OUTPUT



```

PS C:\Users\sjsjs\Desktop\code c++\guided 5 2> cd "c:\Users\sjsjs\Desktop\code c++\guided 5 2\" ; if ($?) { g++ main.cpp -o main } ; if ($?)
) { .\main }
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\sjsjs\Desktop\code c++\guided 5 2>

```

## DESKRIPSI PROGRAM

Pada program ini sama seperti sebelumnya membuat hash table, akan tetapi pada program ini memiliki perbedaan dari sebelumnya, yang mana jika sebelumnya menggubakan array untuk menampung data, pada program ini menggunakan vector. Vector sebenarnya juga hampir sama seperti array, Cuma yang membedakan yaitu kalo vector dia bersifat dinamis, lalu data vector berbentuk object dengan nama hash node. Di program ini terdapat 2 data hash node yaitu nama dan nomer HP. Hash pada program ini dijalankan dengan menjumlahkan nilai dari key lalu dibagi dengan ukuran hash table dan diambil sisa hasil bagiannya. HashFunc berfungsi untuk menghasilkan nilai hash dari kunci atau key, insert digunakan untuk menambahkan data baru, remove digunakan untuk menghapus data dengan cara mencari berdasarkan nama, searchByName digunakan untuk mencari nomor HP berdasarkan nama. Proses pencarian dilakukan dengan menggunakan algoritma sequential search, yaitu dengan mengecek satu per satu data secara berurutan dari awal hingga akhir atau hingga data yang sesuai ditemukan. Jika tidak ditemukan, maka program akan mengembalikan string kosong. Yang terakhir ada print, ini digunakan untuk menampilkan data dari data yang telah dibuat menggunakan menu-menu sebelumnya.

## D. UNGUIDED

### SOURCE CODE

```
#include <iostream>
#include <string>
#include <vector>
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;
        }
        else
        {
            cout << "Tidak Ditemukan" << endl;
        }
    }
}

int searchByNim(string nim)
{
    int hash_val = hashFunc(nim);
    for (auto node : table[hash_val])
    {
        if (node->nim == nim)
        {
            // return node->nilai;
            cout << "\nMahasiswa dengan NIM " << node->nim << " memiliki nilai " <<
node->nilai << endl;
        }
    }
}

```

```

    }
    return 0;
}
int searchByNilai(int minNilai, int maxNilai)
{
    for (const auto &bucket : table)
    {
        for (auto node : bucket)
        {
            if (node->nilai >= minNilai && node->nilai <= maxNilai)
            {
                cout << "[ NIM : " << node->nim << ", NILAI : " << node->nilai << " ]" <<
endl;
            }
        }
    }
    return 0;
}

void print()
{
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        cout << i << ": ";
        for (auto pair : table[i])
        {
            if (pair != nullptr)
            {
                cout << "[ NIM : " << pair->nim << ", NILAI : " << pair->nilai << " ]";
            }
        }
        cout << endl;
    }
}
};
int main()
{
    HashMap mahasiswa_map;
    bool menu = true;
    int choice;
    do
    {
        cout << " Data Mahasiswa " << endl;
        mahasiswa_map.print();
        cout << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Cari berdasarkan NIM" << endl;
        cout << "4. Cari berdasarkan Nilai" << endl;
    }
}

```

```

cout << "0. Keluar" << endl;
cout << "Pilihan Anda: ";
cin >> choice;
switch (choice)
{
case 1:
    cout << "Masukkan NIM: ";
    cin >> nim;
    cout << "Masukkan Nilai: ";
    cin >> nilai;

    mahasiswa_map.insert(nim, nilai);
    break;
case 2:
    cout << "Masukkan NIM: ";
    cin >> nim;
    mahasiswa_map.remove(nim);
    break;
case 3:
    cout << "Masukkan NIM: ";
    cin >> nim;
    mahasiswa_map.searchByNim(nim);

    break;
case 4:
    int maxNilai, minNilai;
    cout << "Masukkan Nilai Tertinggi: ";
    cin >> maxNilai;
    cout << "Masukkan Nilai Terendah: ";
    cin >> minNilai;
    mahasiswa_map.searchByNilai(minNilai, maxNilai);
    break;
case 0:
    menu = false;
    break;

default:
    break;
}

} while (menu == true);
return 0;
}

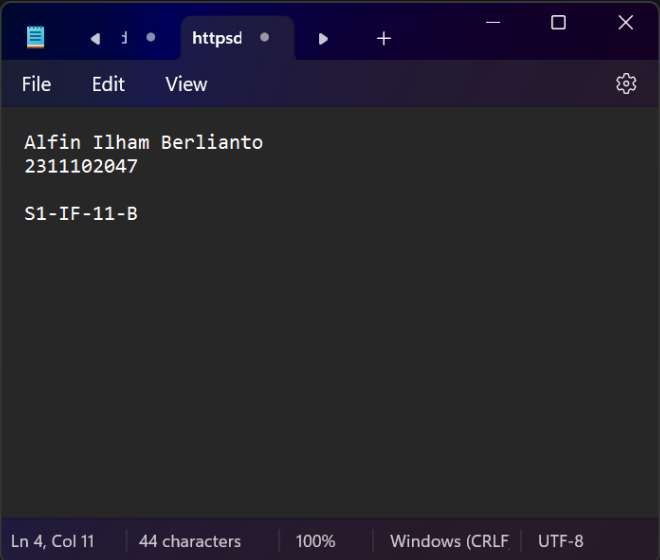
```

