

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



Disusun Oleh:

Arvan Murbiyanto

2311102074

Dosen

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

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

2024

Modul V

A. Tujuan

1. Mahasiswa dapat memahami konsep **HASH TABLE**
2. Mahasiswa mampu mengimplementasikan HASH TABLE

B. Dasar Teori

Hash Table adalah sebuah struktur data yang terdiri atas sebuah tabel dan fungsi yang bertujuan untuk memetakan nilai kunci yang unik untuk setiap record (baris) menjadi angka (hash) lokasi record tersebut dalam sebuah tabel.

Keunggulan dari struktur hash table ini adalah waktu aksesnya yang cukup cepat, jika record yang dicari langsung berada pada angka hash lokasi penyimpanannya. Akan tetapi pada kenyataannya sering sekali ditemukan hash table yang record-recordnya mempunyai angka hash yang sama (bertabrakan). Pemetaan hash function yang digunakan bukanlah pemetaan satusatu, (antara dua record yang tidak sama dapat dibangkitkan angka hash yang sama) maka dapat terjadi bentrokan (collision) dalam penempatan suatu data record. Untuk mengatasi hal ini, maka perlu diterapkan kebijakan resolusi bentrokan (collision resolution policy) untuk menentukan lokasi record dalam tabel. Umumnya kebijakan resolusi bentrokan adalah dengan mencari lokasi tabel yang masih kosong pada lokasi setelah lokasi yang berbentrokan.

Operasi Pada Hash Tabel

- insert: diberikan sebuah key dan nilai, insert nilai dalam table
 - find: diberikan sebuah key, temukan nilai yang berhubungan dengan key
 - remove: diberikan sebuah key, temukan nilai yang berhubungan dengan key, kemudian hapus nilai tersebut
 - getIterator: mengembalikan iterator, yang memeriksa nilai satu demi satu
- Struktur dan Fungsi pada Hash Tabel.

Hash table menggunakan struktur data array asosiatif yang mengasosiasikan record dengan sebuah field kunci unik berupa bilangan (hash) yang merupakan representasi dari record tersebut. Misalnya, terdapat data berupa string yang hendak disimpan dalam sebuah hash table. String tersebut direpresentasikan dalam sebuah field kunci k.

Cara untuk mendapatkan field kunci ini sangatlah beragam, namun hasil akhirnya adalah sebuah bilangan hash yang digunakan untuk menentukan lokasi record. Bilangan hash ini dimasukan ke dalam hash function dan menghasilkan indeks lokasi record dalam tabel.

$k(x)$ = fungsi pembangkit field kunci (1)

$h(x)$ = hash function (2)

Contohnya, terdapat data berupa string “abc” dan “xyz” yang hendak disimpan dalam struktur hash table. Lokasi dari record pada tabel dapat dihitung dengan menggunakan $h(k(“abc”))$ dan $h(k(“xyz”))$.

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;
            }
        }
        prev = current;
        current = 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);

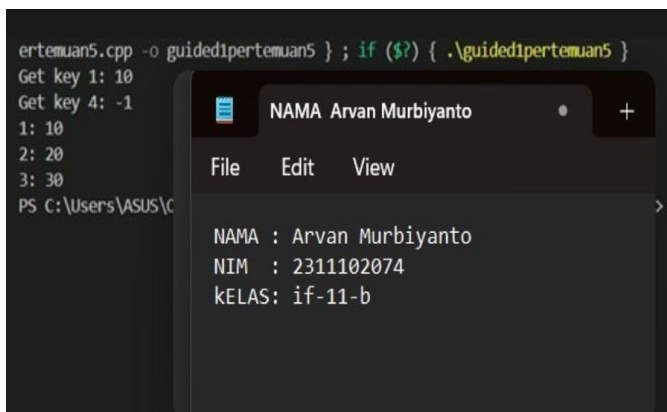

// Traversal

ht.traverse();


return 0;
}

```

Output:



```

ertemuan5.cpp -o guided1pertemuan5 } ; if ($?) { .\guided1pertemuan5 }
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\Users\ASUS\C

```

Deskripsi:

Pada program di atas memiliki fungsi `hash_func` yang menghitung indeks dalam tabel hash berdasarkan kunci (key) dengan menggunakan operasi modulo terhadap `MAX_SIZE`. Setiap elemen direpresentasikan oleh struktur `Node`. Yang setiap node nya memiliki atribut `key`, `value` dan `next`. untuk fungsi `main`, membuat objek `HashTable` (`ht`) dan melakukan beberapa operasi: Menyisipkan pasangan kunci-nilai (1, 10), (2, 20), dan (3, 30). Mencari nilai untuk kunci 1 dan 4. Menghapus elemen dengan kunci 4. Menampilkan seluruh isi tabel hash.

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;
}

