

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



Disusun Oleh :

Anisah Syifa Mustika Riyanto
2311102080

Dosen

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

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

A. Dasar Teori

Pengertian dan Fungsi

Hash Table adalah sebuah struktur data yang sangat cepat dalam insertion dan searching. Hash table diimplementasikan menggunakan array. Penambahan dan pencarian sebuah key pada hash table berdasarkan fungsi hash yang digunakan. Fungsi hash memetakan elemen pada indeks 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. Fungsi hash yang baik memiliki sifat berikut: mudah dihitung, cukup mampu mendistribusikan key, meminimalkan jumlah collision (tabrakan) yang terjadi.

Cara Membuat Hash Table

Untuk membuat *hash table*, sepotong memori perlu diblokir dengan cara yang sama seperti saat membuat *array*. Anda perlu membuat indeks yang didasarkan pada kunci dengan menggunakan fungsi *hash* karena indeks yang dihasilkan harus sesuai dengan potongan memori.

Ada dua pemeriksaan yang dibutuhkan saat menempatkan data baru pada *hash table*, yaitu nilai *hash* dari kunci dan bagaimana nilainya dibandingkan dengan objek lain. Pemeriksaan ini diperlukan saat membuatnya dengan Python karena saat data dimasukkan, kunci akan di-*hash* dan di-*mask* agar diubah menjadi larik atau indeks yang efisien.

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 indeksnya. 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.

B. 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; //pengurutan
    int value; //nilai
    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;

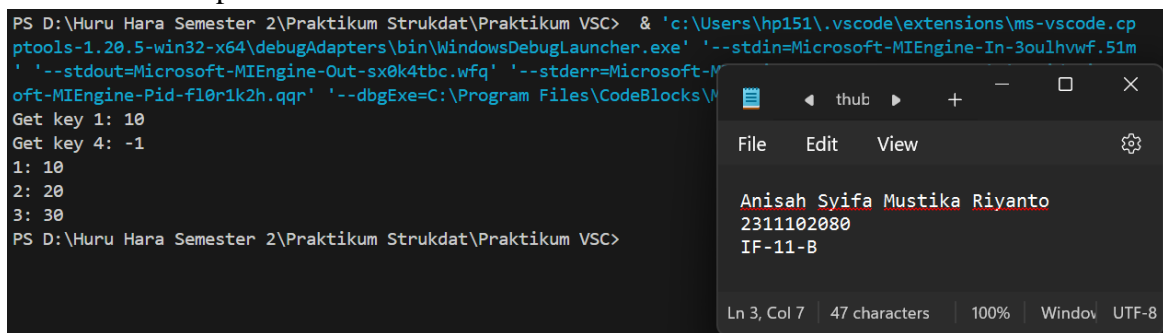
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

Screenshots Output



```

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp151\.vscode\extensions\ms-vscode.cp
ptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-3oulhvfwf.51m
' '--stdout=Microsoft-MIEngine-Out-sx0k4tbc.wfq' '--stderr=Microsoft-M
oft-MIEngine-Pid-fl0r1k2h.qqr' '--dbgExe=C:\Program Files\CodeBlocks\
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>

```

Deskripsi:

Program ini mengimplementasikan sebuah tabel hash sederhana dengan penggunaan chaining. Setiap indeks dalam array mewakili wadah di mana key dengan nilai hash yang sama disimpan. Program memiliki fungsi untuk menginput, mencari, dan menghapus pasangan key-value, serta melakukan traversal tabel hash untuk menampilkan semua pasangan key-value yang disimpan. Dalam fungsi main, kode ini menunjukkan operasi-operasi dengan menginput tiga pasangan key-value, mencari dua key, menghapus key yang tidak ada, dan kemudian melakukan traversal tabel hash untuk menampilkan pasangan key-value yang tersisa.

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;
            }
        }
    }
}
```

```

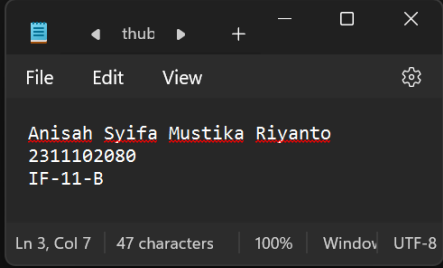
    }
    string searchByName(string name)
    {
        int hash_val = hashFunc(name);
        for (auto node : table[hash_val])
        {
            if (node->name == name)
            {
                return node->phone_number;
            }
        }
        return "";
    }
    void print()
    {
        for (int i = 0; i < TABLE_SIZE; i++)
        {
            cout << i << ": ";
            for (auto pair : table[i])
            {
                if (pair != nullptr)
                {
                    cout << "[" << pair->name << ", " << pair-
>phone_number << "]"<< 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;
}

```

Screenshots Output

```
ptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ewefpdgy.yfd
' '--stdout=Microsoft-MIEngine-Out-b2oyse2j.ixr' '--stderr=Microsoft-MIEngine-Error-kscplfb3.hka' '--pid=Micros
oft-MIEngine-Pid-5ialnndp.pq0' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi'
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:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>
```



Deskripsi:

Program ini mengimplementasikan sebuah tabel hash untuk mengatur dan mencari nomor telepon pekerja berdasarkan nama. Setiap node dalam hash table terdiri dari nama dan nomor telepon karyawan. Tabel hash ini menggunakan fungsi hash yang sederhana untuk menghitung indeks di mana nama karyawan disimpan. Class HashMap digunakan untuk mengimplementasikan struktur hash table dengan menggunakan vector yang menampung pointer ke HashNode. Fungsi-fungsi yang tersedia termasuk menginput, menghapus, dan mencari nama karyawan serta nomor telepon yang terkait.

C. Unguided

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 :

- Setiap mahasiswa memiliki NIM dan nilai.
- Program memiliki tampilan pilihan menu berisi poin 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 <list>

using namespace std;

// class untuk menyimpan data mahasiswa
class Mahasiswa {
public:
    long long NIM;
    int nilai;
};

// class untuk hash table
class HashTable {
private:
    static const int tableSize = 10;
```



```

        list<Mahasiswa> table[tableSize];

public:
    // hash function untuk NIM
    int hashFunction(long long NIM) {
        return NIM % tableSize;
    }

    // fungsi untuk menambahkan data mahasiswa ke hash table
    void addData(long long NIM, int nilai) {
        int index = hashFunction(NIM);
        Mahasiswa mhs;
        mhs.NIM = NIM;
        mhs.nilai = nilai;
        table[index].push_back(mhs);
    }

    // fungsi untuk mencari data mahasiswa berdasarkan NIM
    void searchDataByNIM(long long NIM) {
        int index = hashFunction(NIM);
        bool found = false;
        for (auto it = table[index].begin(); it != table[index].end();
it++) {
            if (it->NIM == NIM) {
                cout << "Data mahasiswa dengan NIM " << NIM << "
ditemukan.\n";
                cout << "Nilai: " << it->nilai << "\n";
                found = true;
                break;
            }
        }
        if (!found) {
            cout << "Data mahasiswa dengan NIM " << NIM << " tidak
ditemukan.\n";
        }
    }

    // fungsi untuk mencari data mahasiswa berdasarkan rentang nilai
(80 - 90)
    void searchDataByRange() {
        bool found = false;
        for (int i = 0; i < tableSize; i++) {
            for (auto it = table[i].begin(); it != table[i].end();
it++) {
                if (it->nilai >= 80 && it->nilai <= 90) {
                    if (!found) {
                        cout << "Data mahasiswa dengan nilai antara 80
dan 90:\n";
                        found = true;
                    }
                    cout << "NIM: " << it->NIM << ", nilai: " << it-
>nilai << "\n";
                }
            }
        }
        if (!found) {
            cout << "Tidak ada data mahasiswa dengan nilai antara 80
dan 90.\n";
        }
    }

```

```

// fungsi untuk menghapus data mahasiswa berdasarkan NIM
void deleteData(long long NIM) {
    int index = hashFunction(NIM);
    bool found = false;
    for (auto it = table[index].begin(); it != table[index].end();
++it) {
        if (it->NIM == NIM) {
            table[index].erase(it);
            cout << "Data mahasiswa dengan NIM " << NIM << "
berhasil dihapus.\n";
            found = true;
            break;
        }
    }
    if (!found) {
        cout << "Data mahasiswa dengan NIM " << NIM << " tidak
ditemukan.\n";
    }
}

// fungsi untuk menampilkan data yang tersimpan
void print() {
    for (int i = 0; i < tableSize; i++) {
        cout << i << ": ";
        for (auto pair : table[i]) {
            cout << "[" << pair.NIM << ", " << pair.nilai << "]";
        }
        cout << endl;
    }
}

};

int main() {
    char ulang;
    HashTable hashTable;
    int choice, nilai;
    long long NIM;
    do {
        cout << "=====\n";
        cout << "          MENU DATA MAHASISWA:\n";
        cout << "=====\n";
        cout << "1. Tambah Data \n";
        cout << "2. Cari data berdasarkan NIM\n";
        cout << "3. Cari data berdasarkan nilai (80 - 90)\n";
        cout << "4. Hapus Data \n";
        cout << "5. Tampilkan Data \n";
        cout << "\nPilih menu: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                cout << "Masukkan NIM : ";
                cin >> NIM;

                cout << "Masukkan Nilai : ";
                cin >> nilai;
                hashTable.addData(NIM, nilai);
                break;
            }
            case 2: {
                cout << "Masukkan NIM : ";
                cin >> NIM;

```

```

        hashTable.searchDataByNIM(NIM);
        break;
    }
    case 3: {
        hashTable.searchDataByRange();
        break;
    }
    case 4: {
        cout << "Masukkan NIM : ";
        cin >> NIM;
        hashTable.deleteData(NIM);
        break;
    }
    case 5: {
        hashTable.print();
        break;
    }
    default:
        break;
}
cout << "Lanjutkan memilih?(y/n)";
cin >> ulang;
} while (ulang == 'y');
return 0;
}

```

Screenshots Output:

```

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp151\.vscode\extensions\ms-vscode.cpptools\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-01bbjlbk.ne4' '--stdout=Microsoft-MIEngine-Output-01bbjlbk.ne4' '--pid=Microsoft-MIEngine-Pid-aknzsni.zq5' '--dbgExe=C:\Program Files\Co
xe' '--interpreter=mi'
=====
MENU DATA MAHASISWA:
=====
1. Tambah Data
2. Cari data berdasarkan NIM
3. Cari data berdasarkan nilai (80 - 90)
4. Hapus Data
5. Tampilkan Data

Pilih menu: 1
Masukkan NIM : 2311102080
Masukkan Nilai : 88
Lanjutkan memilih?(y/n)y
=====
MENU DATA MAHASISWA:
=====
1. Tambah Data
2. Cari data berdasarkan NIM
3. Cari data berdasarkan nilai (80 - 90)
4. Hapus Data
5. Tampilkan Data

Pilih menu: 5
0: [2311102080, 88]
1:
2:
3:

```

File Edit View

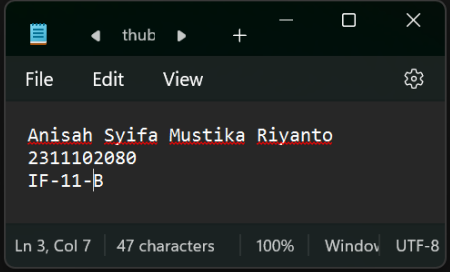
Anisah Syifa Mustika Riyanto
2311102080
IF-11-B

Ln 3, Col 7 | 47 characters | 100% | Window | UTF-8

```
Lanjutkan memilih?(y/n)y
=====
MENU DATA MAHASISWA:
=====
1. Tambah Data
2. Cari data berdasarkan NIM
3. Cari data berdasarkan nilai (80 - 90)
4. Hapus Data
5. Tampilkan Data

Pilih menu: 2
Masukkan NIM : 2311102080
Data mahasiswa dengan NIM 2311102080 ditemukan.
Nilai: 88
Lanjutkan memilih?(y/n)y
=====
MENU DATA MAHASISWA:
=====
1. Tambah Data
2. Cari data berdasarkan NIM
3. Cari data berdasarkan nilai (80 - 90)
4. Hapus Data
5. Tampilkan Data

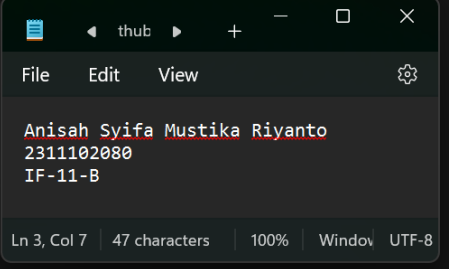
Pilih menu: 3
Data mahasiswa dengan nilai antara 80 dan 90:
NIM: 2311102080, nilai: 88
Lanjutkan memilih?(y/n)y
=====
MENU DATA MAHASISWA:
=====
```



```
=====
MENU DATA MAHASISWA:
=====
1. Tambah Data
2. Cari data berdasarkan NIM
3. Cari data berdasarkan nilai (80 - 90)
4. Hapus Data
5. Tampilkan Data

Pilih menu: 4
Masukkan NIM : 2311102080
Data mahasiswa dengan NIM 2311102080 berhasil dihapus.
Lanjutkan memilih?(y/n)y
=====
MENU DATA MAHASISWA:
=====
1. Tambah Data
2. Cari data berdasarkan NIM
3. Cari data berdasarkan nilai (80 - 90)
4. Hapus Data
5. Tampilkan Data

Pilih menu: 5
0:
1:
2:
3:
4:
5:
6:
7:
8:
9:
Lanjutkan memilih?(y/n)n
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>
```



Deskripsi:

Program tersebut adalah implementasi sederhana dari struktur data hash table yang digunakan untuk menyimpan data mahasiswa. Kelas Mahasiswa digunakan untuk merepresentasikan data individu mahasiswa dengan NIM dan nilai. Kelas HashTable mengimplementasikan hash table dengan ukuran tetap 10, yang menggunakan chaining (menggunakan linked list) untuk menangani tumpukan kunci yang sama. Fungsi hash digunakan untuk menentukan indeks di mana data akan disimpan dalam tabel. Kode ini juga menyediakan beberapa fungsi utama, termasuk penambahan data mahasiswa, pencarian berdasarkan NIM, pencarian berdasarkan rentang nilai, penghapusan data berdasarkan NIM, dan penampilan semua data yang tersimpan. Program utama memberikan menu untuk pengguna agar dapat memilih operasi yang dilakukan pada data mahasiswa.

D. Kesimpulan

1. Hash table adalah cara menyimpan data yang efektif. Data disimpan dalam bentuk array, dan setiap data dipetakan ke lokasi tertentu menggunakan fungsi hash. Hal ini memungkinkan kita untuk menambahkan, menghapus, dan mencari data dengan cepat.
2. Fungsi hash adalah cara membuat key unik untuk setiap data. Key ini digunakan untuk menentukan lokasi data di array. Fungsi hash yang baik harus dapat membuat key yang tidak sama untuk data yang berbeda.
3. Ketika terjadi konflik, hash table menggunakan beberapa cara untuk mengatasi masalah. Cara-cara ini termasuk mencari lokasi lain di array atau menghitung kembali kode hash untuk menemukan lokasi lain.
4. Hash table sangat efektif untuk aplikasi yang memerlukan akses cepat ke data. Mereka sangat berguna jika kita memiliki banyak data yang perlu diakses atau diupdate.
5. Meskipun hash table efektif, hash table juga memiliki beberapa keterbatasan. Salah satu keterbatasan adalah konflik, yang dapat membuat proses lebih lambat jika tidak dikelola dengan baik.

E. Referensi

AlgoritmaDataScienceSchool. Apa itu Hash Table dan Bagaimana Penggunaannya?. (2022). Diakses 13 Mei 2024 dari <https://algorit.ma/blog/hash-table-adalah-2022/>

Holle, K. F. H. (2022). Modul praktikum struktur data.

Malik, D. S. (2023). *C++ programming*. Cengage Learning, EMEA.

Soetanto, H. STRUKTUR DATA.

Stroustrup, B. (2022). *A Tour of C++*. Addison-Wesley Professional.