

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

**MODUL V
HASH TABLE**



DISUSUN OLEH :
MUHAMAD NASRULLOH
2311102044

DOSEN :
WAHYU ANDI SAPUTRA, S.PD., M.ENG.

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

A. DASAR TEORI

- Pengertian

Hash table adalah struktur data yang memetakan kunci (keys) ke dalam nilai-nilai (values).

KEY	DATA
-----	------

Hash table menggunakan fungsi hash untuk menghitung indeks berdasar kunci. Nilai dapat disimpan pada lokasi yang sesuai berdasar indeks hash table. Jadi dalam implementasi hash table, kunci digunakan sebagai input untuk fungsi hash yang menghasilkan indeks dimana nilai yang terkait dengan kunci tersebut disimpan.

- Key : bilangan bulat unik yang digunakan untuk mengindeks nilai
- Data : nilai yang terkait dengan kunci

Struktur data fungsi hash table biasanya terdiri dari 2 komponen utama: array dan fungsi hash. Array digunakan untuk menyimpan item-item yang terkait dengan kunci, sedangkan fungsi hash digunakan untuk menghitung indeks dimana item-item tersebut disimpan.

Dengan hash table ini akses data menjadi sangat cepat karena dapat langsung menghitung indeks tempat data disimpan.

- Bentuk Operasi

1. Pencarian (search) : untuk mencari elemen/data dalam hash table berdasar kunci atau indeksinya, dengan fungsi hash untuk menghasilkan indeks elemen yang dicari.
2. Penyisipan (insertion) : untuk menyisipkan elemen/data baru ke dalam hash table.
3. Penghapusan (deletion) : untuk menghapus elemen/data dari hash table berdasar kunci atau indeksinya.
4. Update : untuk mengubah nilai elemen/data yang sudah ada dalam hash table.
5. Collision Handling: terjadi ketika 2 atau lebih elemen memiliki indeks yang sama setelah melalui fungsi hash, untuk menangani collision dan memastikan elemen-elemen dengan indeks yang sama dapat disimpan dan diakses dengan benar.
6. Resize : untuk mengubah ukuran hash table jika jumlah elemen/data yang disimpan melebihi kapasitas yang ditentukan.
7. Iterasi : untuk mengakses dan memproses semua elemen/data secara urut.

B. GUIDED

- Guided I

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;

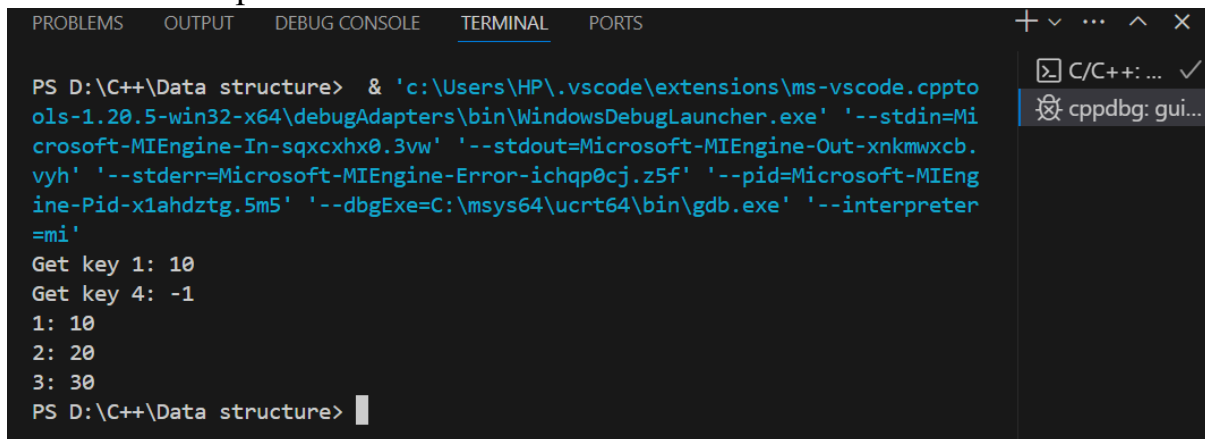
    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}

```

Screenshot Output



