LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL V
HASH TABLE

Disusun Oleh :

NAMA : FAISAL KHOIRUDDIN

NIM : 2311102046

Dosen

WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

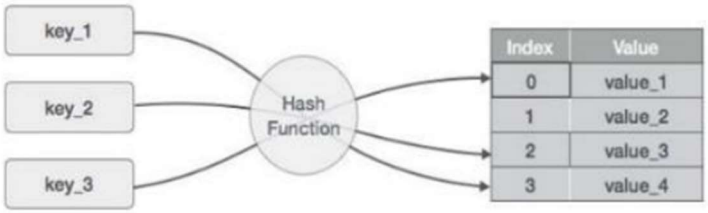
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. Dasar Teori

1) 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. Tabel hash pada dasarnya adalah array yang berisi nilai atau data lainnya struktur yang dapat menyimpan nilai berantai, seperti linked list, seperti pada std::unordered_map implementasi GCC dari pustaka standar C++. 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 memetakkannya 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



2) Fungsi Hash Table

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.

3) 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

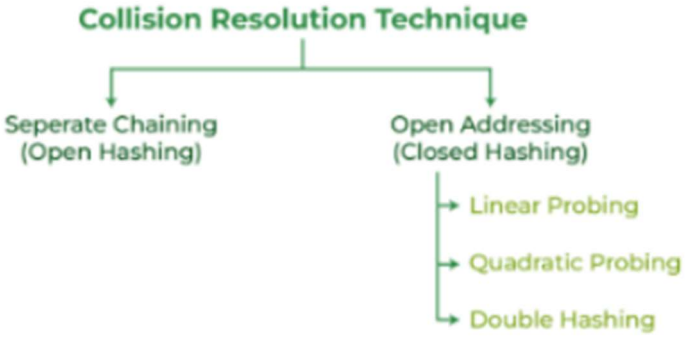
   Memperbarui data dalam hash table dengan mencari data menggunakan fungsi hash, dan kemudian memperbarui data yang ditemukan.

5. Traversal

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

4) 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

a. Linear Probing

b. 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.

c. Quadratic Probing

d. Penanganannya hampir sama dengan metode linear, hanya lompatannya tidak satu-satu, tetapi quadratic (12, 22, 32, 42, ...) Double Hashing

e. Pada saat terjadi collision, terdapat fungsi hash yang kedua untuk menentukan posisinya kembali.

Guided

1. Guided 1

Source Code

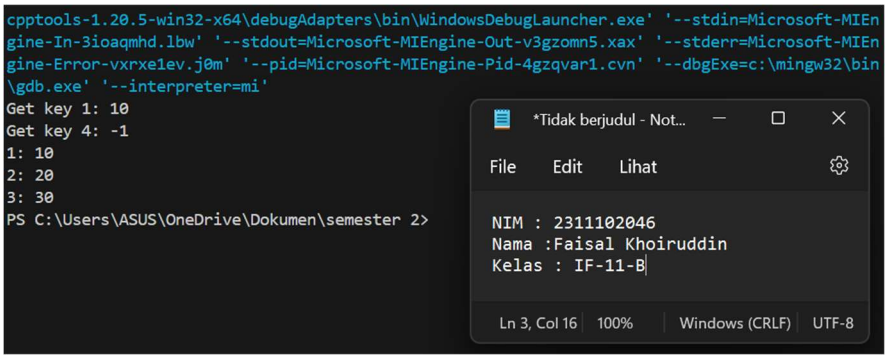
```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
// Fungsi hash sederhana
int hash_func(int key) // mengembalikan
hasing
{
    return key % MAX_SIZE; // ukuran
maksimal data hash
}
// Struktur data untuk setiap node
struct Node //pengurutan indeks 1,2 ,3
{
    int key;
    int value;
    Node *next;
    Node(int key, int value) : key(key),
value(value),
                                next(nullptr
) {}
};
// Class hash table
class HashTable // hashing
{
private: // buat private mode
    Node **table; // node tidak bisa oleh
class lain

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) //
masukin key dan 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) // untuk mencari
    {
        int index = hash_func(key);
        Node *current = table[index];
        while (current != nullptr)
        {
            if (current->key == key)
            {
                return current->value;
            }
            current = current->next;
        }
        return -1; // misalkan searching
tidak ada maka ada data -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() // eksekusi 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; // mencari node urutan 1
    cout << "Get key 4: " << ht.get(4) <<
endl;

    // Deletion
    ht.remove(4);

    // Traversal
    ht.traverse();

    return 0;
}
```

Screenshots Output



Deskripsi:

Program tersebut merupakan Hash Table. Program tersebut melakukan operasi seperti insert, get, remove, dan traverse pada Hash Table. Program tersebut menampilkan output hasil dari operasi void insert, int get, void remove, dan void traverse.

- #include <iostream> ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++

- using namespace std; ➔ digunakan untuk mendeklarasikan/ memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std

- const int MAX_SIZE = 10; ➔ mendefinisikan ukuran maksimal tabel hash sebagai konstanta
- int hash_func(int key) ➔ fungsi untuk menghitung nilai hash dari kunci
- return key % MAX_SIZE; ➔ mengembalikan nilai hash sebagai module dari ukuran maksimal tabel hash
- struct Node ➔ definisikan struktur node
- int key; ➔ variabel untuk menyimpan key
- int value; ➔ variabel untuk menyimpan nilai
- Node *next; ➔ pointer ke node berikutnya dalam rantai
- Node(int key, int value) : key(key), value(value), next(nullptr) {} ➔ konstruktor node
- class HashTable ➔ mendefinisikan kelas hashtable
- private: ➔ mendefinisikan anggota privat
- Node **table; ➔pointer ke Array of pointer node
- public: ➔ mendefinisikan anggota publik
- HashTable() ➔ konstruktor hashtable
- table = new Node *[MAX_SIZE](); ➔ membuat array of pointer node dengan ukuran Max_size dan penginisialisasi semua elemen dengan nullptr
- HashTable() ➔ destruktor hash table
- for (int i = 0; i < MAX_SIZE; i++) ➔ perulangan untuk setiap indeks tabel
- Node *current = table[i]; ➔ mengambil node pertama di indeks tabel
- while (current != nullptr) ➔ perulangan selama node bukan null ptr
- Node *temp = current; ➔ menyimpan node saat ini ke variabel temp
- current = current->next; ➔ memindahkan current ke node berikutnya

- delete temp; ➔ menghapus noda yang disimpan di temp
- delete[] table; ➔ menghapus array of pointer node
- void insert(int key, int value) ➔ prosedur untuk memasukkan data ke tabel hash
- int index = hash_func(key); ➔ menghitung indeks tabel dari kunci
- Node *current = table[index]; ➔ mengambil node pertama di indeks tabel
- while (current != nullptr) ➔ perulangan selama node bukan null ptr
- if (current->key == key) ➔ percabangan if jika kunci sudah ada di tabel
- current->value = value; ➔ mengupdate nilai
- return; ➔ keluar dari fungsi
- current = current->next; ➔ memindahkan current ke mode berikutnya
- Node *node = new Node(key, value); ➔ membuat node baru
- node->next = table[index]; ➔ menyambungkan node baru ke mode pertama di index tabel
- table[index] = node; ➔ menyimpan mode baru sebagai node pertama di indeks tabel
- int get(int key) ➔ fungsi untuk mencari data berdasarkan kunci
- int index = hash_func(key); ➔ menghitung indeks tabel dari kunci
- Node *current = table[index]; ➔ mengambil node pertama di indeks tabel
- while (current != nullptr) ➔ perulangan selama node bukan null ptr
- if (current->key == key) ➔ percabangan if jika kunci ditemukan
- return current->value; ➔ mengembalikan nilai

