# Belajaritma

## Data Structure

## Sesi VII : Hashes
## (PDF Example)

Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY

# DAFTAR ISI

Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY

# 1. TUJUAN

- Memahami dan mengetahui materi Hashing and Hash Tables
- Menerapkan konsep pembuatan *hash tables* sederhana
- Mengetahui cara mengatasi *collision* pada sebuah *hash table*

# 2. PENDAHULUAN

Secara umum, *data structure* merupakan sebuah metode atau cara untuk menyusun kolom-kolom yang berisi data, lalu setiap kolom yang berisi data disebut record. Ukuran dari setiap kolom bisa diatur baik secara dinamis (ukuran dapat diubah tergantung input) atau tetap. *Data structure* ini dapat dimanfaatkan untuk mengolah database dan *word processor.* Secara garis besar *data structure* terbagi menjadi 2, yaitu:

a. Tipe Data Sederhana, yaitu:
    - Data Sederhana Tunggal (contohnya: karakter, integer, real, dan boolean).
    - Data Sederhana Majemuk (contohnya: string).
b. Struktur Data
    - Struktur Data Sederhana (contohnya: record dan array).
    - Struktur Data majemuk, yaitu:
        - Linier/lurus, (contohnya: stack, queue, dan linked list).
        - Nonlinier/bercabang, (contohnya: binary tree, graph, binary search tree, dll).

# 3. ISI

## 3.1 TEORI

### 3.1.1. Hashing

Merupakan proses pengubahan sebuah objek data menjadi nilai yang lebih pendek atau juga bisa disebut dengan *key.* Biasanya, hashing digunakan agar objek data bisa lebih mudah untuk ditemukan di dalam *database.*

### 3.1.2. Hash table

Hash table adalah sebuah struktur data array asosiatif yang berbentuk tabel untuk menyimpan suatu data/*record* dengan suatu nilai unik sebagai alamatnya (disebut “key”). Singkatnya, hash table adalah *array*-nya dan nilai unik tersebut adalah *index*-nya. Key tersebut dapat diatur melalui jumlah *index* dari (0) sampai (*index*-1). Misalnya, jika key diambil dari huruf pertama sebuah string, maka indexnya ada 26, dari A yaitu 0 sampai Z yaitu 25. Tetapi

dengan penyimpanan seperti itu, akan memungkinkan terjadinya *collision* (tabrakan) jika ditemukan key yang sama.

### 3.1.3. Hash Function

Hashing memiliki beberapa metode atau fungsi untuk merubah sebuah objek menjadi *key*, seperti:

| hash table | |
|---|---|
| key | value |
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |

1. Mid-square

   Metode yang menggunakan *seed value* (nilai awal) untuk dikuadratkan dan nantinya nilai tengah dari hasil kuadrat tersebut akan diambil untuk dijadikan *hash key* (nilai hash). Contoh:

   | For example, | | |
   |---|---|---|
   | Key | squared value | middle part |
   | 3121 | 9740641 | 406 |

   Dari hasil kuadrat nilai awal 3121 yaitu 9740641, akan diambil angka 406 sebagai *hash key*-nya.

2. Division

Metode yang menggunakan seed value (nilai awal) untuk dimodulus dengan suatu angka (bebas, tapi disarankan untuk menggunakan angka prima ataupun *table size*) dan nantinya hasil modulus tersebut akan dijadikan *hash key* (nilai hash).

Contoh:

Nilai awal: 5758

*table size*: 9

*Hash key* yang didapat adalah 7, dikarenakan 7 merupakan hasil modul dari 5758 dengan 9.

3. Folding

Metode yang menggunakan *seed value* (nilai awal) untuk nantinya dipecah menjadi beberapa bagian dan bagian-bagian tersebut akan dijumlahkan menjadi sebuah angka yang baru. Kemudian, angka tersebut akan dimodulus dengan jumlah lokasi dari *hash tables* sehingga hasil akhirnya akan menjadi *hash key*.