```
PS D:\C++\Data structure> & 'c:\Users\HP\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-sqxcxhx0.3vw' '--stdout=Microsoft-MIEngine-Out-xnkmwxcbyyh' '--stderr=Microsoft-MIEngine-Error-ichqp0cj.z5f' '--pid=Microsoft-MIEngine-Pid-x1ahdztg.5m5' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Get key 1: 10
Get key 4: -1
1: 10
2: 20
3: 30
PS D:\C++\Data structure>
```

Deskripsi

Program diatas merupakan implementasi program hash table sederhana dalam C++. Hash table digunakan untuk menyimpan pasangan kunci-nilai, dimana kunci digunakan untuk mencari nilai terkait dengan cepat. Program ini memiliki fungsi untuk menyisipkan (insertion), mencari (searching), dan menghapus (deletion) pasangan kunci-nilai dalam hash table. Setiap elemen hash table direpresentasikan oleh struktur data node yang terdiri dari kunci (key), nilai (value) dan pointer ke node selanjutnya (next) dalam kasus terjadi tumpang tindih (collision). Program ini juga memiliki fungsi untuk menelusuri dan mencetak seluruh hash table setelah operasi yang dilakukan (traverse).

- 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(){

```

```

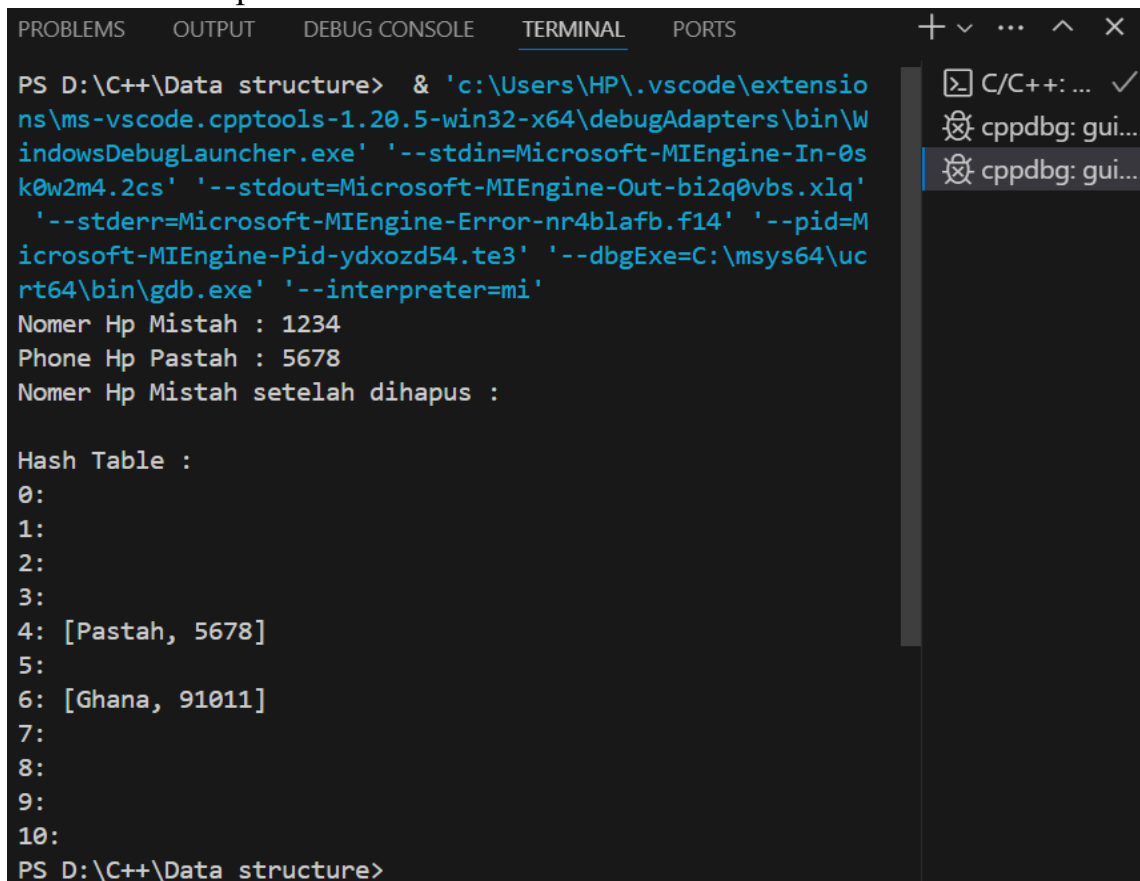
        for (int i=0; i < TABLE_SIZE; i++){
            cout << i << ": ";
            for (auto pair : table[i]){
                if(pair !=nullptr){
                    cout << "[" << pair->name << ", " << pair->phone_number
<< "]";
                }
            }
            cout << endl;
        }
    }
};

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 D:\C++\Data structure> & 'c:\Users\HP\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-0sk0w2m4.2cs' '--stdout=Microsoft-MIEngine-Out-bi2q0vbs.xlq' '--stderr=Microsoft-MIEngine-Error-nr4blafb.f14' '--pid=Microsoft-MIEngine-Pid-ydxozd54.te3' '--dbgExe=C:\msys64\ucrt64\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:\C++\Data structure>
```

Deskripsi

Program diatas merupakan implementasi sederhana dari hash table menggunakan teknik chaining untuk menyelesaikan kasus tumpang tindih (collision) dalam C++. Memungkinkan pengguna untuk menyimpan pasangan nama dan nomer telepon dalam struktur data hash map. Setiap elemen hash table direpresentasikan oleh vektor yang berisi pointer ke node-node yang menyimpan pasangan nama (name) dan nomer telepon (phone_number).