- current = current->next; ➔ memindahkan current ke node berikutnya
- return -1; ➔ misalkan searching tidak ada maka ada data -1
- void remove(int key) ➔ prosedur untuk menghapus data dari tabel hash
- int index = hash_func(key); ➔ menghitung indeks tabel dari kunci
- Node *current = table[index]; ➔ mengambil noda pertama di indeks tabel
- Node *prev = nullptr; ➔ membuat pointer prive untuk menyimpan node sebelum current
- While (current != nullptr) ➔ perulangan selama node bukan nullptr
- if (current->key == key) ➔ percabangan if jika kunci ditemukan
- if (prev == nullptr) ➔ jika current adalah node pertama
- table[index] = current->next; ➔ menyimpan node berikutnya sebagai mode pertama di indeks tabel
- else ➔ jika current bukan node pertama
- prev->next = current->next; ➔ menyambungkan node sebelumnya ke node berikutnya
- delete current; ➔ menghapus node saat ini
- return; ➔ keluar dari fungsi
- prev = current; ➔ memindahkan prev ke mode saat ini
- current = current->next; ➔ memindahkan current ke node berikutnya
- void traverse() ➔ prosedur untuk mencetak semua data
- for (int i = 0; i < MAX_SIZE; i++) ➔ perulangan untuk setiap indeks tabel
- Node *current = table[i]; ➔ mengambil node pertama di indeks tabel

- while (current != nullptr) ➔ perulangan selama node tidak null
- cout << current->key << ": " << current->value << endl; ➔ cout mencetak kunci dan nilai
- current = current->next; ➔ memindahkan current ke node berikutnya
- int main() ➔ merupakan fungsi utama
- HashTable ht; ➔ membuat objek ht dari kelas hashtable
- ht.insert(1, 10); ➔ memasukkan pasangan kunci dan nilai ke dalam hashtable
- ht.insert(2, 20); ➔ memasukkan pasangan kunci dan nilai ke dalam hashtable
- ht.insert(3, 30); ➔memasukkan pasangan kunci dan nilai ke dalam hashtable
- cout << "Get key 1: " << ht.get(1) << endl; ➔ mencari dan mencetak nilai dari kunci 1
- cout << "Get key 4: " << ht.get(4) << endl; ➔ mencari dan mencetak nilai dari kunci 4
- ht.remove(4); ➔ menghapus kunci 4 dari hashtable
- ht.traverse(); ➔ menelusuri dan mencetak semua kunci dan nilai dalam hashtable
- return 0; ➔ program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

Guided 2 : Linked List Circular

```
#include <iostream>
#include <string>
#include <vector> // membaca nilai vektor


using namespace std;
const int TABLE_SIZE = 11;
```

```
string name;
string phone_number;
class HashNode {
public: //ada kelas lain yang mengambil
name dan phone number
    string name; //memasukkan nama dan no
hp
    string phone_number;

    HashNode(string name, string
phone_number) {
        this->name = name;
        this->phone_number = phone_number;
    }
};

class HashMap { // memasukkan nama dan no
hp keluarnya tabel
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;
```

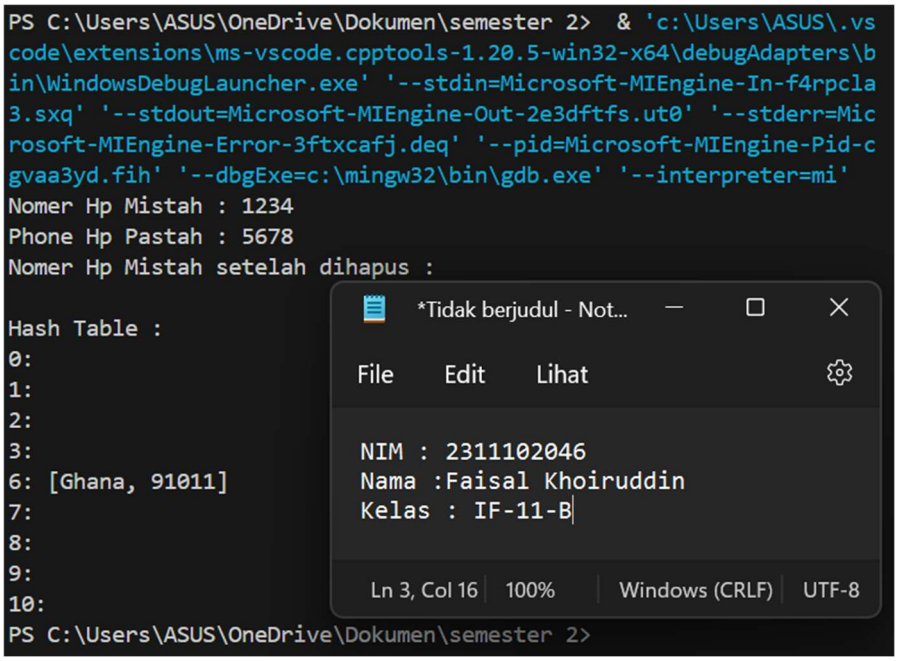
```
        }
        void insert(string name, string
phone_number) { //masukin nama dan no telp
               int hash_val =
hashFunc(name);


               for (auto node :
table[hash_val]) { // otomatis node
1,2,3,4
                   if (node->name == name)
{
                       node->phone_number
= phone_number;
                       return;
                   }
               }
               table[hash_val].push_back(n
ew HashNode(name, phone_number));
        }


        void remove(string name) { //
untuk menghapus berdasarkan nama
            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)
{ // cari suatu data berdasarkan nama
        int hash_val = hashFunc(name);
        for (auto node : table[hash_val])
{
            if (node->name == name) {
                return node-
>phone_number;
            }
        }
        return "";
        }


        void print() { // menampilkan
tabel size dan value dari hash
            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() { //eksekusi program
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; // di dalam employee map keluarnya
mistah
cout << "Phone Hp Pastah : "
<<employee_map.searchByName("Pastah") <<
endl;

employee_map.remove("Mistah"); // remove
no hp mistah

cout << "Nomer Hp Mistah setelah dihapus :
" <<employee_map.searchByName("Mistah") <<
endl << endl;

cout << "Hash Table : " << endl;
```

```
employee_map.print(); // print tabel


return 0;
}
```

Screenshots Output



Deskripsi:

Program tersebut merupakan contoh dari Hash Table. Program tersebut melakukan operasi seperti void insert untuk menambahkan, void remove untuk menghapus , string searchByName untuk mencari berdasarkan nama, dan void print untuk mencetak. Output yang ditampilkan pada program tersebut berupa operasi dari void insert, void remove, string searchByName, dan void print.