```

Output:

The screenshot shows the output of a C++ program in a terminal window. The output is as follows:

```

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\ASUS\OneDri

```

Overlaid on the terminal is a Notepad window titled "NAMA Arvan Murbiyanto". The Notepad window contains the following text:

```

File Edit View

NAMA : Arvan Murbiyanto
NIM : 2311102074
kELAS: if-11-b

```

Deskripsi:

Pada program di atas merupakan implementasi dari tabel hash menggunakan chaining. Setiap elemen tabel berisi vektor yang menyimpan simpul-simpul hash. Metode-metode yang didefinisikan adalah hashFunc(string key), insert, remove, searchByName, print(). Di fungsi main membuat objek employee_map dari kelas HashMap. Kemudian memasukkan beberapa data (nama dan nomor telepon) ke dalam tabel hash. Terakhir, kita menghapus data dengan nama "Mistah" dan menampilkan isi tabel hash.

C. UNGUIDED

1. 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 :
 - 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 rentang nilai (80 – 90).

SOURCE CODE

```
#include <iostream>
#include <vector>
using namespace std;

// Struktur data untuk menyimpan data mahasiswa
struct Mahasiswa {
    int nim;
    int nilai;
};

// Ukuran hash table
const int SIZE = 10;

// Hash table untuk menyimpan data mahasiswa
vector<Mahasiswa> hashTable[SIZE];

// Fungsi hash sederhana
int hashFunction(int nim) {
    return nim % SIZE;
}

// Fungsi untuk menambahkan data mahasiswa ke hash table
void tambahData(int nim, int nilai) {
    int index = hashFunction(nim);
    hashTable[index].push_back({nim, nilai});
}

// Fungsi untuk menghapus data mahasiswa dari hash table
// berdasarkan NIM
void hapusData(int nim) {
    int index = hashFunction(nim);
    for (int i = 0; i < hashTable[index].size(); i++) {
        if (hashTable[index][i].nim == nim) {
```

```

        hashTable[index].erase(hashTable[index].begin() + i);
        break;
    }
}

// Fungsi untuk mencari data mahasiswa berdasarkan NIM
void cariByNIM(int nim) {
    int index = hashFunction(nim);
    for (int i = 0; i < hashTable[index].size(); i++) {
        if (hashTable[index][i].nim == nim) {
            cout << "Data ditemukan - NIM: " <<
hashTable[index][i].nim << ", Nilai: " << hashTable[index][i].nilai
<< endl;
            return;
        }
    }
    cout << "Data tidak ditemukan." << endl;
}

// Fungsi untuk mencari data mahasiswa berdasarkan rentang nilai
void cariByNilai(int minNilai, int maxNilai) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < hashTable[i].size(); j++) {
            if (hashTable[i][j].nilai >= minNilai &&
hashTable[i][j].nilai <= maxNilai) {
                cout << "NIM: " << hashTable[i][j].nim << ", Nilai:
" << hashTable[i][j].nilai << endl;
            }
        }
    }
}

// Fungsi untuk menampilkan menu
void tampilkanMenu() {
    cout << "Menu: \n";
    cout << "1. Tambah Data Mahasiswa\n";
    cout << "2. Hapus Data Mahasiswa\n";
    cout << "3. Cari Data Mahasiswa berdasarkan NIM\n";
    cout << "4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 -
90)\n";
    cout << "5. Keluar\n";
}

int main() {
    int pilihan;
    do {
        tampilkanMenu();

```

```

    cout << "Masukkan pilihan: ";
    cin >> pilihan;

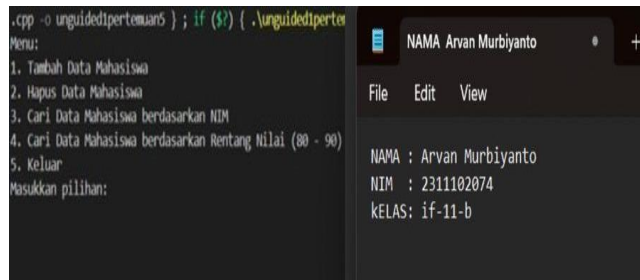
    switch (pilihan) {
        case 1: {
            int nim, nilai;
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan nilai: ";
            cin >> nilai;
            tambahData(nim, nilai);
            break;
        }
        case 2: {
            int nim;
            cout << "Masukkan NIM yang akan dihapus: ";
            cin >> nim;
            hapusData(nim);
            break;
        }
        case 3: {
            int nim;
            cout << "Masukkan NIM yang akan dicari: ";
            cin >> nim;
            cariByNIM(nim);
            break;
        }
        case 4: {
            cariByNilai(80, 90);
            break;
        }
        case 5:
            cout << "Program selesai.\n";
            break;
        default:
            cout << "Pilihan tidak valid.\n";
    }
} while (pilihan != 5);

return 0;
}