Contoh:

| Key | 5678 |
|---|---|
| Parts | 56 and 78 |
| Sum | 134 |
| Hash Value | 34 (ignore the last carry) |

Dari yang awalnya memiliki nilai 5678, akan dipecah menjadi bagian 1 (56) dan bagian 2 (78). Kedua bagian ini akan dijumlahkan yang hasilnya berupa angka 134, lalu apabila disediakan *size table* sebesar 100, maka *hash key*-nya adalah 34 (dari 134 angka 1-nya diabaikan).

4. Digit Extraction

Metode yang menggunakan *seed value* (nilai awal) untuk nantinya dipecah menjadi beberapa digit dan akan diambil berdasarkan urutan digit yang telah ditentukan. Setelah itu, digit yang telah diambil akan disatukan dan dijadikan sebagai *hash key*.

Contoh:

Nilai awal: 13589

Digit yang ingin diambil: urutan ke-1, ke-3, dan ke-5 dari nilai awal

Maka, *hash key* yang didapat adalah 159.

5. Rotating Hash

   Metode yang menggunakan *hash key* yang didapat dari metode hashing lainnya untuk dibalik menjadi *hash key* yang baru.

   Contoh:

   *hash key* dari Mid-square: 406

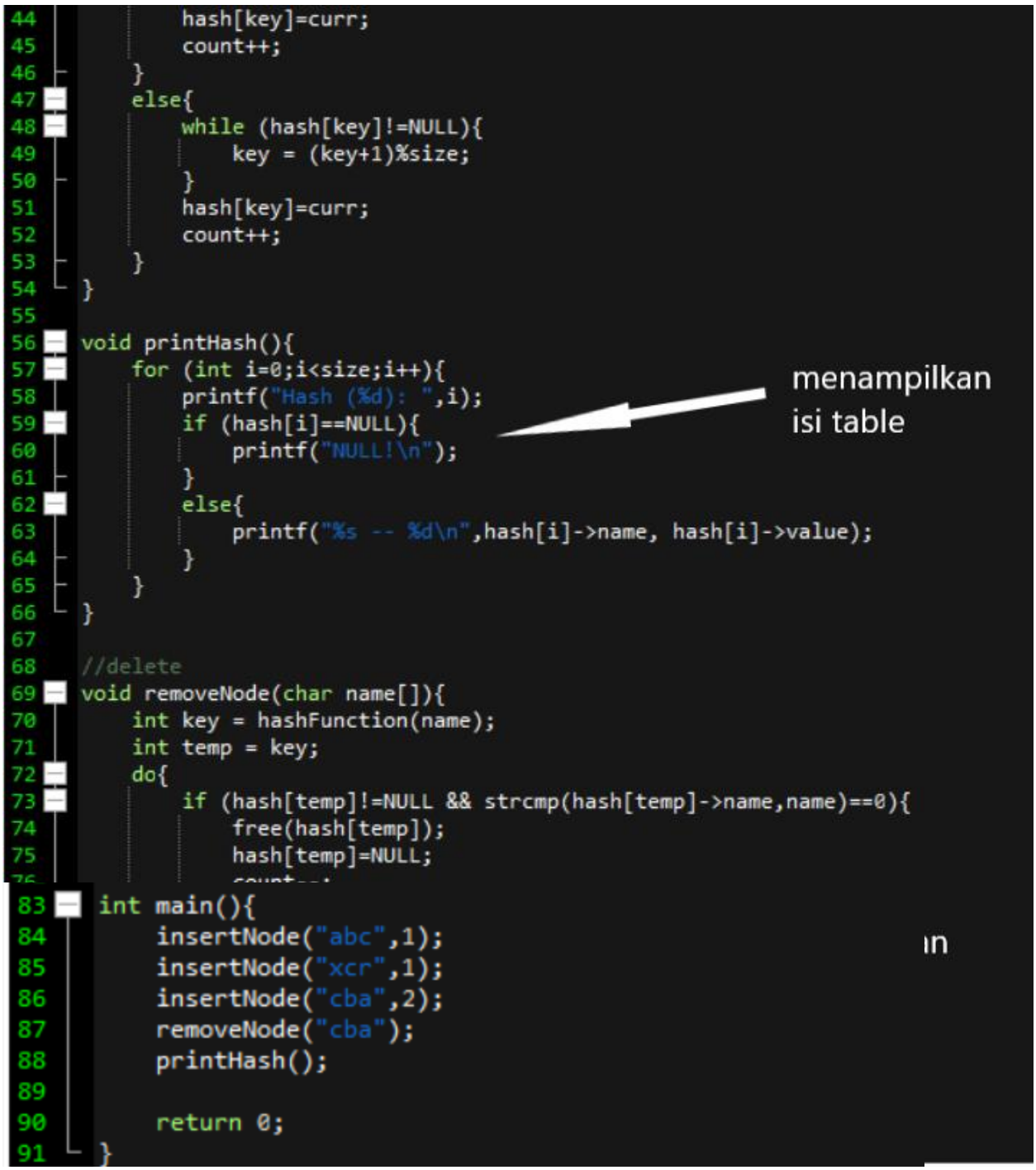
   Maka, *hash key* baru yang akan didapat adalah 604 (angka 406 cukup dibalik).