- #include <iostream> ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++

- #include <string> ➔ untuk memanipulasi data bertipe string
- #include <vector> ➔ vektor menyimpan elemen jenis tertentu dalam pengaturan linier, dan memungkinkan akses acak cepat ke elemen apa pun
- using namespace std; ➔ digunakan untuk mendeklarasikan/ memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- const int TABLE_SIZE = 11; ➔ mendefinisikan ukuran tabel hash sebagai konstanta
- string name; ➔ deklarasi variabel global untuk menyimpan nama
- string phone_number; ➔ deklarasi variabel global untuk menyimpan nomor telepon
- class HashNode { ➔ mendefinisikan kelas hashnode
- public: ➔ mendefinisikan anggota publik
- string name; ➔ deklarasi variabel untuk menyimpan nama
- string phone_number; ➔ deklarasi variabel untuk menyimpan nomor telepon
- HashNode(string name, string phone_number) { ➔ struktur kelas hashnode
- this->name = name; ➔ menginisialisasi nama
- this->phone_number = phone_number; ➔ menginisialisasi nomor telepon
- class HashMap { ➔ mendefinisikan kelas hashmap
- private: ➔ mendefinisikan anggota privat
- vector<HashNode*> table[TABLE_SIZE]; ➔ membuat array of vector dengan ukuran table_size
- public: ➔ mendefinisikan anggota publik
- int hashFunc(string key) { ➔ fungsi hash untuk mengubah kunci menjadi indeks tabel

- int hash_val = 0; ➔ inisialisasi nilai hash
- for (char c : key) { ➔ perulangan untuk setiap karakter di kunci
- hash_val += c; ➔ menambahkan nilai asli karakter ke nilai hash
- return hash_val % TABLE_SIZE; ➔ mengembalikan indeks tabel dengan modula ukuran tabel
- void insert(string name, string phone_number) { //masukin nama dan no telp ➔ fungsi untuk menambahkan data ke tabel hash
- int hash_val = hashFunc(name); ➔ menghitung nilai hash dari nama
- for (auto node : table[hash_val]) { ➔ perulangan untuk setiap noda di indeks tabel yang dihasilkan
- if (node->name == name) { ➔ percabangan if jika nama sudah ada di tabel
- node->phone_number = phone_number; ➔ mengupdate nomor telepon
- return; ➔ keluar dari fungsi
- table[hash_val].push_back(new HashNode(name, phone_number)); ➔ jika nama belum ada tambahkan data baru ke tabel
- void remove(string name) { ➔ prosedur untuk menghapus data dari tabel
- int hash_val = hashFunc(name); ➔ menghitung nilai hash dari nama
- for (auto it = table[hash_val].begin(); it != table[hash_val].end(); it++) { ➔ perulangan untuk setiap node di indeks tabel yang dihasilkan
- if ((*it)->name == name) { ➔ percabangan if jika nama ditemukan
- table[hash_val].erase(it); ➔ hapus data dari tabel
- return; ➔ keluar dari fungsi
- string searchByName(string name) { // cari suatu data

berdasarkan nama ➔ fungsi untuk mencari data berdasarkan nama

- int hash_val = hashFunc(name); ➔ menghitung nilai hash dari nama
- for (auto node : table[hash_val]) { ➔ perulangan untuk setiap noda di indeks tabel yang dihasilkan
- if (node->name == name) { ➔ percabangan if jika nama ditemukan
- return node->phone_number; ➔ mengembalikan nomor telepon
- return ""; ➔ jika nama tidak ditemukan kembalikan string kosong
- void print() { // menampilkan tabel size dan value dari hash ➔ prosedur untuk mencetak semua data
- for (int i = 0; i < TABLE_SIZE; i++) { ➔ perulangan untuk setiap indeks tabel
- cout << i << ": "; ➔ cout mencetak indeks tabel
- for (auto pair : table[i]) { ➔ perulangan untuk setiap pasangan di indeks tabel
- if(pair != nullptr){ ➔ percabangan if jika pasangan tidak null
- cout << "[" << pair->name << ", " << pair->phone_number << "]";➔ chord mencetak nama dan nomor telepon
- cout << endl; ➔ cout mencetak baris baru
- int main() { ➔ fungsi utama untuk mengeksekusi program
- HashMap employee_map; ➔ membuat objek employee_map dari kelas hashmap
- employee_map.insert("Mistah", "1234"); ➔ memasukkan data ke tabel hash
- employee_map.insert("Pastah", "5678"); ➔ memasukkan data ke tabel hash
- employee_map.insert("Ghana", "91011"); ➔ memasukkan data ke tabel hash

- cout << "Nomer Hp Mistah : " <<employee_map.searchByName("Mistah") << endl; // di dalam employee map keluarnya Mistah ➔ mencari dan mencetak nomor telepon Mistah
- cout << "Phone Hp Pastah : " <<employee_map.searchByName("Pastah") << endl; ➔ mencari dan mencetak nomor telepon Pastah
- employee_map.remove("Mistah"); ➔ menghapus data Mistah dari tabel hash
- cout << "Nomer Hp Mistah setelah dihapus : " <<employee_map.searchByName("Mistah") << endl << endl; ➔ mencari dan mencetak nomor telepon Mistah setelah dihapus
- cout << "Hash Table : " << endl; ➔ cout mencetak judul tabel hash
- employee_map.print(); ➔ mencetak semua data dalam tabel hash
- return 0; ➔ program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

B. Unguided/Tugas

1. 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 <string>
#include <vector>
#include <iomanip>
#include <limits>

using namespace std;

const int TABLE_SIZE = 11;

class Mahasiswa
{
public:
    string NIM;
    string nama;
    int nilai;
    Mahasiswa (string NIM, string nama,
int nilai)
    {
            this->NIM = NIM;
            this->nama = nama;
            this->nilai = nilai;
    }
```

```
};

class HashMap
{
private:
    vector<Mahasiswa*> tabel[TABLE_SIZE];
public:
    int hashFunc (string key)
    {
        int hash_val = 0;
        for (char c : key)
        {
            hash_val += c;
        }
        return hash_val % TABLE_SIZE;
    }
void tambahData(string NIM, string nama,
int nilai)
{
    int hash_val = hashFunc(NIM);
    for (auto node : tabel[hash_val])
    {
        if (node->NIM == NIM)
        {
            node->nama = nama;
            node->nilai = nilai;
            return;
        }
    }
    tabel[hash_val].push_back(new
Mahasiswa(NIM, nama, nilai));
}
void hapusData (string NIM)
{
    int hash_val = hashFunc(NIM);
    bool found = false;
    for (auto it =
tabel[hash_val].begin(); it !=
tabel[hash_val].end(); it++)
    {
        if ((*it)->NIM == NIM)
        {
            tabel [hash_val].erase(it);
            found = true;
            break;
        }
```

