

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



Disusun Oleh :

Muhammad Arsyad Zaidan (2311102058)

Dosen

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

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

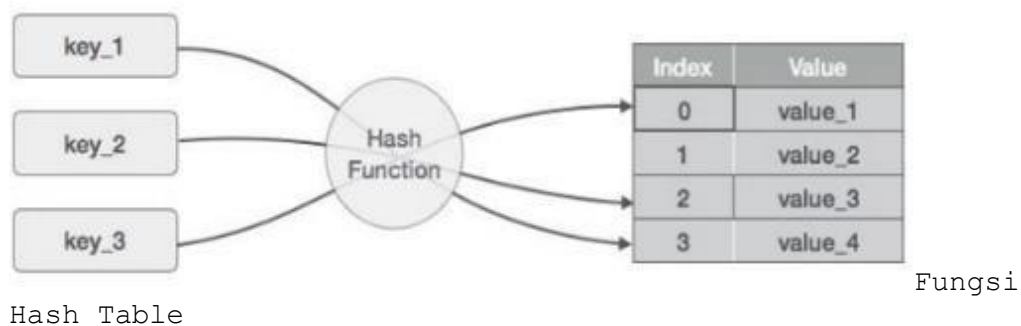
B. Dasar Teori

a. Pengertian Hash Table

Hash Table adalah struktur data yang mengorganisir data ke dalam pasangan kunci-nilai. Hash table biasanya terdiri dari dua komponen utama: array (atau vektor) dan fungsi hash. Hashing adalah teknik untuk mengubah 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 memetakannya 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 fungsi hash, 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 dilakukan melalui penggunaan rumus matematika yang dikenal sebagai fungsi hash. Hasil dari fungsi hash disebut sebagai nilai hash atau hash. Nilai hash adalah representasi dari string karakter asli tetapi biasanya lebih kecil dari aslinya.

b. Operasi Hash Table

1. Insertion:

Memasukkan data baru ke dalam hash table dengan memanggil fungsi hash 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:

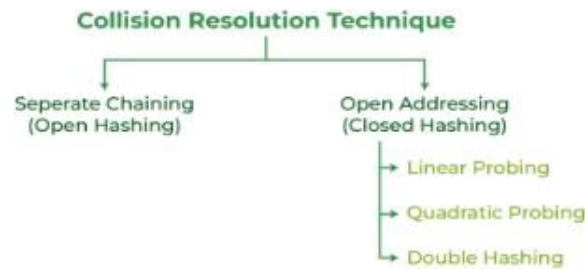
Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

5. Traversal:

Melalui seluruh hash table untuk memproses semua data yang ada dalam tabel.

c. 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 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. Closed 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.

1. Latihan Guided dan Unguided

1) Guided

Guided 1 (Linked List Non Circular)

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;
}

```

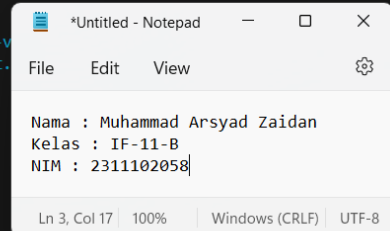
Output Code

```

PS C:\Learning Code\STRUKDAT (PRAKTEK)\Laprak 5_2311102058_Muhammad Arsyad Zaidan>
n32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-v
tderr=Microsoft-MIEngine-Error-5tcab1ey.2a5' '--pid=Microsoft-MIEngine-Pid-cxlegvzt.

Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS C:\Learning Code\STRUKDAT (PRAKTEK)\Laprak 5_2311102058_Muhammad Arsyad Zaidan>

```



Deskripsi Code

Bucket yang disimpan pada hash table di program tersebut menggunakan table sebagai array dinamisnya yang diwakili sebuah linkedlist. Fungsi pada "hash_function" itu yang menjadi kunci argument dan mengembalikan indeks dengan menghitung sisa hasil bagi dari kunci dengan ukuran (MAX_SIZE). Penerapan hash table pada program menggunakan metode chaining untuk menangani terjadinya tabrakan pada program.

Guided 2 (Linked List Circular)

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])

```



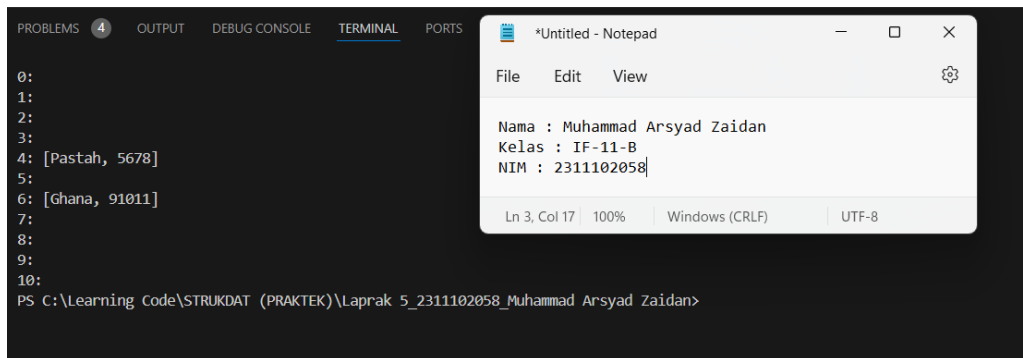
```

        {
            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;
}

```

Output Code



The screenshot shows a terminal window with the following output:

```
0:
1:
2:
3:
4: [Pastah, 5678]
5:
6: [Ghana, 91011]
7:
8:
9:
10:
PS C:\Learning Code\STRUKDAT (PRAKTEK)\Laprak 5_2311102058_Muhammad Arsyad Zaidan>
```

Overlaid on the terminal is a Notepad window titled "*Untitled - Notepad" containing the following text:

```
File Edit View
Nama : Muhammad Arsyad Zaidan
Kelas : IF-11-B
NIM : 2311102058
```

The Notepad window also shows status information at the bottom: "Ln 3, Col 17 | 100% | Windows (CRLF) | UTF-8".