```

SCREENSHOT OUTPUT

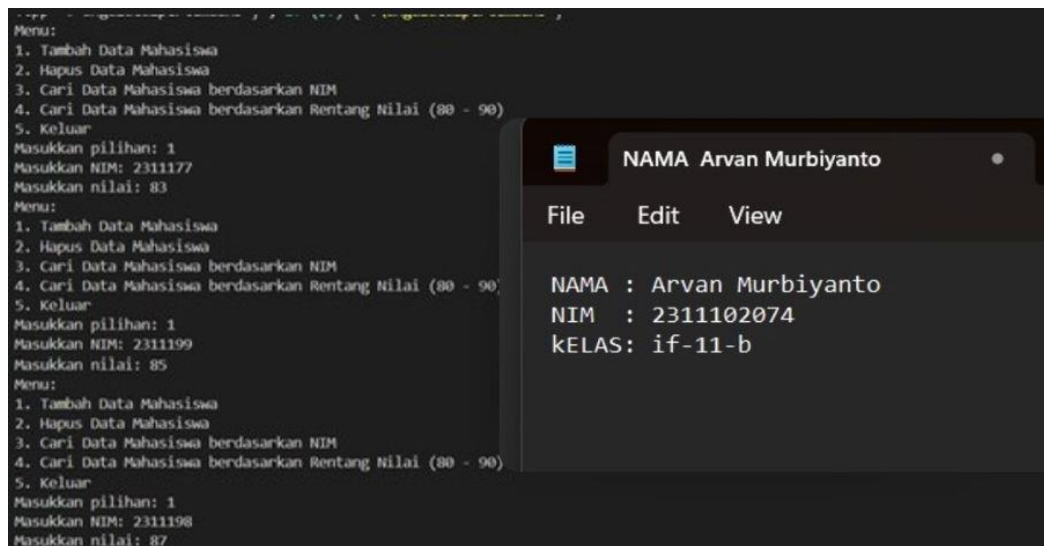
- Menu



The screenshot shows a terminal window on the left with a C++ program's menu. The menu options are: 1. Tambah Data Mahasiswa, 2. Hapus Data Mahasiswa, 3. Cari Data Mahasiswa berdasarkan NIM, 4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90), and 5. Keluar. Below the menu, it says 'Masukkan pilihan:'. To the right is a window titled 'NAMA Arvan Murbiyanto' with a menu bar 'File Edit View'. The window displays the student's information: 'NAMA : Arvan Murbiyanto', 'NIM : 2311102074', and 'KELAS: if-11-b'.

```
.cpp -o unguidedipertemuan5 ); if ($?) { .\unguidedipertemuan5 }  
Menu:  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari Data Mahasiswa berdasarkan NIM  
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)  
5. Keluar  
Masukkan pilihan:
```