```
    }
    if (! found)
    {
        cout << "Data tidak ditemukan.\n";
    }
}
    Mahasiswa* cariBerdasarkanNIM(string
NIM)
    {
        int hash_val = hashFunc(NIM);
        for (auto node : tabel[hash_val])
            {
                if (node->NIM == NIM)
                {
                    return node;
                }
            }
            return nullptr;
    }
void cariBerdasarkanNilai (int nilai)
{
    bool found = false;
    if (nilai < 80 || nilai > 90)
    {
        cout << "Nilai di luar rentang
yang ditentukan (80-90).\n";
        return;
    }
    cout << "Mahasiswa dengan nilai: " <<
nilai << "\n";
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        for (auto node : tabel[i])
        {
            if (node != nullptr && node-
>nilai == nilai)
            {
                cout << "NIM: " << node-
>NIM << ", Nama: " << node->nama << endl;
                found = true;
            }
        }
    }
    if (!found)
    {
        cout << "Data tidak ditemukan.\n";
```

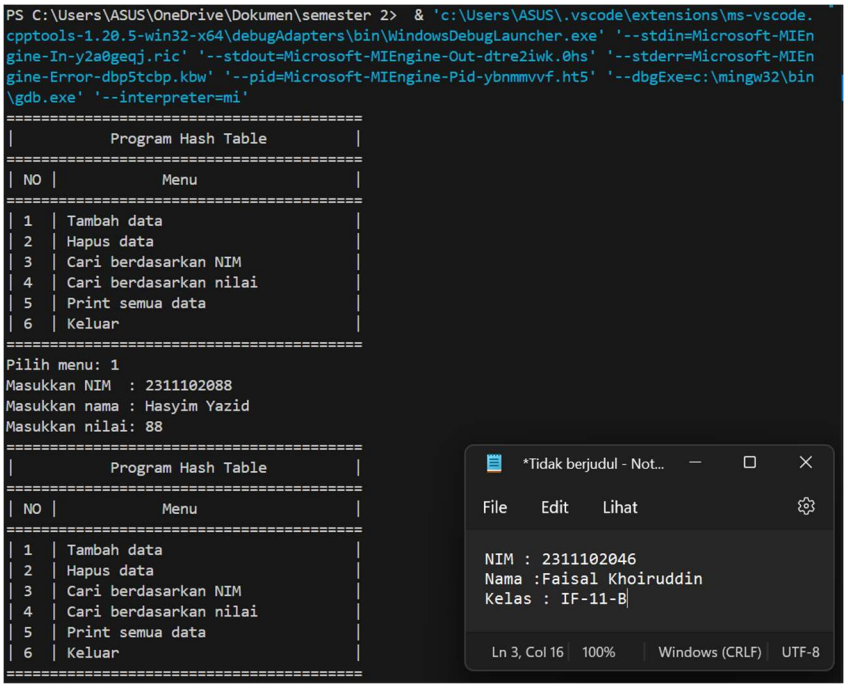
```
    }
  }
void print ()
{
    cout <<
"========================================
=========\n";
    cout << "| " << left << setw(10) <<
"NIM" << " | " << left << setw(25) <<
"Nama" << " | " << left << setw(5) <<
"Nilai" << " |\n";
    cout <<
"========================================
=========\n";
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        for (auto node : tabel[i])
        {
            if (node != nullptr)
            {
                cout << "| " << left <<
setw(10) << node->NIM << " | " << left <<
setw(25) << node->nama << " | " << left <<
setw(5) << node->nilai << " |\n";
            }
        }

    }
    cout <<
"========================================
=========\n\n";
  }
};

int main()
{
    HashMap dataMahasiswa;
    int pilihan;
    string NIM;
    string nama;
    int nilai;
    Mahasiswa* mhs;

