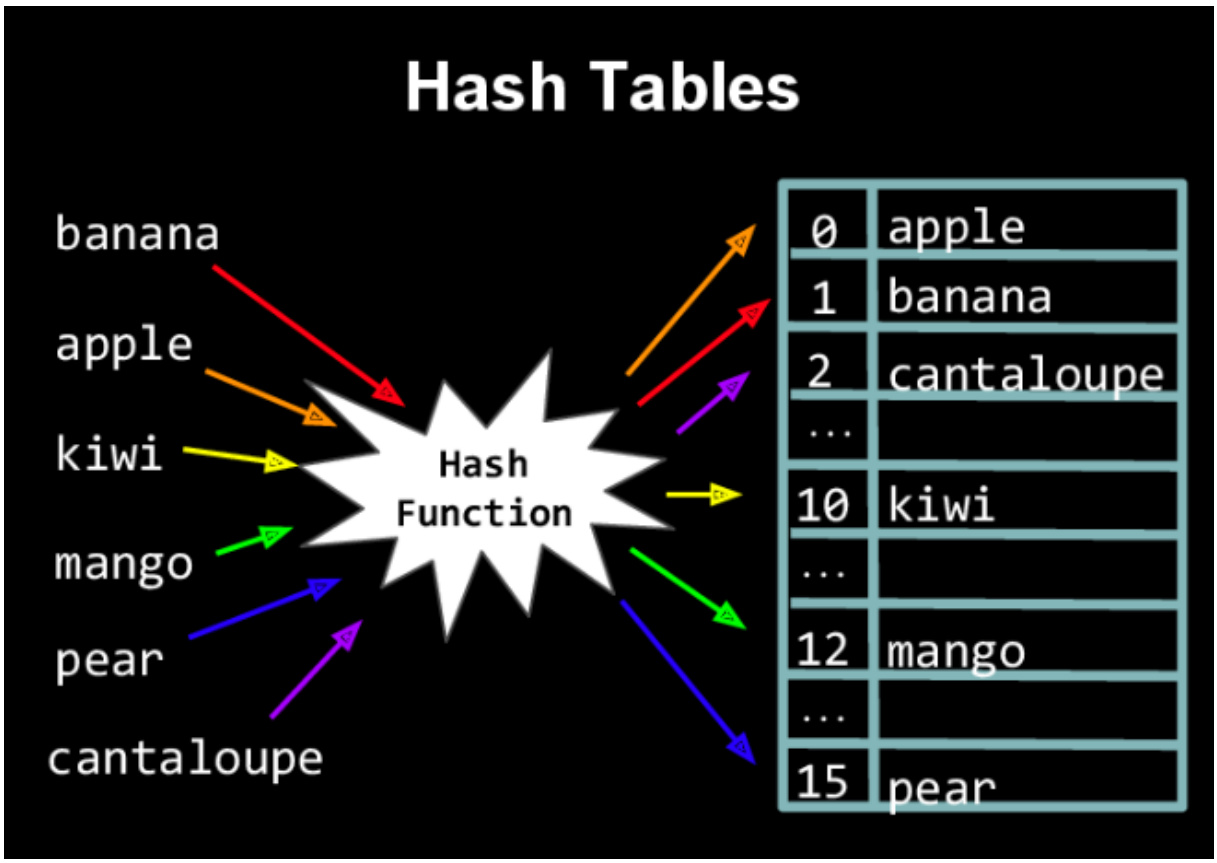
MINGGU 8
Hashing

DESKRIPSI TEMA

Hashing merupakan pengembangan dari direct access table. Idenya adalah untuk menggunakan Hash Function untuk mengonversi key data menjadi angka dan nomor yang lebih kecil dan menggunakan nomornya sebagai index pada table yang dinamakan Hash Table.

Hash Table : Array yang digunakan untuk menyimpan pointer ke sebuah record yang nantinya data akan dicari dengan Hash Function.
Hash Function : Function untuk melakukan konversi key data menjadi nomor / index.