### 3.1.4. Collision

Collision adalah pertabrakan di antara 2 data/record atau lebih sehingga yang tersimpan dalam hash table hanya yang terakhir. Collision terjadi karena angka unik untuk index mengalami kesamaan antar data. Cara untuk menyelesaikan masalah ini ada 2, yaitu:

a. Linear Probing = memindahkan data yang memiliki index sama, ke index berikutnya yang kosong.

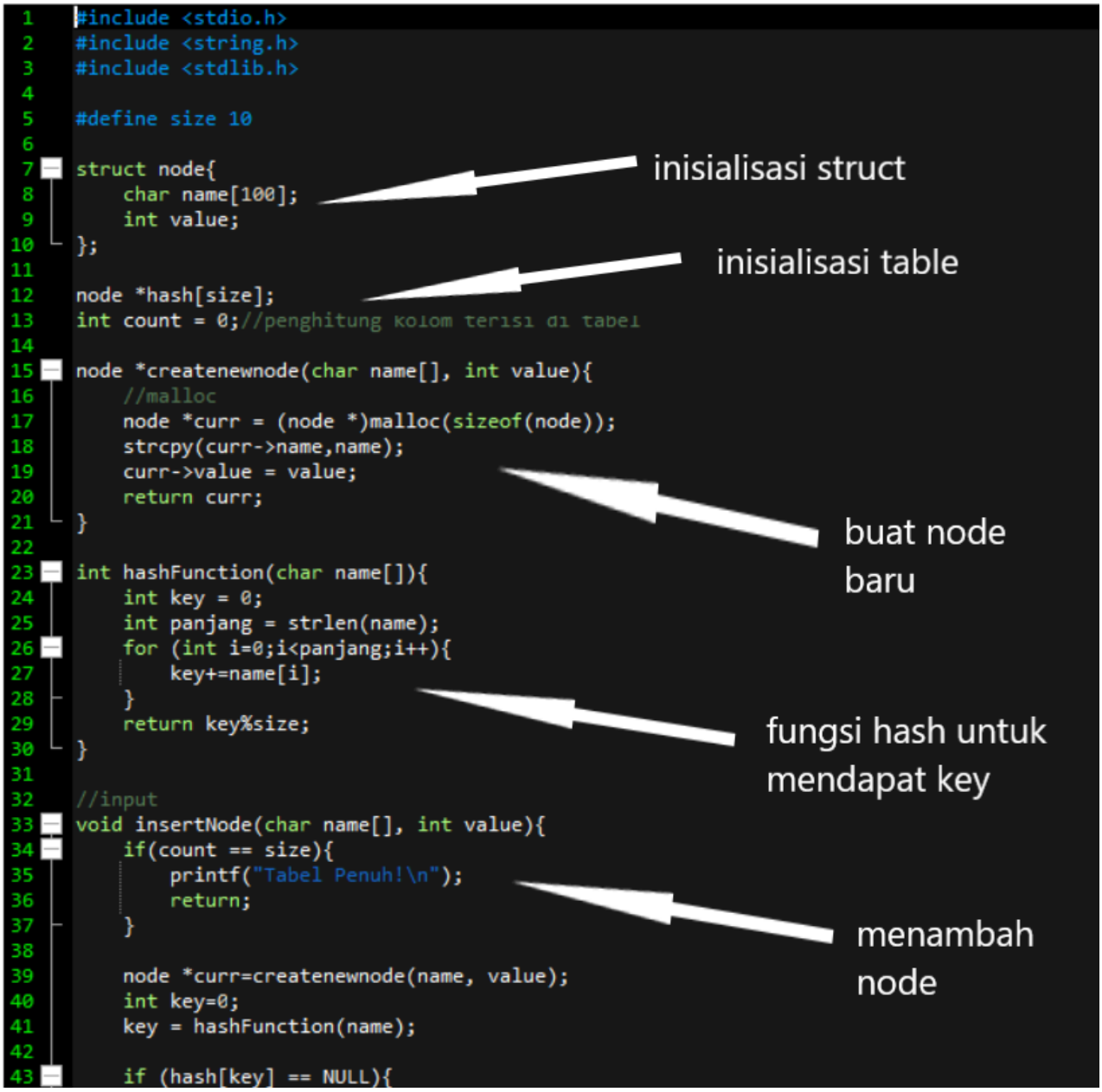
b. Chaining = membuat linked list dari data awal ke data baru.




```
        hash[key]=curr;
        count++;
    }
    else{
        while (hash[key]!=NULL){
            key = (key+1)%size;
        }
        hash[key]=curr;
        count++;
    }
}

void printHash(){
    for (int i=0;i<size;i++){
        printf("Hash (%d): ",i);
        if (hash[i]==NULL){
            printf("NULL!\n");
        }
        else{
            printf("%s -- %d\n",hash[i]->name, hash[i]->value);
        }
    }
}

//delete
void removeNode(char name[]){
    int key = hashFunction(name);
    int temp = key;
    do{
        if (hash[temp]!=NULL && strcmp(hash[temp]->name,name)==0){
            free(hash[temp]);
            hash[temp]=NULL;
```

```
int main(){
    insertNode("abc",1);
    insertNode("xcr",1);
    insertNode("cba",2);
    removeNode("cba");
    printHash();

    return 0;
}
```

Contoh pembuatan linear probing:




```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define size 10

struct node{
    char name[100];
    int value;
};

node *hash[size];
int count = 0;//penghitung kolom terisi di tabel

node *createnewnode(char name[], int value){
    //malloc
    node *curr = (node *)malloc(sizeof(node));
    strcpy(curr->name,name);
    curr->value = value;
    return curr;
}

int hashFunction(char name[]){
    int key = 0;
    int panjang = strlen(name);
    for (int i=0;i<panjang;i++){
        key+=name[i];
    }
    return key%size;
}

//input
void insertNode(char name[], int value){
    if(count == size){
        printf("Tabel Penuh!\n");
        return;
    }

    node *curr=createnewnode(name, value);
    int key=0;
    key = hashFunction(name);

    if (hash[key] == NULL){
```