    while (true)
    {
        cout <<
```

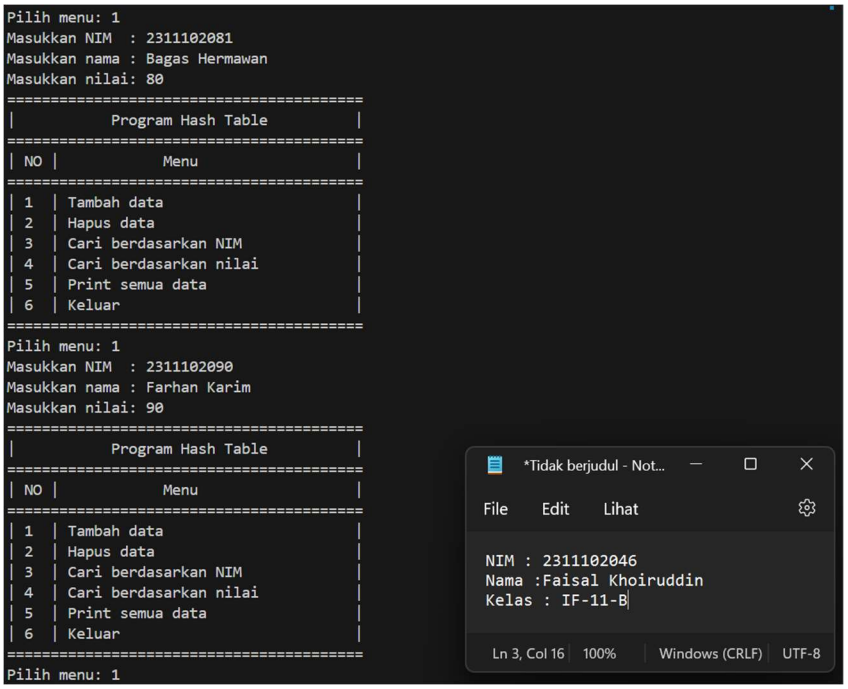
```
"========================================
\n";
        cout << "|           Program Hash
Table          |\n";
        cout <<
"========================================
\n";
        cout << "| NO
|            Menu                    |\n";
        cout <<
"========================================
\n";
        cout << "| 1  | Tambah
data                          |\n";
        cout << "| 2  | Hapus
data                            |\n";
        cout << "| 3  | Cari berdasarkan
NIM             |\n";
        cout << "| 4  | Cari berdasarkan
nilai           |\n";
        cout << "| 5  | Print semua
data                   |\n";
        cout << "| 6  |
Keluar                               |\n";
        cout <<
"========================================
\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan)
        {
            case 1:
                cout << "Masukkan NIM  :
";
                cin >> NIM;
                cout << "Masukkan nama :
";
                cin.ignore();
                getline(cin, nama);
                cout << "Masukkan nilai:
";
                cin >> nilai;
                dataMahasiswa.tambahData(N
IM, nama, nilai);
                break;
```

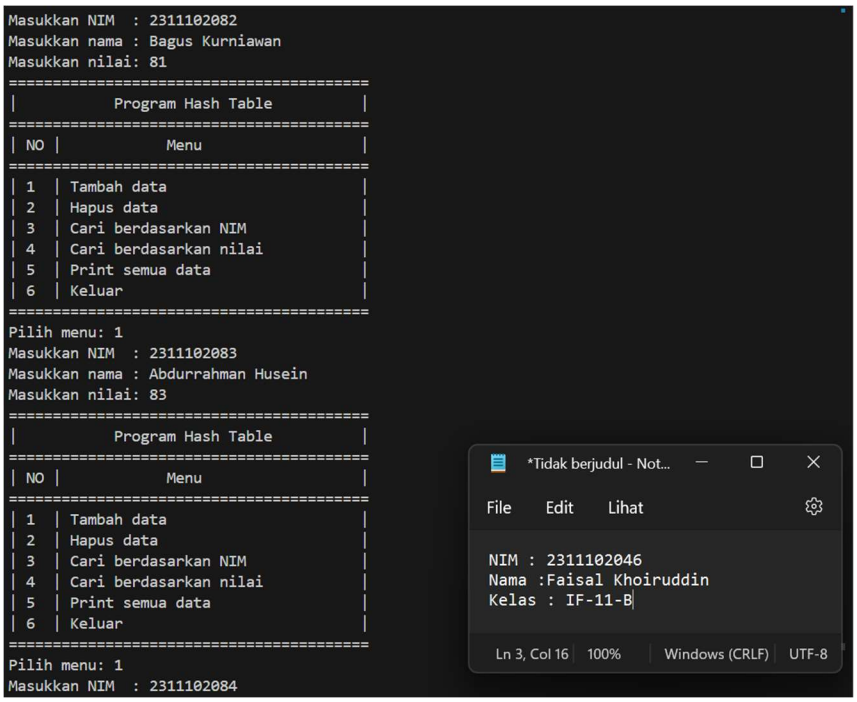
```
            case 2:
                cout << "Masukkan NIM  :
";
                cin >> NIM;
                dataMahasiswa.hapusData(NI
M);
                break;
            case 3:
                cout << "Masukkan NIM  :
";
                cin >> NIM;
                mhs =
dataMahasiswa.cariBerdasarkanNIM(NIM);
                if (mhs == nullptr)
                {
                    cout << "Data tidak
ditemukan\n";
                }
                else
                {
                    cout << "Nama : " <<
mhs->nama << ", Nilai: " << mhs->nilai <<
endl;
                }
                break;
            case 4:
                cout << "Masukkan nilai :
";
                cin >> nilai;
                dataMahasiswa.cariBerdasar
kanNilai(nilai);
                break;
            case 5:
                dataMahasiswa.print();
                break;
            case 6:
                cout << "Terima kasih
telah menggunakan program ini\n";
                return 0;
            default:
                cout << "Pilihan tidak
valid. Silahkan masukkan angka antara 1
sampai 6.\n";
                cin.clear();
                cin.ignore(numeric_limits<
streamsize>::max(), '\n');
```

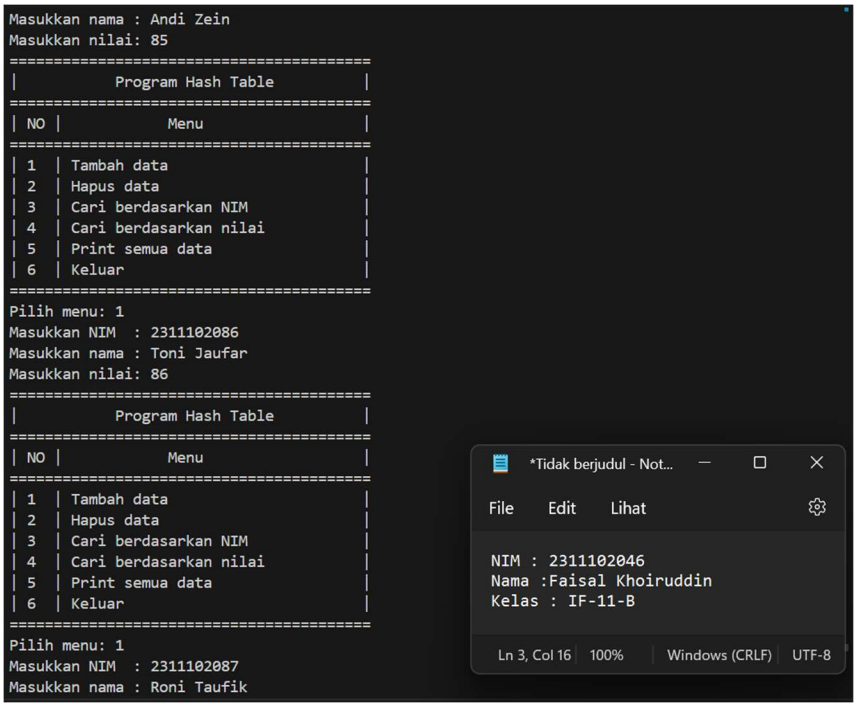
```
            }
        }
        return 0;
}
```

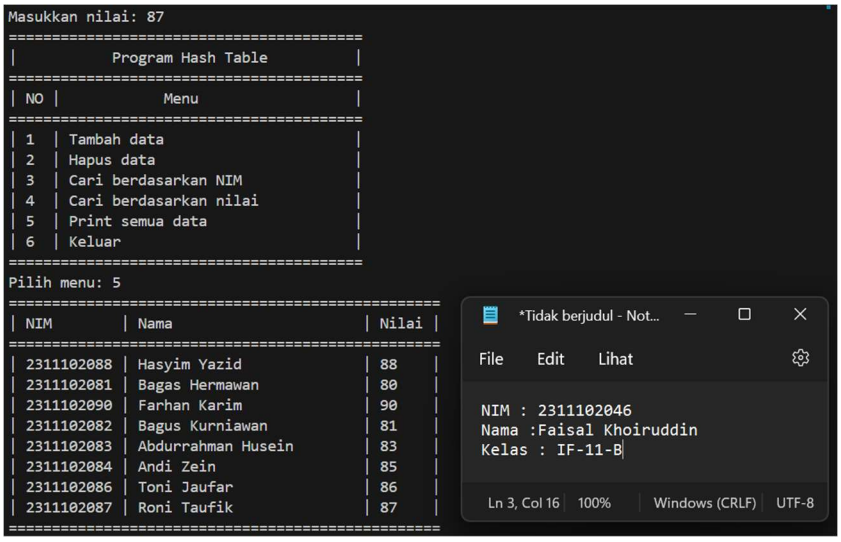
Screenshots Output

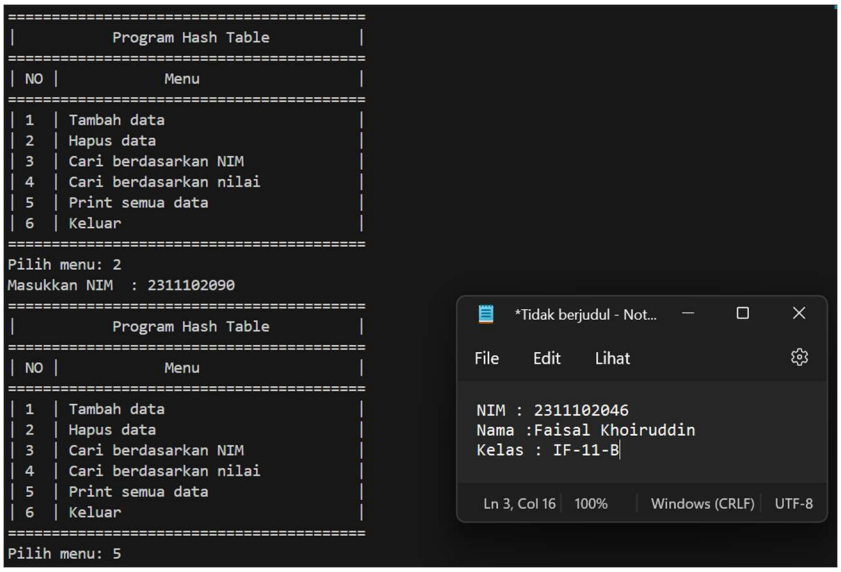


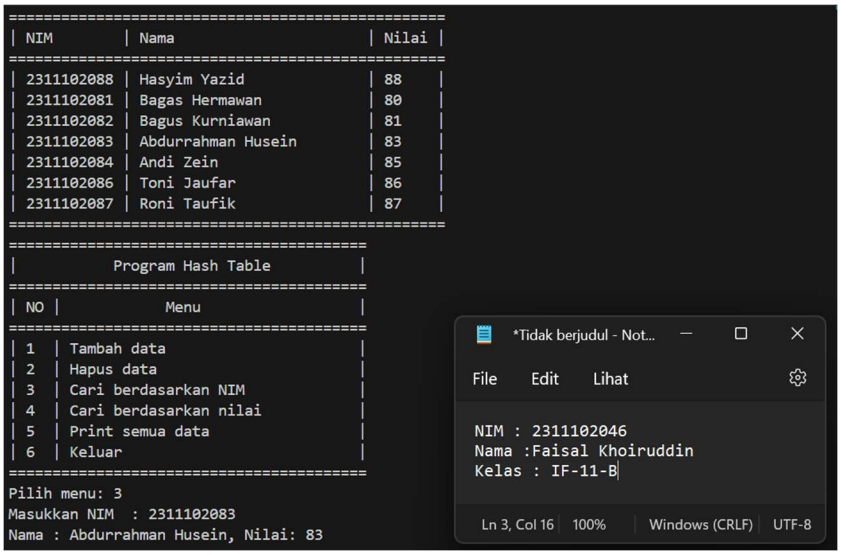

Pilih menu: 1
Masukkan NIM : 2311102081
Masukkan nama : Bagas Hermawan
Masukkan nilai: 80
Program Hash Table
NO
Menu
1 Tambah data
2 Hapus data
3 Cari berdasarkan NIM
4 Cari berdasarkan nilai
5 Print semua data
6 Keluar
Pilih menu: 1
Masukkan NIM : 2311102090
Masukkan nama : Farhan Karim
Masukkan nilai: 90
Program Hash Table
NO
Menu
1 Tambah data
2 Hapus data
3 Cari berdasarkan NIM
4 Cari berdasarkan nilai
5 Print semua data
6 Keluar
Pilih menu: 1
*Tidak berjudul - Not...
File
Edit
Lihat
NIM : 2311102046
Nama :Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16
100%
Windows (CRLF)
UTF-8

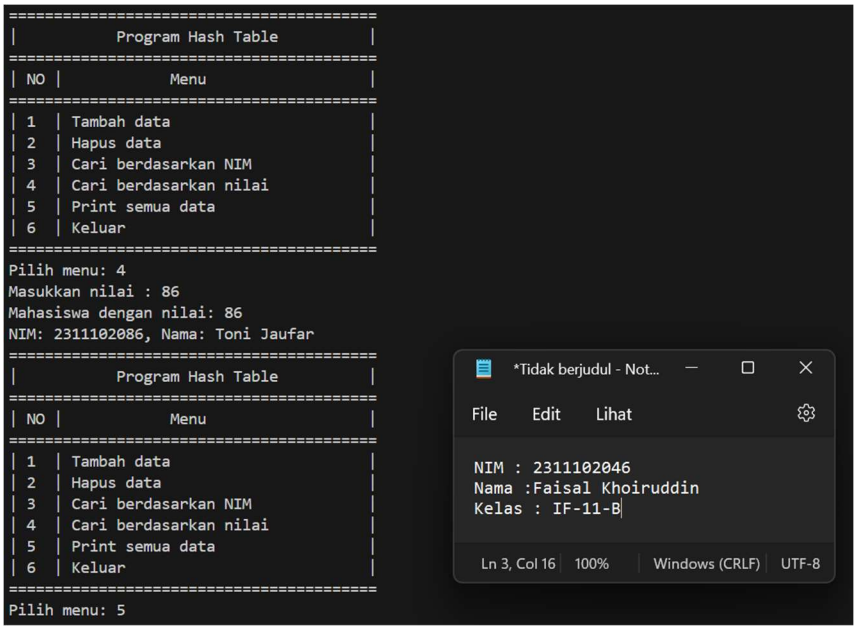

Masukkan NIM : 2311102082
Masukkan nama : Bagus Kurniawan
Masukkan nilai: 81
Program Hash Table
NO
Menu
1 Tambah data
2 Hapus data
3 Cari berdasarkan NIM
4 Cari berdasarkan nilai
5 Print semua data
6 Keluar
Pilih menu: 1
Masukkan NIM : 2311102083
Masukkan nama : Abdurrahman Husein
Masukkan nilai: 83
Program Hash Table
NO
Menu
1 Tambah data
2 Hapus data
3 Cari berdasarkan NIM
4 Cari berdasarkan nilai
5 Print semua data
6 Keluar
Pilih menu: 1
Masukkan NIM : 2311102084
*Tidak berjudul - Not...
File
Edit
Lihat
NIM : 2311102046
Nama :Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16
100%
Windows (CRLF)
UTF-8

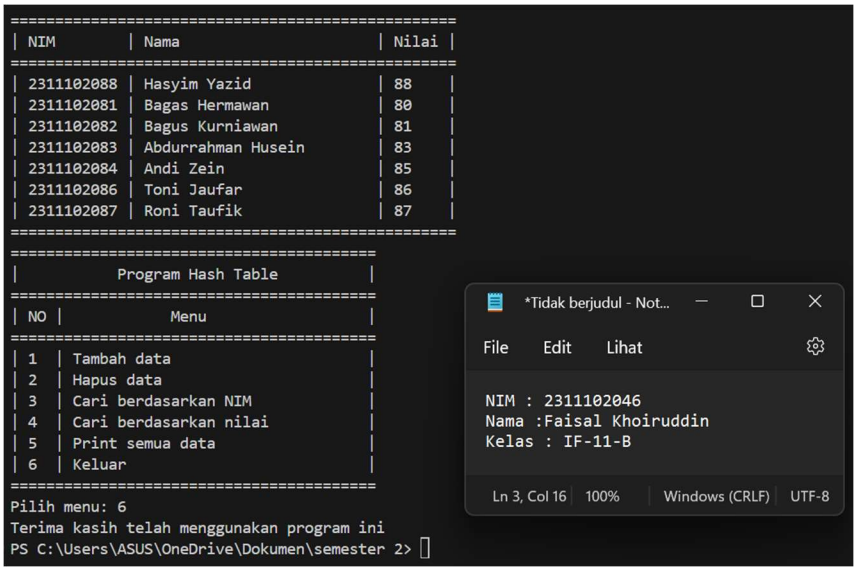

Masukkan nama : Andi Zein
Masukkan nilai: 85
| Program Hash Table |
NO	Menu
1	Tambah data
2	Hapus data
3	Cari berdasarkan NIM