Pada Hashing, terdapat sebuah masalah yang bernama Collision. Collision merupakan sebuah kondisi yang menyebabkan index hasil dari Hash Function menghasilkan value yang sama dengan data sebelumnya atau data yang sudah ada. Jika menggunakan gambar di atas, contohnya adalah dengan menambahkan data bernama Anggur. Jika Hash Function hanya mengonversi huruf pertama dan menjadikannya sebagai index, maka Anggur dan Apel akan bertabrakan. Terdapat 4 cara untuk mengatasi Collision, yaitu Linear Probing, Rehashing, Quadratic Probing, dan Separate Chaining. Masing-masing cara ini akan dijelaskan pada tutorial berikut.

CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

1. Mahasiswa mampu menerapkan Hashing menggunakan Bahasa C.
2. Mahasiswa mampu mengatasi Collision dengan menggunakan metode Linear Probing menggunakan Bahasa C.
3. Mahasiswa mampu mengatasi Collision dengan menggunakan metode Rehashing menggunakan Bahasa C.
4. Mahasiswa mampu mengatasi Collision dengan menggunakan metode Quadratic Probing menggunakan Bahasa C.
5. Mahasiswa mampu mengatasi Collision dengan menggunakan metode Separate Chaining menggunakan Bahasa C.

PENUNJANG PRAKTIKUM

1. Aplikasi CodeBlock
2. Aplikasi Dev-C++ (alternatif)

LANGKAH-LANGKAH PRAKTIKUM

A. Pendahuluan

Tutorial dasar Hashing terkait Hash Function dan Hash Table tidak ditunjukkan secara terpisah, melainkan bersamaan dengan Tutorial Hashing. Hal ini dikarenakan Hash Function bisa dalam berbagai macam dan Hash Table tidak hanya sebuah Array biasa, namun dapat juga Array of Pointer.

B. Hashing dengan Linear Probing.

Metode Linear Probing digunakan untuk menghindari Collision dengan menggunakan slot pada Hash Table yang masih tersedia setelah data di proses pada Hash Function.

- Tutorial 1.1 – Linear Probing
  a. Buat sebuah file dengan nama W08_HashingLinearProbing.c
  b. Salinlah potongan code di bawah ini.

```c
#include <stdio.h>
#include <stdbool.h>

void hashFunction(int input, int hashTable[]);

int main(){
  // #1 Create the Hash Table
  int HashT[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  int inputKey;

  while (1){
    printf("Input a number: ");
    scanf("%d", &inputKey);
    printf("Processing in Hash Fucntion\n");

    hashFunction(inputKey, HashT);

    //Print Hasil
    int i;
    printf("Current Hash Table: ");
    for (i = 0; i < 10; i++){
      printf("[%d]", HashT[i]);
    }
    printf("\n\n");
  }

  return 0;
}
```

```c
// #2 Create the Hash Function
void hashFunction(int input, int hashTable[]){
  int hash;
  bool check = false;
  hash = input % 10;
  int temp = hash;
  while (1){
    if (hashTable[temp] == 0){
      printf("Inserting data to Hash Table\n");
      hashTable[temp] = input;
      break;
    }
    temp++;
    if (hash == temp){
      printf("HashTable Full\n");
      break;
    }
    if (temp > 9){
      temp = 0;
    }
  }
}
```

C. Hashing dengan Rehashing

Rehashing adalah salah satu cara untuk menghindari Collision dengan membuat Hash Table baru yang memiliki ukuran (size) yang lebih besar.

- Tutorial 2.1 – Rehashing
  a. Buat sebuah file dengan nama W08_HashingRehashing.c
  b. Salinlah potongan code di bawah ini.
  c. Perhatikan nomor baris untuk mengikuti tutorial ini.

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
#include <windows.h>

void sieve(int prime[], int n, int *x){
  int i, j, p;
  int temp[n*2];

  for(i = 0;i < n*2; i++){
    temp[i] = 1;
  }

  for(p = 2; p * p <= n * 2; p++){
    if(temp[p] == 1){
      for(i = p * 2; i <= n; i += p){
        temp[i] = 0;
      }
    }
  }

  j = 0;
  for(i = 2; i < n;i++){
    if(temp[i] == 1){
      prime[j] = i;
      j++;
    }
  }
  *x = j;
}