## SCREENSHOT OUTPUT

### Tambah data

```
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 1
Masukkan NIM: 2311102047
Masukkan Nilai: 98
Data Mahasiswa
0:
1:
2:
3:
4:
5:
6: [ NIM : 2311102047, NILAI : 98 ]
7:
8:
9:
10:

1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 
```

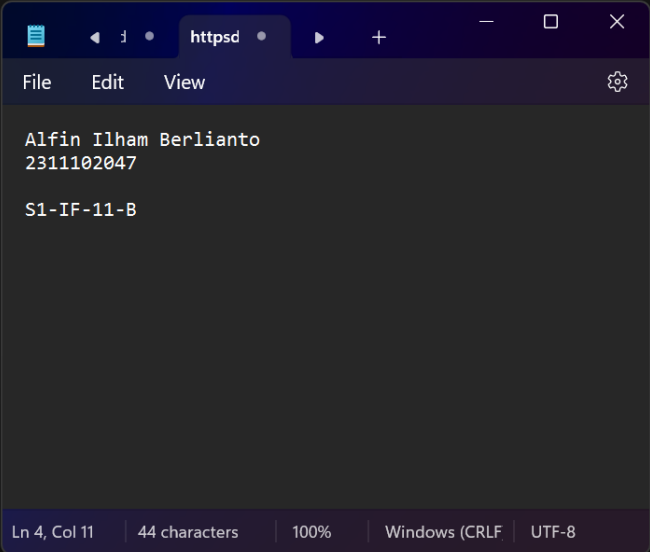


The screenshot shows a web browser window with a single tab titled 'httpsd'. The browser's address bar is empty. The page content displays a form with three lines of text: 'Alfin Ilham Berlianto', '2311102047', and 'S1-IF-11-B'. The browser's status bar at the bottom indicates 'Ln 4, Col 11', '44 characters', '100%', 'Windows (CRLF)', and 'UTF-8'.

### Hapus Data

```
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 2
Masukkan NIM: 2311102045
Data Mahasiswa
0:
1:
2:
3:
4:
5:
6: [ NIM : 2311102047, NILAI : 98 ]
7: [ NIM : 2311102048, NILAI : 55 ]
8:
9:
10:

1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 
```



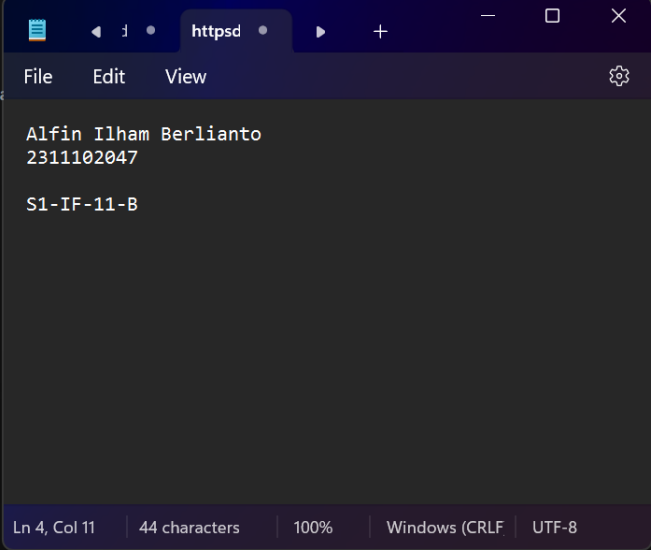
The screenshot shows a web browser window with a single tab titled 'httpsd'. The browser's address bar is empty. The page content displays a form with three lines of text: 'Alfin Ilham Berlianto', '2311102047', and 'S1-IF-11-B'. The browser's status bar at the bottom indicates 'Ln 4, Col 11', '44 characters', '100%', 'Windows (CRLF)', and 'UTF-8'.

## Cari dengan NIM

```
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 3
Masukkan NIM: 2311102047

Mahasiswa dengan NIM 2311102047 memiliki nilai:
Data Mahasiswa
0:
1:
2:
3:
4:
5:
6: [ NIM : 2311102047, NILAI : 98 ]
7: [ NIM : 2311102048, NILAI : 55 ]
8:
9:
10:

1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 
```



The screenshot shows a web browser window with a single tab titled 'httpsd'. The browser's address bar is empty. The page content displays the following information:

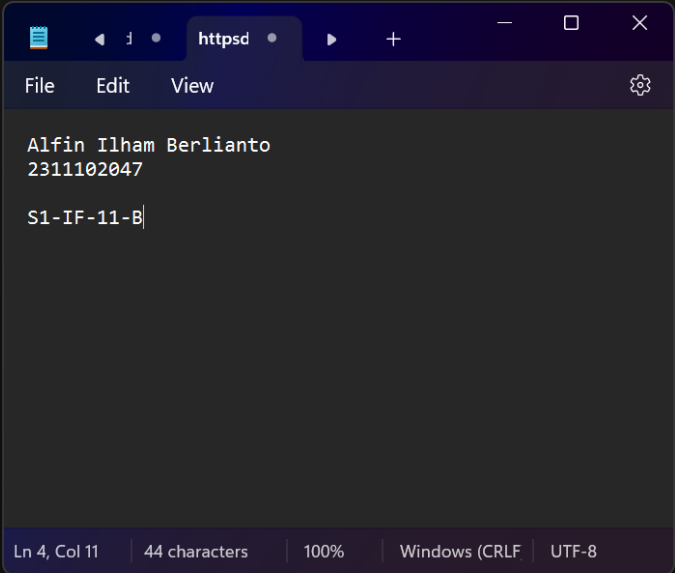
- Alfin Ilham Berlianto
- 2311102047
- S1-IF-11-B

The browser's status bar at the bottom indicates the cursor is at line 4, column 11, with 44 characters, 100% zoom, and Windows (CRLF) line endings in UTF-8 encoding.

## Cari dengan Nilai

```
1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 4
Masukkan Nilai Tertinggi: 98
Masukkan Nilai Terendah: 55
[ NIM : 2311102047, NILAI : 98 ]
[ NIM : 2311102048, NILAI : 55 ]
Data Mahasiswa
0:
1:
2:
3:
4:
5:
6: [ NIM : 2311102047, NILAI : 98 ]
7: [ NIM : 2311102048, NILAI : 55 ]
8:
9:
10:

1. Tambah Data
2. Hapus Data
3. Cari berdasarkan NIM
4. Cari berdasarkan Nilai
0. Keluar
Pilihan Anda: 
```



The screenshot shows a web browser window with a single tab titled 'httpsd'. The browser's address bar is empty. The page content displays the following information:

- Alfin Ilham Berlianto
- 2311102047
- S1-IF-11-B

The browser's status bar at the bottom indicates the cursor is at line 4, column 11, with 44 characters, 100% zoom, and Windows (CRLF) line endings in UTF-8 encoding.

## DESKRIPSI PROGRAM

Pada program ini membuat program hash table yang nilai inputannya diinputkan oleh user, di dalam program ini menyimpan data nilai mahasiswa dan NIM. Program ini mengimplementasikan struktur data hash table untuk menyimpan dan mengelola data mahasiswa. Pertama program mendefinisikan class HashNode, setiap node memiliki dua atribut yaitu NIM untuk menyimpan NIM Mahasiswa sebagai key atau kunci, dan nilai untuk menyimpan nilai mahasiswa. Kemudian terdapat class HashMap, yang merupakan class utama yang merepresentasikan hash table, class ini memiliki array vector table yang digunakan untuk menyimpan node-node hash table. Fungsi hashFunc digunakan untuk menghitung nilai hash dari NIM mahasiswa. Selanjutnya, terdapat fungsi insert untuk menambahkan data mahasiswa ke dalam hash table. Jika terjadi collision, data akan disimpan dalam bentuk chaining di dalam vector pada indeks yang sama dalam array table. Fungsi remove digunakan untuk menghapus data mahasiswa berdasarkan NIM, fungsi searchByNim digunakan untuk mencari data berdasarkan NIM, dan fungsi searchByNilai digunakan untuk mencari data berdasarkan nilai tertinggi mahasiswa dan nilai terendah mahasiswa.



## KESIMPULAN

Hash table adalah salah satu struktur data yang paling penting dan sering digunakan dalam pemrograman computer. Dengan menggunakan hash function untuk menentukan kunci ke indeks dalam tabel hash, hash table memungkinkan penyimpanan data yang efisien dan pencarian data yang cepat. Hal ini terutama disebabkan oleh waktu eksekusi konstan yang dapat dicapai dalam operasi dasar seperti penyisipan, pengambilan, dan penghapusan data. Namun, meskipun hash table menawarkan kinerja yang sangat baik dalam kasus rata-rata, kinerja terburuknya dapat menjadi masalah jika hash function tidak seimbang atau jika collision terjadi selalu sering. Oleh karena itu, perencanaan yang cermat dalam pemilihan ukuran tabel hash dan implementasi hash function yang efisien sangat penting untuk memastikan kinerja yang optimal dari hash tab

## DAFTAR PUSTAKA

1. Algoritma Data Science School, (2022, 22 Maret) Apa Itu Hash Table dan Bagaimana Penggunaannya?. Diakses pada 14 Mei 2024, dari <https://algorit.ma/blog/hash-table-adalah-2022/>
- 2, IR-NLP Lab, (2020, 21 November) Pengantar Hash Table. Diakses pada 14 Mei 2024, dari <https://ir.cs.ui.ac.id/alfan/sda/hashtables/hashtable.pdf>