Deskripsi Code

Program dengan menerapkan hash table yang berfungsi menyimpan nama dan nomer telepon. Sama dengan pemrograman sebelumnya, pemrograman hash table tersebut menggunakan metode chaining

2) Unguided

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);
}

```

```

    }
} while (pilihan != 5);

return 0;
}

```

Output Code

1) Menambahkan data pada program

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Masukkan pilihan: 1
Masukkan NIM: 058
Masukkan nilai: 1
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: 059
Masukkan nilai: 2
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: 060
Masukkan nilai: 3
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: 061
Masukkan nilai: 4

```

2) Penghapusan data pada program

```

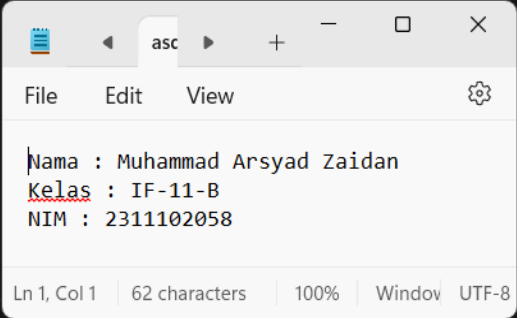
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Masukkan pilihan: 2
Masukkan NIM yang akan dihapus: 058
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM

```

3) Pencarian data (NIM) pada program

- Jika ditemukan

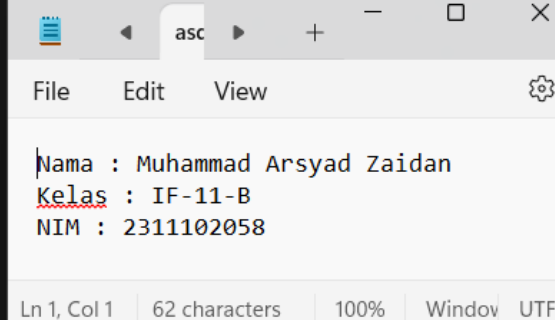
```
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Masukkan pilihan: 3
Masukkan NIM yang akan dicari: 059
Data ditemukan - NIM: 59, Nilai: 2
Menu:
1. Tambah Data Mahasiswa
2. Hapus Data Mahasiswa
3. Cari Data Mahasiswa berdasarkan NIM
4. Cari Data Mahasiswa berdasarkan Rentang Nilai (80 - 90)
5. Keluar
Masukkan pilihan: 
```



A screenshot of a Notepad window titled 'asc'. The text inside reads: 'Nama : Muhammad Arsyad Zaidan', 'Kelas : IF-11-B', and 'NIM : 2311102058'. The status bar at the bottom shows 'Ln 1, Col 1', '62 characters', '100%', 'Window', and 'UTF-8'.

- Jika tidak ditemukan

```
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: 058
Data tidak ditemukan.
Menu:
```

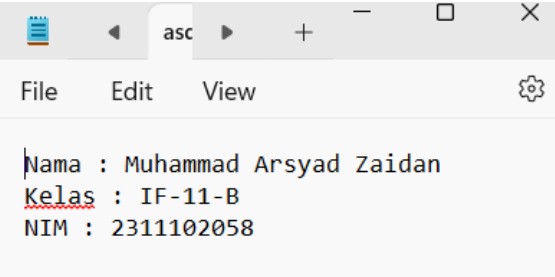


A screenshot of a Notepad window titled 'asc'. The text inside reads: 'Nama : Muhammad Arsyad Zaidan', 'Kelas : IF-11-B', and 'NIM : 2311102058'. The status bar at the bottom shows 'Ln 1, Col 1', '62 characters', '100%', 'Window', and 'UTF-8'.

4) Pencarian data (nilai (80 - 90)) pada program

5) Keluar pada program

```
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: 5
Program selesai.
PS C:\Learning Code\STRUKDAT (PRAKTEK)\Laprak 5 2311102058
```



A screenshot of a Notepad window titled 'asc'. The text inside reads: 'Nama : Muhammad Arsyad Zaidan', 'Kelas : IF-11-B', and 'NIM : 2311102058'. The status bar at the bottom shows 'Ln 1, Col 1', '62 characters', '100%', 'Window', and 'UTF-8'.

2. Kesimpulan

Dari praktikum ini, mahasiswa dapat mempelajari mengimplementasikan hash table pada pemrograman C. struktur data yang mengorganisir data ke dalam pasangan kunci-nilai, memanfaatkan array (atau vektor) dan fungsi hash untuk memetakan rentang nilai kunci ke indeks array. Hash table menggunakan teknik chaining untuk menangani tabrakan atau hash collision, di mana beberapa data memiliki nilai hash yang sama dan disimpan dalam bucket yang sama. Operasi hash table terdapat insertion (penyisipan), deletion (penghapusan), searching (pencarian), dan update (pembaruan). Masing-masing operasi ini menggunakan fungsi hash untuk menentukan posisi bucket yang tepat dalam array, memastikan efisiensi dalam pengelolaan data.

3. Referensi

[1] Tim Asisten Asprak, "Hash Table", Learning Management System, 2024.

[2] Algoritma team, "Pengertian Hash Table dan Cara Penggunaannya", <https://algoritma.blog/hash-table-adalah-2022>, terakhir kali diakses pada tanggal 10 Mei 2024.