4	Cari berdasarkan nilai
5	Print semua data
6	Keluar
Pilih menu: 1
Masukkan NIM : 2311102086
Masukkan nama : Toni Jaufar
Masukkan nilai: 86
Program Hash Table
NO	Menu
1	Tambah data
2	Hapus data
3	Cari berdasarkan NIM
4	Cari berdasarkan nilai
5	Print semua data
6	Keluar
Pilih menu: 1
Masukkan NIM : 2311102087
Masukkan nama : Roni Taufik
*Tidak berjudul - Not...
File Edit Lihat
NIM : 2311102046
Nama :Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16 100% Windows (CRLF) UTF-8


Masukkan nilai: 87
| Program Hash Table |
NO	Menu
1	Tambah data
2	Hapus data
3	Cari berdasarkan NIM
4	Cari berdasarkan nilai
5	Print semua data
6	Keluar
Pilih menu: 5
NIM	Nama	Nilai
2311102088	Hasyim Yazid	88
2311102081	Bagas Hermawan	80
2311102090	Farhan Karim	90
2311102082	Bagus Kurniawan	81
2311102083	Abdurrahman Husein	83
2311102084	Andi Zein	85
2311102086	Toni Jaufar	86
2311102087	Roni Taufik	87
*Tidak berjudul - Not...
File Edit Lihat
NIM : 2311102046
Nama :Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16 100% Windows (CRLF) UTF-8


==========
| Program Hash Table |
==========
| NO | Menu |
==========
1	Tambah data
2	Hapus data
3	Cari berdasarkan NIM
4	Cari berdasarkan nilai
5	Print semua data
6	Keluar
==========
Pilih menu: 2
Masukkan NIM : 2311102090
==========
| Program Hash Table |
==========
| NO | Menu |
==========
1	Tambah data
2	Hapus data
3	Cari berdasarkan NIM
4	Cari berdasarkan nilai
5	Print semua data
6	Keluar
==========
Pilih menu: 5
*Tidak berjudul - Not...
File Edit Lihat
NIM : 2311102046
Nama :Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16 100% Windows (CRLF) UTF-8


==========
| NIM | Nama | Nilai |
==========
2311102088	Hasyim Yazid	88
2311102081	Bagas Hermawan	80
2311102082	Bagus Kurniawan	81
2311102083	Abdurrahman Husein	83
2311102084	Andi Zein	85
2311102086	Toni Jaufar	86
2311102087	Roni Taufik	87
==========
==========
| Program Hash Table |
==========
| NO | Menu |
==========
1	Tambah data
2	Hapus data
3	Cari berdasarkan NIM
4	Cari berdasarkan nilai
5	Print semua data
6	Keluar
==========
Pilih menu: 3
Masukkan NIM : 2311102083
Nama : Abdurrahman Husein, Nilai: 83
*Tidak berjudul - Not...
File Edit Lihat
NIM : 2311102046
Nama :Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16 100% Windows (CRLF) UTF-8





Deskripsi:

Program tersebut program menu Hash Table. Program tersebut untuk menyimpan dan mengelola data mahasiswa, dengan menggunakan input dari user. Pengguna diminta memilih operasi dengan memasukkan angka dari menu Tambah Data, Hapus Data, mencari data berdasarkan nim, mencari data berdasarkan nilai, mencetak semua data dan keluar. Output program tersebut yaitu menampilkan nim, nama, dan nilai mahasiswa.

- #include <iostream> ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- #include <string> ➔ untuk memanipulasi data bertipe string
- #include <vector> ➔ vektor menyimpan elemen jenis tertentu dalam pengaturan linier, dan memungkinkan akses acak cepat ke elemen apa pun
- #include <iomanip> ➔ stantdar untuk menentukan beberapa manipulator yang masing-masing mengambil satu argumen
- #include <limits> ➔ digunakan untuk memasukkan file header limits ke dalam program yang berisi konstanta dan fungsi
- using namespace std; ➔ digunakan untuk mendeklarasikan/ memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- const int TABLE_SIZE = 11; ➔ mendefinisikan ukuran table hash sebagai konstanta
- class Mahasiswa ➔ mendefinisikan class mahasiswa
- public: ➔ mendefinisikan anggota publik
- string NIM; ➔ variabel untuk menyimpan nim mahasiswa
- string nama; ➔ variabel untuk menyimpan nama mahasiswa
- int nilai; ➔ variabel untuk menyimpan nilai mahasiswa
- Mahasiswa (string NIM, string nama, int nilai) ➔ konstruktor kelas mahasiswa
- this->NIM = NIM; ➔ inisialisasi nim
- this->nama = nama; ➔ inisialisasi nama
- this->nilai = nilai; ➔ inisialisasi nilai
- class HashMap ➔ mendefinisikan class hashmap
- private: ➔ mendefinisikan anggota private
- vector<Mahasiswa*> tabel[TABLE_SIZE]; ➔ membuat

array of ventor dengan ukuran table_size

- public: ➔ mendefinisikan anggota publik
- int hashFunc (string key) ➔ fungsi hash untuk mengubah kunci menjadi indeks tabel
- int hash_val = 0; ➔ inisialisasi nilai hash
- for (char c : key) ➔ perulangan untuk setiap karakter di kunci
- hash_val += c; ➔ menambahkan nilai ascii karakter ke nilai hash
- return hash_val % TABLE_SIZE; ➔ mengembalikan indeks table dengan modulo ukuran tabel
- void tambahData(string NIM, string nama, int nilai) ➔ fungsi untuk menambahkan data ke table hash
- int hash_val = hashFunc(NIM); ➔ menghitung nilai hash dari nim
- for (auto node : tabel[hash_val]) ➔ perulangan untuk setiap node di indeks table yang dihasilkan
- if (node->NIM == NIM) ➔ Jika NIM sudah ada di tabel
- node->nama = nama; ➔ mengupdate nama
- node->nilai = nilai; ➔ mengupdate nilai
- return; ➔ keluar dari fungsi
- tabel[hash_val].push_back(new Mahasiswa(NIM, nama, nilai)); ➔ jika NIM belum ada tambahkan data baru ke tabel
- void hapusData (string NIM) ➔ prosedur untuk menghapus data dari tabel hash
- int hash_val = hashFunc(NIM); ➔ menghitung nilai hash dari NIM
- bool found = false; ➔ variabel untuk mengecek apakah data ditemukan atau tidak
- for (auto it = tabel[hash_val].begin(); it != tabel[hash_val].end(); it++) ➔ perulangan untuk setiap noda di indeks tabel yang dihasilkan

- if ((*it)->NIM == NIM) ➔ percabangan if jika NIM ditemukan
- tabel [hash_val].erase(it); ➔ hapus data dari tabel
- found = true; ➔ mengatur variabel found menjadi true
- break; ➔ keluar dari perulangan
- if (! found) ➔ percabangan if jika data tidak ditemukan
- cout << "Data tidak ditemukan.\n"; ➔ cout menampilkan cetak pesan error
- Mahasiswa* cariBerdasarkanNIM(string NIM) ➔ fungsi untuk mencari data berdasarkan NIM
- int hash_val = hashFunc(NIM); ➔ menghitung nilai hash dari NIM
- for (auto node : tabel[hash_val]) ➔ Perulangan untuk setiap node di indeks tabel yang dihasilkan
- if (node->NIM == NIM) ➔ Percabangan if jika NIM ditemukan
- return node; ➔ mengembalikan data
- return nullptr; ➔ jika data tidak ditemukan kembalikan null
- void cariBerdasarkanNilai (int nilai). ➔ Prosedur untuk mencari data berdasarkan nilai
- bool found = false; ➔ variabel untuk mengecek apakah data ditemukan atau tidak
- if (nilai < 80 || nilai > 90) ➔ percabangan if jika nilai di luar rentang 80 sampai 90
- cout << "Nilai di luar rentang yang ditentukan (80-90).\n"; ➔ cout mencetak pesan error
- return; ➔ keluar dari fungsi
- cout << "Mahasiswa dengan nilai: " << nilai << "\n"; ➔ cout menampilkan nilai yang dicari
- for (int i = 0; i < TABLE_SIZE; i++) ➔ perulangan untuk setiap indeks tabel

- for (auto node : tabel[i]) ➔ perulangan untuk setiap node di indeks tabel
- if (node != nullptr && node->nilai == nilai) ➔ percabangan if jika node tidak null dan nilai sama dengan nilai yang dicari
- cout << "NIM: " << node->NIM << ", Nama: " << node->nama << endl; ➔ cout menampilkan NIM dan nama
- found = true; ➔ mengatur variabel found menjadi true
- if (!found) ➔ percabangan if jika data tidak ditemukan
- cout << "Data tidak ditemukan.\n"; ➔ cout menampilkan pesan data tidak ditemukan
- void print () ➔ prosedur untuk mencetak semua data
- cout << "==========\n"; ➔ cout menampilkan garis pembatas
- cout << "| " << left << setw(10) << "NIM" << " | " << left << setw(25) << "Nama" << " | " << left << setw(5) << "Nilai" ➔ cout menampilkan header tabel
- cout << "==========\n"; ➔ cout menampilkan garis pembatas
- for (int i = 0; i < TABLE_SIZE; i++). ➔ perulangan untuk setiap indeks tabel
- for (auto node : tabel[i]) ➔ perulangan untuk setiap node di indeks tabel
- if (node != nullptr) ➔ percabangan if jika mode tidak kosong
- cout << "| " << left << setw(10) << node->NIM << " | " << left << setw(25) << node->nama << " | " << left << setw(5) << node->nilai << " |\n"; ➔ cout menampilkan NIM nama dan nilai
- cout << "==========\n\n"; ➔ cout menampilkan garis pembatas
- int main() ➔ merupakan fungsi utama

- HashMap dataMahasiswa; ➔ Membuat objek data mahasiswa dari kelas HashMap
- int pilihan; ➔ variabel untuk menyimpan pilihan menu
- string NIM; ➔ variabel untuk menyimpan NIM
- string nama; ➔ variabel untuk menyimpan nama
- int nilai; ➔ variabel untuk menyimpan nilai dengan tipe data integer
- Mahasiswa* mhs; ➔ pointer untuk menyimpan data mahasiswa
- while (true) ➔ perulangan selamat dalam kondisi true
- cout << "==========\n" ➔ cout menampilkan garis pembatas