C. UNGUIDED

- Unguided I

Implementasikan hash table untuk menyimpan data mahasiswa. Setiap mahasiswa memiliki NIM dan nilai. Implementasikan fungsi untuk menambah data baru, menghapus data, dan mencari data berdasarkan NIM, dan mencari data berdasarkan nilai. Dengan ketentuan :

- Setiap mahasiswa memiliki NIM dan nilai
- Program memiliki tampilan menu berisi poin C
- Implementasikan fungsi untuk menambahkan data baru, menghapus data, mencari data berdasar NIM, dan mencari data berdasar rentang nilai (80-90).

Source Code

```
#include <iostream>
#include <vector>

using namespace std;

// Struktur data untuk mahasiswa
struct Mahasiswa {
    string nim; // NIM 10 digit (tipe data string)
    float nilai;
};

// Kelas HashTable
class HashTable {
private:
    static const int size = 10; // Ukuran tabel hash
    vector<Mahasiswa> table[size];

    // Fungsi hash sederhana
    int hashFunction(const string &nim) {
        int sum = 0;
        for (char c : nim) {
            sum += c;
        }
        return sum % size;
    }

public:
    // Menambahkan data mahasiswa ke tabel hash
    void tambahData(const string &nim, float nilai) {
        Mahasiswa mhs;
        mhs.nim = nim;
        mhs.nilai = nilai;
        int index = hashFunction(nim);
```

```

        table[index].push_back(mhs);
    }

    // Menghapus data mahasiswa dari tabel hash berdasarkan NIM
    void hapusData(const string &nim) {
        int index = hashFunction(nim);
        for (auto it = table[index].begin(); it != table[index].end(); ++it)
        {
            if (it->nim == nim) {
                table[index].erase(it);
                return;
            }
        }
        cout << "Data mahasiswa dengan NIM " << nim << " tidak ditemukan."
<< endl;
    }

    // Mencari data mahasiswa berdasarkan NIM
    void cariNIM(const string &nim) {
        int index = hashFunction(nim);
        for (const auto &mhs : table[index]) {
            if (mhs.nim == nim) {
                cout << "NIM: " << mhs.nim << ", Nilai: " << mhs.nilai <<
endl;
                return;
            }
        }
        cout << "Data mahasiswa dengan NIM " << nim << " tidak ditemukan."
<< endl;
    }

    // Mencari data mahasiswa berdasarkan nilai (80-90)
    void cariNilai() {
        cout << "Data mahasiswa dengan nilai antara 80-90:" << endl;
        for (int i = 0; i < size; ++i) {
            for (const auto &mhs : table[i]) {
                if (mhs.nilai >= 80 && mhs.nilai <= 90) {
                    cout << "NIM: " << mhs.nim << ", Nilai: " << mhs.nilai
<< endl;
                }
            }
        }
    }
};

int main() {
    HashTable hashTable;
    int choice;

```

```
do {  
    cout << endl;  
    cout << "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^" << endl;  
    cout << "OPSI MENU" << endl;  
    cout << "1. Tambah data mahasiswa" << endl;  
    cout << "2. Hapus data mahasiswa" << endl;  
    cout << "3. Cari NIM data mahasiswa" << endl;  
    cout << "4. Cari NILAI data mahasiswa (80-90)" << endl;  
    cout << "5. Keluar/Exit" << endl;  
    cout << "Pilih: ";  
    cin >> choice;  
  
    switch (choice) {  
        case 1: {  
            string nim;  
            float nilai;  
            cout << "Masukkan NIM (10 digit): ";  
            cin >> nim;  
            cout << "Masukkan nilai: ";  
            cin >> nilai;  
            hashTable.tambahData(nim, nilai);  
            break;  
        }  
        case 2: {  
            string nim;  
            cout << "Masukkan NIM yang ingin dihapus: ";  
            cin >> nim;  
            hashTable.hapusData(nim);  
            cout << "Data mahasiswa dengan NIM " << nim << " telah  
berhasil dihapus." << endl;  
            break;  
        }  
        case 3: {  
            string nim;  
            cout << "Masukkan NIM yang ingin dicari: ";  
            cin >> nim;  
            hashTable.cariNIM(nim);  
            break;  
        }  
        case 4: {  
            hashTable.cariNilai();  
            break;  
        }  
        case 5: {  
            cout << "Program telah selesai. Terimakasih!" << endl;  
            break;  
        }  
        default:
```

```

        cout << "Pilihan Anda salah. Silakan coba lagi." << endl;
    }
} while (choice != 5);

return 0;
}