int hashFunction(int number,int size){
  return number % size;
}
```

```c
int main(int argc, char const *argv[]){
  int *hashTable, *primeTable;
  int i,j, choose, index;
  int primeCount = 0;
  int capacity = 0, percentage = 0;
  int initSize = 10;
  int size = initSize;

  hashTable = (int*)malloc(sizeof(int)*initSize);
  primeTable = (int*)malloc(sizeof(int)*10000);

  for(i = 0; i < initSize;i++){
    hashTable[i] = -1;
  }

  for(i = 0; i < 10000;i++){
    primeTable[i] = 0;
  }

  sieve(primeTable, 10000, &primeCount);
  index = 0;
  do{
    system("cls");
    printf("1. Input number\n2. Show hash table\n3. Exit\n0. Choose: ");
    scanf("%d",&choose);
    if(choose == 1){
      system("cls");
      int number, hash;
      int done = 0;
      printf("Input one random number : "); scanf("%d", &number);
      capacity++;
      //using linear probing
      hash = hashFunction(number, size);
      while(!done){
        if(hashTable[hash] == -1){
          hashTable[hash] = number;
          done = 1;
        }
        else hash++;

        if(hash == size)hash = 0;
      }
      percentage = (capacity*100)/size;
      printf("Capacity of hash table = %d%%\n",percentage);

```

```
82      if(percentage >= 70){
83        printf("hashTable almost full. Making new hash table!\n");
84        //moving all data form hashTable to temporary array
85        int *arrTemp;
86        arrTemp = (int*)malloc(sizeof(int)*capacity);
87        j = 0;
88        for(i = 0;i < size; i++){
89          if(hashTable[i] != -1){
90            arrTemp[j] = hashTable[i];
91            j++;
92          }
93        }
94        //free hashTable
95        free(hashTable);
96
97        //new size is twice as many as before
98        //with the nearest prime number after the size
99        size *= 2;
100       //looking for prime
101       for(i = 0; i < primeCount; i++){
102         if(primeTable[i] > size){
103           size = primeTable[i];
104           break;
105         }
106       }
107
108       //create new hash table with new size
109       hashTable = (int*) malloc(sizeof(int)*size);
110       for(i = 0;i < size; i++){
111         hashTable[i] = -1;
112       }
113       //hash all data in arrTemp to hashTable
114       for(i = 0; i < capacity; i++){
115         hash = hashFunction(arrTemp[i], size);
116         done = 0;
117         while(!done){
118           if(hashTable[hash] == -1){
119             hashTable[hash] = arrTemp[i];
120             done = 1;
121           }
122           else hash++;
123
124           if(hash == size)hash = 0;
125         }
126       }
127       //free unused pointer
128       free(arrTemp);
129     }
```

```
130         else{
131           //still using hashTable
132           printf("hashTable still can be use\n");
133         }
134         getch();
135       }
136       else if(choose == 2){
137         system("cls");
138         printf("Content of hash table:\n");
139         for(i = 0;i <size;i++){
140           printf("hashTable[%d] = %d\n",i , hashTable[i]);
141         }
142         getch();
143       }
144       else if(choose == 0){
145         free(hashTable);
146         break;
147       }
148       else continue;
149     }while(1);
150 
151     return 0;
152   }
```

d. Jawablah pertanyaan di bawah ini dan simpan jawaban dengan nama NIM_Rehashing.txt
   a) Mengapa dilakukan rehashing?
   b) Kapan rehashing dilakukan berdasarkan gambar di atas?
   c) Sebutkan langkah-langkah melakukan rehashing secara singkat, padat, dan jelas!
   d) Jelaskan secara singkat apa itu Sieve of Erathosthenes dalam 1 kalimat!
   e) Apakah size yang baru harus mengikuti bilangan prima?

D. Hashing dengan Quadratic Probing

Quadratic Probing adalah salah satu metode untuk mengatasi Collision dengan menggunakan sebuah fungsi kuadratik untuk menentukan letak sebuah data yang akan ditaruh dalam Hash Table.

- Tutorial 3.1 – Quadratic Probing
  a. Buat sebuah file dengan nama W08_QuadraticProbing.c
  b. Salinlah potongan code di bawah ini.
  c. Perhatikan nomor baris untuk mengikuti tutorial ini.