- cout << "| Program Hash Table |\n"; ➔ cout menampilkan judul program
- cout << "==========\n" ➔ cout menampilkan garis pembatas
- cout << "| NO | Menu |\n"; ➔ cout menampilkan nomor dan menu
- cout << "==========\n" ; ➔ cout menampilkan garis pembatas
- cout << "| 1 | Tambah data |\n"; ➔ cout menampilkan menu nomor satu yaitu tambah data
- cout << "| 2 | Hapus data |\n"; ➔ cout menampilkan pilihan menu nomor 2 yaitu hapus data
- cout << "| 3 | Cari berdasarkan NIM |\n"; ➔ cout menampilkan pilihan menu nomor 3 yaitu cari berdasarkan NIM
- cout << "| 4 | Cari berdasarkan nilai |\n"; ➔ cout menampilkan pilihan menu nomor 4 yaitu cari berdasarkan nilai

- cout << "| 5 | Print semua data |\n"; ➔ cout menampilkan menu pilihan nomor 5 yaitu print semua data
- cout << "| 6 | Keluar |\n"; ➔ cout menampilkan menu pilihan 6 yaitu keluar
- cout << "==========\n"; ➔ cout menampilkan garis pembatas
- cout << "Pilih menu: "; ➔ cout menampilkan pilih menu
- cin >> pilihan; ➔ menerima inputan pilihan menu dari pengguna dan menyimpannya pada variabel pilihan
- switch (pilihan) ➔ switch case berdasarkan pilihan menu
- case 1: ➔ jika pilihan menu adalah 1 yaitu tambah data
- cout << "Masukkan NIM : "; ➔ cout menampilkan masukan NIM
- cin >> NIM; ➔ menerima inputan dari pengguna dan menyimpannya ke dalam variabel NIM
- cout << "Masukkan nama : "; ➔ cout menampilkan masukan nama
- cin.ignore(); ➔ mengabaikan karakter newline di buffer
- getline(cin, nama); ➔ menerima input nama
- cout << "Masukkan nilai: "; ➔ cout menampilkan masukan nilai
- cin >> nilai; ➔ menerima input dari pengguna dan menyimpan ke variabel nilai
- dataMahasiswa.tambahData(NIM, nama, nilai); ➔ menambahkan data ke tabel hash
- break; ➔ keluar dari case 1
- case 2: ➔ jika pilihan adalah dua yaitu hapus data
- cout << "Masukkan NIM : "; ➔ cout menampilkan masukan NIM
- cin >> NIM; ➔ menerima input dari pengguna dan

menyimpan ke variabel NIM

- dataMahasiswa.hapusData(NIM); ➔ menghapus data dari tabel hash
- break; ➔ keluar dari case 2
- case 3: ➔ jika pilihan adalah 3 yaitu cari berdasarkan NIM
- cout << "Masukkan NIM : "; ➔ cout menampilkan masukan NIM
- cin >> NIM; ➔ menerima input NIM dari pengguna dan menyimpan ke dalam variabel NIM
- mhs = dataMahasiswa.cariBerdasarkanNIM(NIM); ➔ mencari data berdasarkan NIM
- if (mhs == nullptr) ➔ percabangan if jika data tidak ditemukan
- cout << "Data tidak ditemukan\n"; ➔ cout menampilkan data tidak ditemukan
- else ➔ jika data ditemukan
- cout << "Nama : " << mhs->nama << ", Nilai: " << mhs->nilai << endl; ➔ cout menampilkan nama dan nilai
- break; ➔ keluar dari case 3
- case 4: ➔ jika pilihan adalah 4 yaitu cari berdasarkan nilai
- cout << "Masukkan nilai : "; ➔ cout menampilkan masukan nilai
- cin >> nilai; ➔ menerima input berupa nilai dari pengguna dan menyimpan ke dalam variabel nilai
- dataMahasiswa.cariBerdasarkanNilai(nilai); ➔ mencari data berdasarkan nilai
- break; ➔ keluar dari case 4
- case 5: ➔ jika pilihan adalah 5 yaitu print semua data
- dataMahasiswa.print(); ➔ mencetak semua data
- break; ➔ keluar dari case 5
- case 6: ➔ jika pilihan adalah 6 yaitu keluar

- cout << "Terima kasih telah menggunakan program ini\n"; ➔ cout menampilkan terima kasih telah menggunakan program ini
- return 0; ➔ keluar dari program
- default: ➔ jika pilihan tidak valid
- cout << "Pilihan tidak valid. Silahkan masukkan angka antara 1 sampai 6.\n"; ➔ cout menampilkan pesan tidak valid
- cin.clear(); ➔ membersihkan error flag pada cin
- cin.ignore(numeric_limits<streamsize>::max(), '\n'); ➔ mengabaikan karakter newline di buffer
- return 0; ➔ program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

C. Kesimpulan

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. 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.

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.

D. Referensi

[1] Asisten Praktikum, Struktur Data “ARRAY", WhatsApp, 2024.

[2] Dr. Jesper Jansson, Key Concepts, Weakness and Benchmark on Hash Table Data Structures, diakses dari

https://www.mdpi.com/1999-4893/15/3/100

[3] BobTylerMSFT.(n.d.).<vector>. diakses dari

https://learn.microsoft.com/id-id/cpp/standard-library/vector-class?view=msvc-170