```

Screenshot Output

```

PS D:\C++\Data structure> & 'c:\Users\HP\.vscode\extensions\ms-vscode.cpptools-1.20.5-win32-x64\debugAdapters\
bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-csrak0yi.ko0' '--stdout=Microsoft-MIEngine-Out-zzo
kkvku.pze' '--stderr=Microsoft-MIEngine-Error-cd4klpjv.afo' '--pid=Microsoft-MIEngine-Pid-siv1ekzu.y2n' '--dbgE
xe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OPSI MENU
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari NIM data mahasiswa
4. Cari NILAI data mahasiswa (80-90)
5. Keluar/Exit
Pilih: 1
Masukkan NIM (10 digit): 2311102044
Masukkan nilai: 90

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OPSI MENU
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari NIM data mahasiswa
4. Cari NILAI data mahasiswa (80-90)
5. Keluar/Exit
Pilih: 1
Masukkan NIM (10 digit): 2311102080
Masukkan nilai: 85

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OPSI MENU
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari NIM data mahasiswa
4. Cari NILAI data mahasiswa (80-90)
5. Keluar/Exit
Pilih: 3
Masukkan NIM yang ingin dicari: 2311102080
NIM: 2311102080, Nilai: 85

```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OPSI MENU
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari NIM data mahasiswa
4. Cari NILAI data mahasiswa (80-90)
5. Keluar/Exit
Pilih: 4
Data mahasiswa dengan nilai antara 80-90:
NIM: 2311102044, Nilai: 90
NIM: 2311102080, Nilai: 85

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OPSI MENU
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari NIM data mahasiswa
4. Cari NILAI data mahasiswa (80-90)
5. Keluar/Exit
Pilih: 2
Masukkan NIM yang ingin dihapus: 2311102044
Data mahasiswa dengan NIM 2311102044 telah berhasil dihapus.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OPSI MENU
1. Tambah data mahasiswa
2. Hapus data mahasiswa
3. Cari NIM data mahasiswa
4. Cari NILAI data mahasiswa (80-90)
5. Keluar/Exit
Pilih: 5
Program telah selesai. Terimakasih!
PS D:\C++\Data structure> 
```

IDENTITAS

File Edit View

NAMA : MUHAMAD NASRULLOH
NIM : 2311102044
KELAS : IF 11-B

Deskripsi

Program diatas sebuah implementasi sederhana dari sebuah hash table dalam C++. Program ini memungkinkan pengguna untuk melakukan operasi dasar pada data mahasiswa seperti menambahkan data (tambahData), menghapus data (hapusData), mencari data berdasar NIM (cariNIM), mencari data berdasar nilai mahasiswa 80-90 (cariNilai). Program juga menyimpan data dengan array dari vektor dan memungkinkan pengguna untuk memilih opsi menu yang tersedia sesuai operasi yang diinginkan hingga memilih keluar/exit.

D. KESIMPULAN

Hash table adalah struktur data yang serbaguna dan efisien yang dapat digunakan untuk menyimpan sejumlah besar data. Selain itu, dengan hash table memungkinkan akses waktu relatif cepat ke nilai yang terkait dengan kunci. Struktur ini menggunakan array sebagai media penyimpanan data dan teknik hashing untuk menghasilkan indeks suatu elemen yang terkait dengan kunci. Kelebihan hash table termasuk efisiensi dan kecepatan, serta kemampuan untuk mengatasi data yang berjumlah sangat banyak. Namun disisi lain juga ada kekurangan seperti kemungkinan terjadi collision (timpang tindih) yang dapat diatasi dengan teknik seperti closed hashing dan open hashing.

E. REFERENSI

[1] Vijaykrishna Ram & Bradley Kouchi, 2023. How To Implement a Sample Hash Table in C/C++. Digital Ocean.

<https://www.digitalocean.com/community/tutorials/hash-table-in-c-plus-plus>.

[2] Hash Table. Programiz. <https://www.programiz.com/dsa/hash-table>.

[3] Annisa, 2023. Struktur Data Hash Table : Pengertian, Cara Kerja dan Operasi Hash Table. Sumatera Utara: Fikti Umsu.

<https://fikti.umsu.ac.id/struktur-data-hash-table-pengertian-cara-kerja-dan-operasi-hash-table/>.