```c
#include <stdio.h>
#include <conio.h>
#include <windows.h>

int quadraticFunction(int number, int n){
  int i, temp;
  temp = n;
  for(i = 0; i < n-1 ; i++){
    temp *= n;
  }
  return (number + temp) % 10;
}

int hashFunction(int number){
  return number % 10;
}
```

```c
int main(){
  int arr[] = {3, 5, 13, 24, 33, 45, 54, 78, 81, 99, 100};
  int hashTable[10];
  int i, n;
  int hashResult, quadraticResult;

  for(i = 0; i < 10; i++){
    hashTable[i] = -1;
  }

  for(i = 0; i < sizeof(arr)/4; i++){
    hashResult = hashFunction(arr[i]);
    n = 0;
    do{
      quadraticResult = quadraticFunction(hashResult, n);
      if(hashTable[quadraticResult] == -1){
        hashTable[quadraticResult] = arr[i];
        break;
      }
      else if(n == sizeof(arr)/4) {
        break;
      }
      else n++;
    }while(1);
  }

  for(i = 0;i < 10 ; i++){
    printf("hashTable[%d] = %d\n",i, hashTable[i]);
  }

  return 0;
}
```

E. Hashing dengan Seperate Chaining

Metode ini menggunakan Array of Linked List dalam menanggulangu Collision. Jika data sudah ada pada suatu Array, maka pada indeks tersebut akan dibuatkan chain.

- Tutorial 4.1 – Seperate Chaining
  a. Buat sebuah file dengan nama W08_SeperateChaining.c
  b. Salinlah potongan code di bawah ini.
  c. Perhatikan nomor baris untuk mengikuti tutorial ini.

```c
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <windows.h>

typedef struct numList{
    int number;
    struct numList *next;
}numList;

void insertToChain(int key, numList **head){
    numList *ptr = (*head);
    numList *node = (numList*)malloc(sizeof( numList));
    node->number = key;
    node->next = NULL;
    if(*head == NULL){
      *head = node;
    }
    else{
      while(ptr->next != NULL){
        ptr = ptr->next;
      }
      ptr->next = node;
    }
}
```

```c
int main(){
  numList *HashT[10];
  int i;
  for(i = 0; i < 10; i++){
    HashT[i] = NULL;
  }
  while(1){
    system("cls");
    for(i = 0; i < 10;i++){
      numList *ptr = HashT[i];
      printf("[%d]->",i);
      while(ptr!= NULL){
        printf("(%d)",ptr->number);
        ptr = ptr->next;
      }
      printf("\n");
    }
    int inputKey;
    scanf("%d",&inputKey);
    insertToChain(inputKey,&HashT[inputKey%10]);
  }