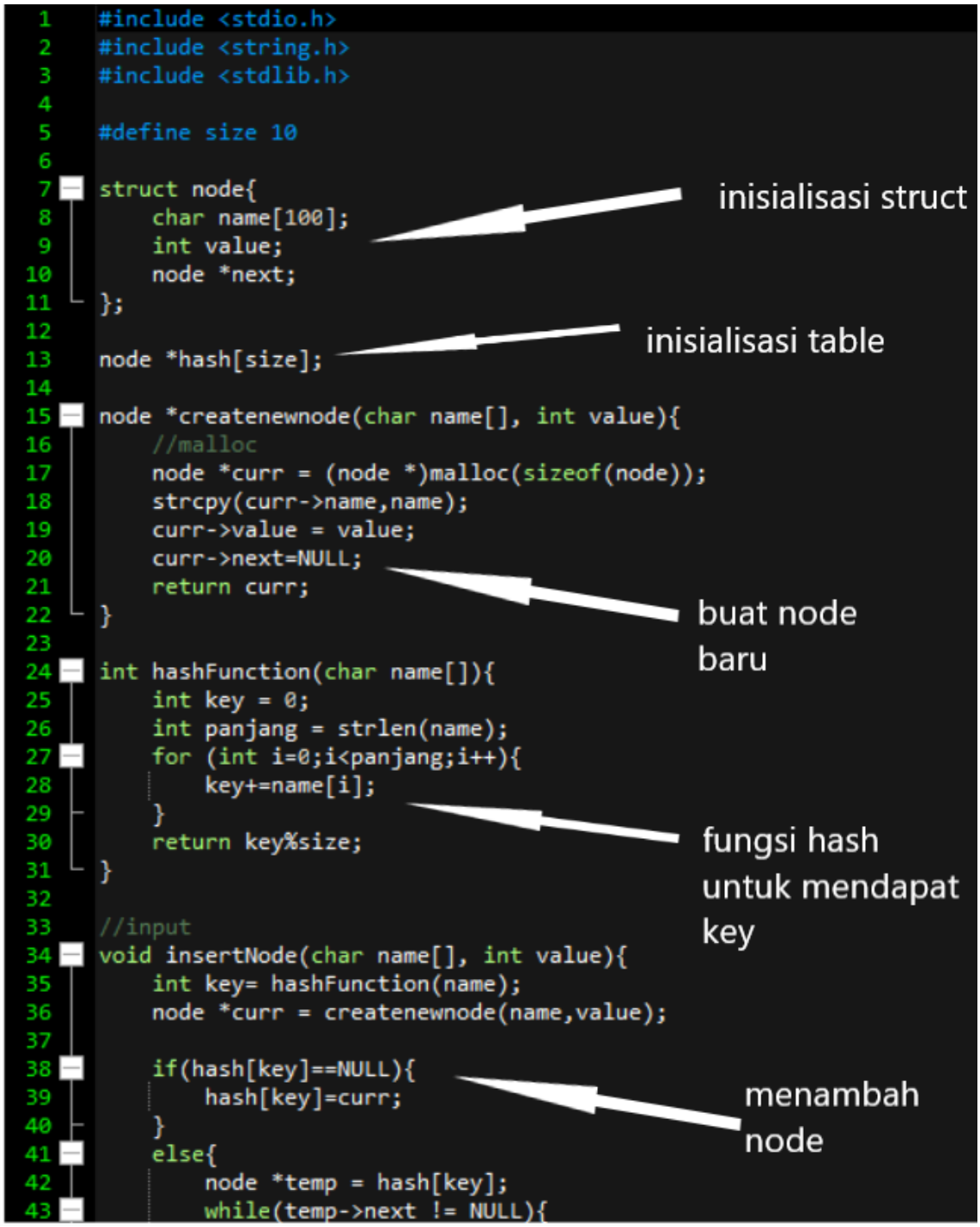
```
44          hash[key]=curr;
45          count++;
46      }
47      else{
48          while (hash[key]!=NULL){
49              key = (key+1)%size;
50          }
51          hash[key]=curr;
52          count++;
53      }
54  }
55
56  void printHash(){
57      for (int i=0;i<size;i++){
58          printf("Hash (%d): ",i);
59          if (hash[i]==NULL){
60              printf("NULL!\n");
61          }
62          else{
63              printf("%s -- %d\n",hash[i]->name, hash[i]->value);
64          }
65      }
66  }
67
68  //delete
69  void removeNode(char name[]){
70      int key = hashFunction(name);
71      int temp = key;
72      do{
73          if (hash[temp]!=NULL && strcmp(hash[temp]->name,name)==0){
74              free(hash[temp]);
75              hash[temp]=NULL;
76              count--;
77              return;
78          }
79          temp=(temp+1)%size;
80      }while(temp!=key);
81  }
82
83  int main(){
84      insertNode("abc",1);
85      insertNode("xcr",1);
86      insertNode("cba",2);
```