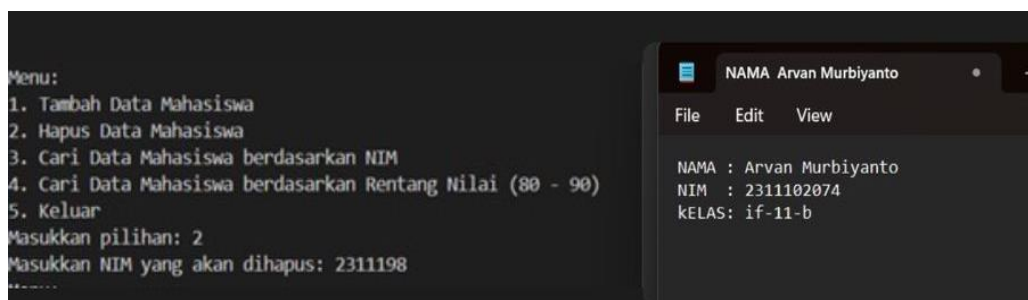
- Data mahasiswa



This screenshot shows the program menu and the student data window after adding a new student. The terminal menu is the same, but the user has entered '1' for 'Tambah Data Mahasiswa', then '2311177' for 'Masukkan NIM', and '83' for 'Masukkan nilai'. The window on the right now shows the updated student data: 'NAMA : Arvan Murbiyanto', 'NIM : 2311102074', and 'KELAS: if-11-b'.

```
Menu:  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari Data Mahasiswa berdasarkan NIM  
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)  
5. Keluar  
Masukkan pilihan: 1  
Masukkan NIM: 2311177  
Masukkan nilai: 83  
Menu:  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari Data Mahasiswa berdasarkan NIM  
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)  
5. Keluar  
Masukkan pilihan: 1  
Masukkan NIM: 2311199  
Masukkan nilai: 85  
Menu:  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari Data Mahasiswa berdasarkan NIM  
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)  
5. Keluar  
Masukkan pilihan: 1  
Masukkan NIM: 2311198  
Masukkan nilai: 87
```

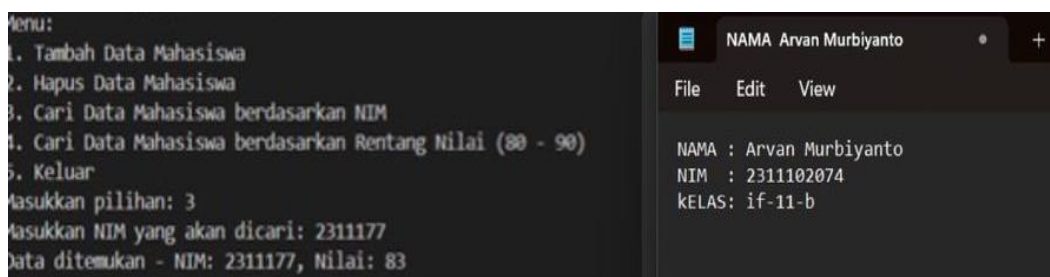
- Hapus data



The screenshot shows the program menu and the student data window after deleting a student. The terminal menu is the same, but the user has entered '2' for 'Hapus Data Mahasiswa', then '2311198' for 'Masukkan NIM yang akan dihapus'. The window on the right still shows the student data: 'NAMA : Arvan Murbiyanto', 'NIM : 2311102074', and 'KELAS: if-11-b'.

```
Menu:  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari Data Mahasiswa berdasarkan NIM  
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)  
5. Keluar  
Masukkan pilihan: 2  
Masukkan NIM yang akan dihapus: 2311198  
.....
```

- Cari data berdasarkan nim



This screenshot shows the program menu and the student data window after searching for a student by NIM. The terminal menu is the same, but the user has entered '3' for 'Cari Data Mahasiswa berdasarkan NIM', then '2311177' for 'Masukkan NIM yang akan dicari'. The window on the right still shows the student data: 'NAMA : Arvan Murbiyanto', 'NIM : 2311102074', and 'KELAS: if-11-b'.

```
Menu:  
1. Tambah Data Mahasiswa  
2. Hapus Data Mahasiswa  
3. Cari Data Mahasiswa berdasarkan NIM  
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)  
5. Keluar  
Masukkan pilihan: 3  
Masukkan NIM yang akan dicari: 2311177  
Data ditemukan - NIM: 2311177, Nilai: 83
```

- Cari data berdasarkan nilai



DESKRIPSI:

Program di atas menggunakan struktur Mahasiswa yang memiliki dua anggota, yaitu nim dan nilai. Pada ukuran hash table ditentukan oleh konstanta SIZE. Untuk Fungsi hashFunction menghitung indeks hash berdasarkan NIM mahasiswa dengan menggunakan operasi modulo. Untuk Operasi pada Hash Table yaitu ada tambahData: ,hapusData, cari NIM dan Mencari data mahasiswa berdasarkan rentang nilai.

D. KESIMPULAN

Tabel hash adalah struktur data yang memungkinkan penyimpanan dan pengambilan data secara efisien. Tabel hash menyimpan data dalam bentuk pasangan nilai kunci. Data disimpan dan diambil menggunakan fungsi hash yang mengubah kunci dalam tabel menjadi indeks. Keuntungan utama tabel hash adalah akses data yang cepat, terutama bila jumlah datanya besar. Namun, penting untuk dicatat bahwa potensi tabrakan hash harus ditangani dengan tepat untuk memastikan efisiensi dan integritas data.

E. REFERENSI

blogspot.com (2013, 01) algoritma dan struktur data hashing

<https://muqoddasrengmadureh.blogspot.com/2013/01/algoritma-dan-struktur-data-hashing.html>

wordpress.com (2012, 22 november) struktur data hash table

<https://informatika11d.wordpress.com/2012/11/22/struktur-data-hash-table>