  return 0;
}
```

F. Tugas

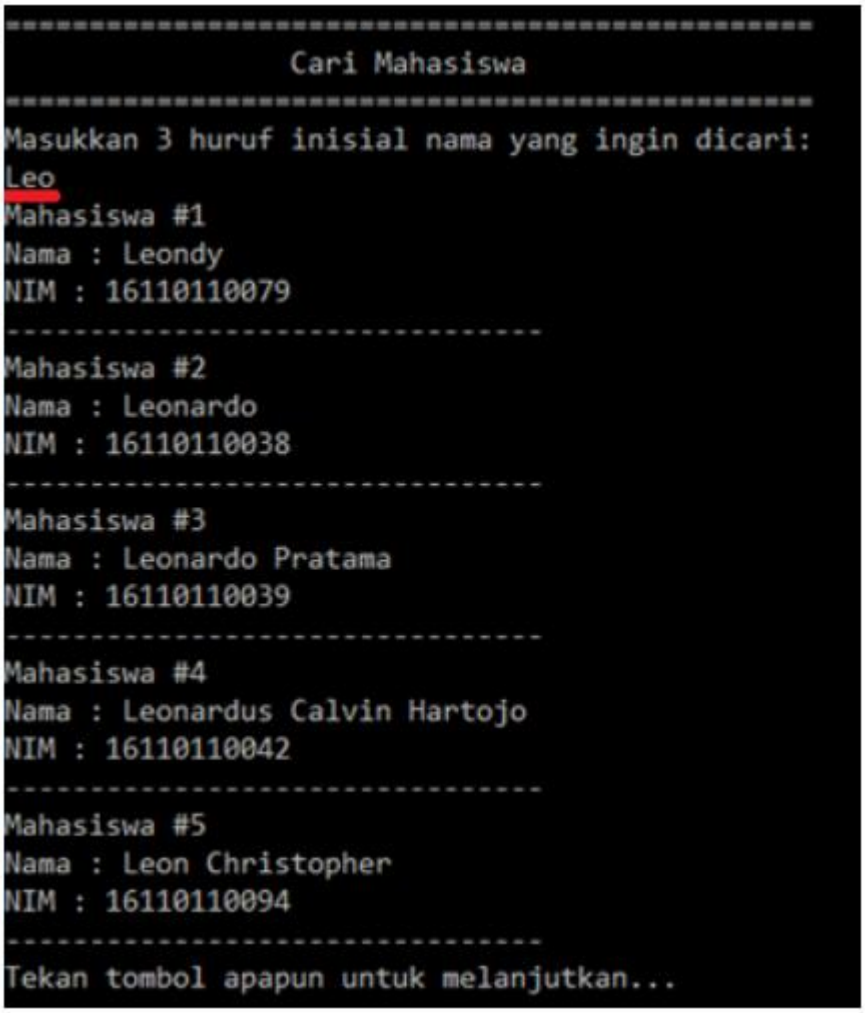
Buatlah program yang dapat menyimpan daftar nama, di mana tiap nama disimpan dalam Hash Table. Entry Hash Table menggunakan Linked List untuk mencegah collision. Hash function yang digunakan akan menggunakan 3 huruf pertama dari nama untuk menentukan lokasi di Hash Table (a=0,b=1 dst). Terdapat juga file pendukung dengan nama **datamhs.txt**. Simpan dengan nama **W08_NIM.c**.

Petunjuk : Karena menggunakan 3 huruf pertama dari nama, maka diperlukan 3D Array of Struct!

- Main Menu

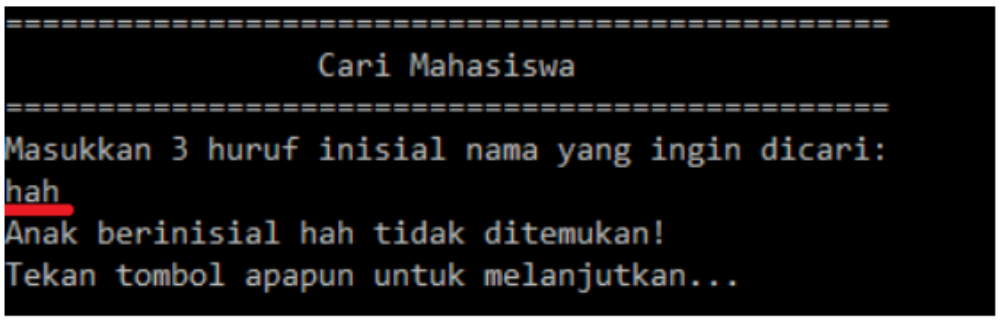
```
================================================
                Daftar Mahasiswa
================================================
(1). Cari Mahasiswa (berdasar 3 huruf pertama)
(2). Delete Mahasiswa
(3). Tambah Mahasiswa
(0). Exit
Pilih :
```

- Menu 1



Input harus 3 karakter dan yang ditampilkan adalah data – data nama mahasiswa yang memiliki huruf 3 pertama yang sama dengan input.

Bila input yang dimasukkan tidak merujuk ke data apa pun, tampilannya adalah sebagai berikut.



- Menu

```
================================================
                Hapus Mahasiswa
================================================
Masukkan nama mahasiswa : Justin Susanto
Mahasiswa dengan nama Justin Susanto berhasil dihapus!
Tekan tombol apapun untuk melanjutkan...
```

Sehingga, jika kembali menu 1 dan mencari inisial nama dari data yang telah dihapus maka akan hilang. Jangan lupa untuk mengatur dimana kondisi jika data yang ingin dihapus tidak ditemukan dan kembali ke menu utama.

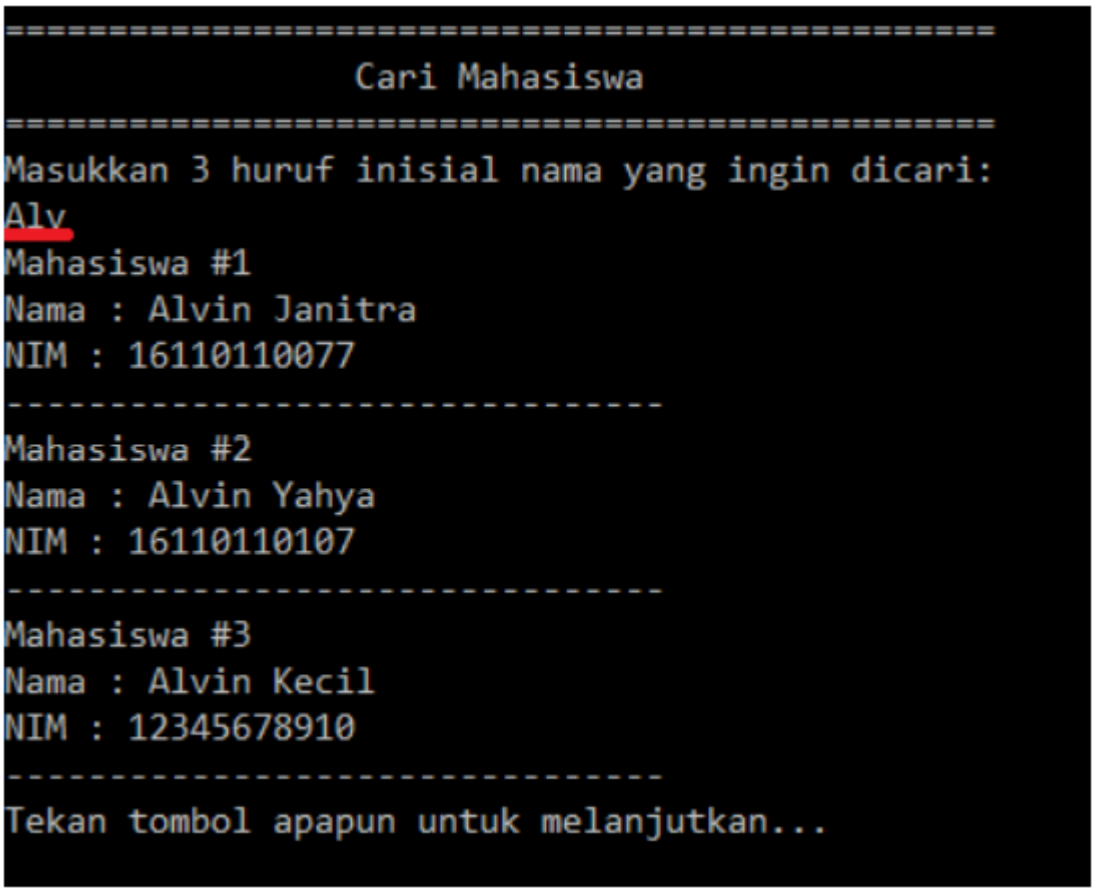
```
================================================
                Cari Mahasiswa
================================================
Masukkan 3 huruf inisial nama yang ingin dicari:
Jus
Anak berinisial Jus tidak ditemukan!
Tekan tombol apapun untuk melanjutkan...
```

Input nama yang dimasukkan selain menghapus data dalam Linked List juga meng-update data yang ada dalam file txt!

- Menu 3

```
================================================
                Tambah Mahasiswa
================================================
Masukkan nama mahasiswa baru : Alvin Kecil
Masukkan NIM mahasiswa baru : 12345678910
Mahasiswa baru telah dimasukkan!
Tekan tombol apapun untuk melanjutkan...
```

Mirip seperti menu sebelumnya, jika kembali ke menu 1 dan melakukan pencarian data sesuai inisial nama yang telah ditambahkan maka data yang baru dimasukkan akan muncul.



Jangan lupa untuk mengupdate isi didalam txt agar data yang baru ditambahkan masuk.

REFERENSI