menampilkan isi table

menghilangkan node

```
83  int main(){
84      insertNode("abc",1);
85      insertNode("xcr",1);
86      insertNode("cba",2);
87      removeNode("cba");
88      printHash();
89
90      return 0;
91  }
```

Contoh dari pembuatan Chaining:




```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define size 10

struct node{
    char name[100];
    int value;
    node *next;
};

node *hash[size];

node *createnewnode(char name[], int value){
    //malloc
    node *curr = (node *)malloc(sizeof(node));
    strcpy(curr->name,name);
    curr->value = value;
    curr->next=NULL;
    return curr;
}

int hashFunction(char name[]){
    int key = 0;
    int panjang = strlen(name);
    for (int i=0;i<panjang;i++){
        key+=name[i];
    }
    return key%size;
}

//input
void insertNode(char name[], int value){
    int key= hashFunction(name);
    node *curr = createnewnode(name,value);

    if(hash[key]==NULL){
        hash[key]=curr;
    }
    else{
        node *temp = hash[key];
        while(temp->next != NULL){
```


```
44              temp=temp->next;
45          }
46          temp->next=curr;
47      }
48  }
49
50  void printHash(){
51      for (int i=0;i<size;i++){
52          printf("hash (%d): ",i);
53          node *temp=hash[i];
54          while(temp != NULL){
55              printf("%s -- %d -> ",temp->name,temp->value);
56              temp=temp->next;
57          }
58          printf("NULL!\n");
59      }
60  }
61
62  void removeNode(char name[]){
63      int key= hashFunction(name);
64      if (hash[key]==NULL){
65          return;
66      }
67      if(strcmp(hash[key]->name,name)==0){
68          node *temp1 = hash[key]->next;
69          free(hash[key]);
70          hash[key]=temp1;
71          return;
72      }
73      else{
74          node *temp = hash[key];
75          while(temp->name != NULL){
76              if(strcmp(temp->next->name,name)==0){
77                  node *temp2= temp->next;
78                  temp->next = temp2->next;
79                  free(temp2);
80                  return;
81              }
82              temp=temp->next;
83          }
84          //value not found
85      }
86  }
```

```
86  }
87
88  int main(){
89      insertNode("abc",1);
90      insertNode("xcr",1);
91      insertNode("tgf",1);
92      insertNode("sde",1);
93      insertNode("hhh",1);
94      insertNode("cba",2);
95  //  removeNode("cba");
96      removeNode("tgf");
97
98      printHash();
99
100     return 0;
101 }
```

## 4. REFERENSI


- https://www.freecodecamp.org/news/what-is-hashing/
- https://www.geeksforgeeks.org/mid-square-hashing/#:~:text=Mid%2DSquare%20hashing%20is%20a,taken%20as%20the%20new%20seed.
- https://www.tutorialspoint.com/Hash-Functions-and-Hash-Tables
- https://www.it-jurnal.com/pengertian-struktur-data/
- https://runestone.academy/runestone/books/published/pythonds/SortSearch/Hashing.html#:~:text=The%20folding%20method%20for%20constructing,give%20the%20resulting%20hash%20value.
- PPT Hashing and Hash Tables

Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY