# LAPORAN PRAKTIKUM 13
## Algoritma Pencarian (Searching)
Semester 2 Tahun Akademik 2022/2023


**DOSEN PENGAMPU:**
**Entin Martiana Kusumaningtyas, S.kom, M.kom**
**PENYUSUN:**
**Arsyita Devanaya Arianto (3122500008)**



**PROGRAM STUDI VOKASI**
**D-III TEKNIK INFORMATIKA**
**DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER**
**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

# Praktikum 13

# Algoritma Pencarian (Searching)

## PERCOBAAN

**Percobaan 1 : Implementasi pencarian dengan metode sequential search**

Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 10

int Data[MAX];

int SequintialSearch(int x){
    int i=0;
    bool ketemu=false;

    while ((!ketemu)&&(i<MAX)){
        if(Data[i]==x){
            ketemu=true;
        }else{
            i++;
        }
    }
```

```
    if(ketemu){
        return i;
    }else{
        return -1;
    }
}

void main(){
    int i;

    srand(0);

    printf("\nData : ");
    for(i=0; i<MAX; i++){
        Data[i]=rand()/1000+1;
        printf("%d ", Data[i]);
    }
    int Kunci;

    printf("\nKunci : ");
    scanf("%d", &Kunci);

    int ketemu=SequintialSearch(Kunci);
    if(ketemu>0){
        printf("Data ditemukan pada posisi %d", ketemu);
    }else{
        printf("Data tidak ditemukan");
    }
}
```

Output:

```
Data : 1 8 22 3 9 12 9 33 11 31
Kunci : 3
Data ditemukan pada posisi 3
```

Analisa:

Program ini adalah implementasi bahasa C sederhana dari algoritma pencarian berurutan. Program ini memiliki kumpulan data berukuran 10 yang diisi dengan nilai acak menggunakan fungsi rand(). Program kemudian meminta pengguna untuk memasukkan nilai kunci yang akan diambil dari tabel data. Kemudian program melakukan pencarian di tabel nilai kunci menggunakan fungsi SequentialSearch(). Jika nilai kunci ditemukan di dalam larik, program akan mencetak pesan "Data ditemukan di posisi {indeks}". Jika nilai kunci tidak ditemukan dalam tabel, program mengeluarkan pesan "Data tidak ditemukan". Algoritma pencarian sekuensial atau pencarian linear digunakan sebagai algoritma pencarian, mengulang setiap elemen array hingga nilai yang diinginkan ditemukan atau hingga semua elemen diperiksa.

**Percobaan 2 : Implementasi pencarian dengan metode binary seach**

Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 10

int Data[MAX];
void Tukar(int *a, int *b){
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;

}

void QuickSort(int L, int R){
    int i, j, x;
    x=Data[(L+R)/2];
    i=L;
    j=R;
    while(i<=j){
        while(Data[i]<x){
            i++;
        }
        while(Data[j]>x){
```

```
            j--;
        }
        if(i<=j){

            Tukar(&Data[i], &Data[j]);
            i++;
            j--;
        }
    }
    if(L<j){
        QuickSort(L,j);
    }
    if(i<R){
        QuickSort(i, R);
    }
}
int BinarySearch(int x){
    int L = 0, R = MAX-1, m;
    bool ketemu = false;
    while((L <= R) && (!ketemu)){
        m = (L + R) / 2;
        if(Data[m] == x){
            ketemu = true;
        }else if (x < Data[m]){
            R = m - 1;
        }else{
            L = m + 1;
        }

    }
    if(ketemu){
        return m;
    }else{
        return -1;
```

```
    }

}
void main(){
    int i;

    srand(0);

    printf("\nData : ");
    for(i=0; i<MAX; i++){
        Data[i]=rand()/1000+1;
        printf("%d ", Data[i]);
    }
    QuickSort(0, MAX-1);
    int Kunci;

    printf("\nKunci : ");
    scanf("%d", &Kunci);

    int ketemu=BinarySearch(Kunci);
    if(ketemu>0){
        printf("Data ditemukan pada posisi %d", ketemu);
    }else{
        printf("Data tidak ditemukan");
    }
}
```

Output:

```
Data : 1 8 22 3 9 12 9 33 11 31
Kunci : 9
Data ditemukan pada posisi 4
```

Analisa:

Program ini adalah implementasi bahasa C sederhana dari algoritma pencarian berurutan. Program ini memiliki kumpulan data berukuran 10 yang diisi dengan nilai acak menggunakan fungsi rand(). Program kemudian meminta pengguna untuk memasukkan nilai kunci yang

akan diambil dari tabel data. Kemudian program melakukan pencarian di tabel nilai kunci menggunakan fungsi SequentialSearch(). Jika nilai kunci ditemukan di dalam larik, program akan mencetak pesan "Data ditemukan di posisi {indeks}". Jika nilai kunci tidak ditemukan dalam tabel, program mengeluarkan pesan "Data tidak ditemukan". Algoritma pencarian sekuensial atau pencarian linear digunakan sebagai algoritma pencarian, mengulang setiap elemen array hingga nilai yang diinginkan ditemukan atau hingga semua elemen diperiksa.

# LATIHAN

1. Tambahkan kode program untuk menghitung jumlah perbandingan yang dilakukan dengan metode sequential search dan binary search.
   Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 10

int Data[MAX];

int SequintialSearch(int x ){
    int i=0, hitung=0;
    bool ketemu=false;

    while ((!ketemu)&&(i<MAX)){
        hitung++;
        if(Data[i]==x){
            ketemu=true;
        }else{
            i++;
        }
    }
    if(ketemu){
        return hitung;
    }else{
        return -1;
    }
}
int Data[MAX];
void Tukar(int *a, int *b){
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;

}

void QuickSort(int L, int R){
    int i, j, x;
    x=Data[(L+R)/2];
    i=L;
    j=R;
    while(i<=j){
        while(Data[i]<x){
```

```
            i++;
        }
        while(Data[j]>x){
            j--;
        }
        if(i<=j){

            Tukar(&Data[i], &Data[j]);
            i++;
            j--;
        }
    }
    if(L<j){
        QuickSort(L,j);
    }
    if(i<R){
        QuickSort(i, R);
    }
}
int BinarySearch(int x){
    int L = 0, R = MAX-1, m, hitung=0;
    bool ketemu = false;
    while((L <= R) && (!ketemu)){
        hitung++;
        m = (L + R) / 2;
        if(Data[m] == x){
            ketemu = true;
        }else if (x < Data[m]){
            R = m - 1;
        }else{
            L = m + 1;
        }

    }
    if(ketemu){
        return hitung;
    }else{
        return -1;
    }

}
void main(){
    int i;

    srand(0);

    printf("\nData : ");
    for(i=0; i<MAX; i++){
        Data[i]=rand()/1000+1;
        printf("%d ", Data[i]);
    }
    int Kunci;

    printf("\nKunci : ");
    scanf("%d", &Kunci);
```

```
    int ketemu=SequintialSearch(Kunci);
    if(ketemu>0){
        printf("Data %d ditemukan pada perulangan (Sequintial) ke-
%d\n",Kunci, ketemu);
    }else{
        printf("Data tidak ditemukan");
    }
    QuickSort(0, MAX-1);
    int ketemuBinary=BinarySearch(Kunci);
    if(ketemu>0){
        printf("Data %d ditemukan pada posisi perulangan (Binary) ke-
%d\n", Kunci,ketemuBinary);
    }else{
        printf("Data tidak ditemukan");
    }

}
```

Output:

```
Data : 1 8 22 3 9 12 9 33 11 31
Kunci : 9
Data 9 ditemukan pada perulangan (Sequintial) ke-5
Data 9 ditemukan pada posisi perulangan (Binary) ke-1
```

Analisa:

Program di atas adalah program implementasi untuk pencarian sekuensial dan pencarian biner dalam bahasa pemrograman C. Program ini pertama-tama menginisialisasi array data berukuran 10 dengan nilai acak menggunakan fungsi rand(). Kemudian program meminta pengguna untuk memasukkan nilai kunci yang dapat dicari di tabel data. Selanjutnya, program melakukan pencarian berurutan dalam tabel data menggunakan fungsi SequentialSearch(). Jika nilai kunci ditemukan dalam larik, program mengembalikan pesan "Data ditemukan dalam iterasi %d (Sequintial)", di mana %d adalah jumlah iterasi yang diperlukan untuk menemukan nilai. Jika nilai kunci tidak ditemukan dalam tabel, program mengeluarkan pesan "Data tidak ditemukan".

Setelah pencarian berurutan selesai, program mengurutkan data dalam larik data menggunakan fungsi QuickSort(). Setelah menyortir data, program melakukan pencarian biner di tabel data menggunakan fungsi BinarySearch(). Jika nilai kunci ditemukan dalam larik, program akan mencetak pesan "Data ditemukan dalam iterasi %d (Binary)", di mana %d adalah jumlah iterasi yang diperlukan untuk menemukan nilai. Jika nilai kunci tidak ditemukan dalam tabel, program mengeluarkan pesan "Data tidak ditemukan".

2. Bandingkan kinerja pencarian dengan sequential search dan binary search berdasarkan latihan point 1.

Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <stdbool.h>
#define MAX 10
```

```
int Data[MAX];

int SequintialSearch(int x ){
    int i=0, hitung=0;
    bool ketemu=false;

    while ((!ketemu)&&(i<MAX)){
        hitung++;
        if(Data[i]==x){
            ketemu=true;
        }else{
            i++;
        }
    }
    if(ketemu){
        return hitung;
    }else{
        return -1;
    }
}
int Data[MAX];
void Tukar(int *a, int *b){
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;

}

void QuickSort(int L, int R){
    int i, j, x;
    x=Data[(L+R)/2];
    i=L;
    j=R;
    while(i<=j){
        while(Data[i]<x){
            i++;
        }
        while(Data[j]>x){
            j--;
        }
        if(i<=j){

            Tukar(&Data[i], &Data[j]);
            i++;
            j--;
        }
    }
    if(L<j){
        QuickSort(L,j);
    }
    if(i<R){
        QuickSort(i, R);
    }
}
```

```
int BinarySearch(int x){
    int L = 0, R = MAX-1, m, hitung=0;
    bool ketemu = false;
    while((L <= R) && (!ketemu)){
        hitung++;
        m = (L + R) / 2;
        if(Data[m] == x){
            ketemu = true;
        }else if (x < Data[m]){
            R = m - 1;
        }else{
            L = m + 1;
        }

    }
    if(ketemu){
        return hitung;
    }else{
        return -1;
    }

}
void main(){
    int i;
    struct timeval begin, end, begin2, end2;
    printf("\nSebelum pengurutan : \n");
    for(i=0; i<MAX; i++){
        srand(time(NULL) * (i+1));
        Data[i] = rand() % 100 + 1;
        printf("%d ", Data[i]);
    }

    int Kunci;

    printf("\nKunci : ");
    scanf("%d", &Kunci);
    gettimeofday(&begin, 0);
    int ketemu=SequintialSearch(Kunci);
    gettimeofday(&end, 0);
    long seconds = end.tv_sec - begin.tv_sec;
    long microseconds = end.tv_usec - begin.tv_usec;
    double elapsed = seconds + microseconds*1e-6;

    if(ketemu>0){
        printf("Data %d ditemukan pada perulangan ke-%d\n",Kunci,
ketemu);
    }else{
        printf("Data tidak ditemukan");
    }
    printf("\nTime measured: %.3f seconds.\n", elapsed);
    gettimeofday(&begin2, 0);
    QuickSort(0, MAX-1);
    int ketemuBinary=BinarySearch(Kunci);
    gettimeofday(&end2, 0);
```

```
    long seconds2= end2.tv_sec - begin2.tv_sec;
    long microseconds2 = end2.tv_usec - begin2.tv_usec;
    double elapsed2 = seconds2 + microseconds2*1e-6;
    if(ketemu>0){
        printf("Data %d ditemukan pada posisi perulangan ke-%d\n",
Kunci,ketemuBinary);
    }else{
        printf("Data tidak ditemukan");
    }
    printf("\nTime measured: %.3f seconds.\n", elapsed2);

}
```

Output:

```
Sebelum pengurutan :
75 12 48 85 21 89 26 62 98 35
Kunci : 26
Data 26 ditemukan pada perulangan ke-7

Time measured: 0.000 seconds.
Data 26 ditemukan pada posisi perulangan ke-3

Time measured: 0.000 seconds.
```

Analisa:

Program ini merupakan implementasi dari dua metode pencarian data, yaitu Sequential Search dan Binary Search, serta pengurutan data dengan metode QuickSort. Program ini memiliki ukuran data sebesar MAX yang telah didefinisikan sebelumnya dengan nilai 10.

Pertama-tama, program akan menghasilkan bilangan acak sebanyak MAX dan menampilkannya di layar. Setelah itu, program akan meminta pengguna memasukkan kunci yang akan dicari pada array Data menggunakan Sequential Search. Sequential Search dilakukan dengan mengiterasi seluruh elemen pada array Data, jika ditemukan kunci pada suatu elemen, maka iterasi akan dihentikan dan program akan menampilkan posisi kunci ditemukan pada perulangan ke-berapa. Jika kunci tidak ditemukan, maka program akan menampilkan pesan "Data tidak ditemukan".

Setelah Sequential Search selesai, program akan melakukan pengurutan array Data menggunakan QuickSort. QuickSort dilakukan dengan memilih elemen pivot pada nilai tengah array Data, kemudian mempartisi array menjadi dua bagian, yaitu elemen yang lebih kecil dan elemen yang lebih besar dari pivot. Setiap bagian kemudian akan diurutkan secara rekursif menggunakan algoritma QuickSort hingga elemen terurut secara keseluruhan.

Setelah array Data terurut, program akan melakukan Binary Search untuk mencari kunci yang diminta pengguna. Binary Search dilakukan dengan membagi array Data menjadi dua bagian pada setiap iterasi, kemudian memeriksa kunci apakah berada pada bagian kiri atau kanan dari pivot. Iterasi dilakukan terus menerus hingga kunci ditemukan atau tidak ditemukan pada array Data. Jika kunci ditemukan, program akan menampilkan posisi kunci ditemukan pada perulangan ke-berapa. Jika kunci tidak ditemukan, maka program akan menampilkan pesan "Data tidak ditemukan".

Program ini juga mengukur waktu yang dibutuhkan oleh Sequential Search dan Binary Search menggunakan struct timeval untuk memperoleh waktu sebelum dan setelah

pencarian. Waktu yang diukur dihitung dalam satuan detik dan mikrodetik, kemudian diubah menjadi satuan detik. Hasil pengukuran waktu akan ditampilkan di layar.

3. Implementasikan method sequential search dengan cara rekursif.

Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 10

int SequentialSearchRecursive(int arr[], int n, int x, int i) {
    if (i == n) {
        return -1;  // elemen tidak ditemukan
    }
    if (arr[i] == x) {
        return i;   // elemen ditemukan pada indeks i
    }
    return SequentialSearchRecursive(arr, n, x, i+1);  // lanjut ke
elemen berikutnya
}

int main(){
    int i;
    int Data[MAX];
    srand(time(0));

    printf("\nData : ");
    for(i=0; i<MAX; i++){
        Data[i]=rand()/1000+1;
        printf("%d ", Data[i]);
    }
    int Kunci;

    printf("\nKunci : ");
    scanf("%d", &Kunci);

    int ketemu=SequentialSearchRecursive(Data, MAX, Kunci,0);
    if(ketemu>=0){
        printf("Data ditemukan pada posisi %d", ketemu);
    }else{
        printf("Data tidak ditemukan");
    }
    return 0;
}
```

Output:

```
Data : 7 27 7 19 29 30 7 8 31 26
Kunci : 29
Data ditemukan pada posisi 4
```

Analisa:

Program di atas merupakan implementasi algoritma Sequential Search menggunakan rekursi. Algoritma ini mencari elemen pada suatu array dengan membandingkan setiap elemen dengan kunci yang dicari, secara berurutan dari awal hingga akhir array.

Pada program ini, terdapat fungsi SequentialSearchRecursive yang memiliki empat parameter, yaitu arr (array yang akan dicari), n (jumlah elemen dalam array), x (kunci yang dicari), dan i (indeks elemen yang akan dicek). Fungsi ini akan melakukan pencarian secara rekursif dengan memeriksa apakah elemen yang dicari (x) sama dengan elemen pada indeks i. Jika sama, fungsi akan mengembalikan indeks tersebut. Jika tidak, fungsi akan memanggil dirinya sendiri untuk memeriksa elemen pada indeks berikutnya. Jika indeks i sudah mencapai n, artinya seluruh elemen pada array sudah diperiksa namun tidak ada yang sama dengan kunci, maka fungsi mengembalikan -1.

Di dalam fungsi main, terdapat proses pengisian array Data dengan bilangan acak, kemudian input kunci yang akan dicari. Setelah itu, fungsi SequentialSearchRecursive dipanggil dengan menggunakan array Data, jumlah elemen dalam array, kunci yang dicari, dan indeks awal (0) sebagai parameter. Hasil dari pencarian kemudian dicetak pada layar.

Program ini memiliki kelebihan yaitu implementasi yang sederhana dan mudah dimengerti, sehingga cocok digunakan untuk kasus sederhana. Namun, kelemahannya terletak pada waktu eksekusi yang lebih lama dibandingkan algoritma pencarian lain seperti Binary Search atau Hashing. Pada kasus-kasus dengan jumlah data yang besar, waktu eksekusi Sequential Search secara rekursif dapat sangat lambat dan tidak efisien.

4. Implementasikan method binary search dengan cara rekursif

Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 10

int Data[MAX];
void Tukar(int *a, int *b){
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;

}

void QuickSort(int L, int R){
    int i, j, x;
    x=Data[(L+R)/2];
    i=L;
    j=R;
    while(i<=j){
        while(Data[i]<x){
            i++;
        }
        while(Data[j]>x){
            j--;
        }
        if(i<=j){

            Tukar(&Data[i], &Data[j]);
            i++;
            j--;
        }
```

```
    }
    if(L<j){
        QuickSort(L,j);
    }
    if(i<R){
        QuickSort(i, R);
    }
}
int BinarySearch(int x, int L, int R) {
    if (L > R) {
        return -1; // elemen tidak ditemukan
    }
    int m = (L + R) / 2;
    if (Data[m] == x) {
        return m; // elemen ditemukan pada indeks m
    } else if (x < Data[m]) {
        return BinarySearch(x, L, m-1); // cari di setengah kiri
    } else {
        return BinarySearch(x, m+1, R); // cari di setengah kanan
    }
}

void main(){
    int i;

    srand(0);

    printf("\nData : ");
    for(i=0; i<MAX; i++){
        Data[i]=rand()/1000+1;
        printf("%d ", Data[i]);
    }
    QuickSort(0, MAX-1);
    int Kunci;

    printf("\nKunci : ");
    scanf("%d", &Kunci);

    int ketemu=BinarySearch(Kunci,0,MAX-1);
    if(ketemu>0){
        printf("Data ditemukan pada posisi %d", ketemu);
    }else{
        printf("Data tidak ditemukan");
    }
}
```

Output:

```
Data : 1 8 22 3 9 12 9 33 11 31
Kunci : 33
Data ditemukan pada posisi 9
```

Analisa:

Program ini merupakan contoh implementasi algoritma QuickSort dan Binary Search dalam bahasa pemrograman C. Algoritma QuickSort digunakan untuk mengurutkan array

Data secara ascending (dari kecil ke besar) sebelum dilakukan pencarian dengan algoritma Binary Search.

Fungsi QuickSort menerima parameter L dan R, yang merupakan indeks awal dan akhir array Data yang akan diurutkan. Pada fungsi ini, elemen yang dipilih sebagai pivot adalah elemen di tengah array (Data[(L+R)/2]). Kemudian, array dibagi menjadi dua bagian, yaitu elemen-elemen yang lebih kecil daripada pivot dan elemen-elemen yang lebih besar daripada pivot. Setiap bagian akan diurutkan secara rekursif dengan menggunakan QuickSort hingga array menjadi terurut.

Setelah array Data terurut, program meminta input dari pengguna berupa nilai Kunci yang akan dicari menggunakan algoritma Binary Search. Fungsi BinarySearch menerima parameter x, L, dan R, yang merupakan nilai yang dicari dan indeks awal dan akhir array yang akan dicari. Algoritma ini bekerja dengan membagi array menjadi dua bagian, dan memeriksa apakah nilai Kunci berada di setengah kiri atau kanan array. Proses ini diulang secara rekursif hingga elemen ditemukan atau elemen tidak ditemukan. Jika elemen ditemukan, fungsi mengembalikan indeks elemen tersebut. Jika tidak ditemukan, fungsi mengembalikan nilai -1.

Kemudian, program menampilkan pesan hasil pencarian kepada pengguna. Jika nilai Kunci ditemukan pada array, program akan menampilkan pesan "Data ditemukan pada posisi X", dengan X merupakan indeks elemen yang ditemukan. Jika tidak ditemukan, program akan menampilkan pesan "Data tidak ditemukan".

5. Implementasikan pencarian data Pegawai pada tugas pendahuluan dengan ketentuan :
   a. Metode pencarian dapat dipilih.
   b. Pencarian dapat dipilih berdasarkan NIP, Nama dan gabungan keduanya.
   c. Gunakan struktur data array.

   Listing Program:

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

struct Mahasiswa{
    char nrp[25];
    char nama[50];
};

int SequentialSearch(struct Mahasiswa data[], int n, char key[], char
mode[]){
    int i = 0;
    bool found = false;

    while (i < n && !found){
        if(strcmp(mode,"nrp") == 0){
            if(strcmp(data[i].nrp, key) == 0){
                found = true;
            }else{
                i++;
            }
        }else if(strcmp(mode,"nama") == 0){
            if(strcmp(data[i].nama, key) == 0){
                found = true;
            }else{
```

```
                i++;
            }
        }
    }

    if(found){
        return i;
    }else{
        return -1;
    }
}

int Partition(struct Mahasiswa arr[], int p, int r){
    int i = p - 1, j = r + 1;
    struct Mahasiswa pivot = arr[p], temp;

    while (1) {
        do {
            i++;
        } while (strcmp(arr[i].nrp, pivot.nrp) < 0);

        do {
            j--;
        } while (strcmp(arr[j].nrp, pivot.nrp) > 0);

        if (i >= j) {
            return j;
        }

        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}

void QuickSort(struct Mahasiswa arr[], int p, int r){
    int q;

    if(p < r) {
        q = Partition(arr, p, r);
        QuickSort(arr, p, q);
        QuickSort(arr, q+1, r);
    }
}

int BinarySearch(struct Mahasiswa arr[], int n, char key[], char
mode[]){
    int L = 0, R = n - 1, m, hitung = 0;
    bool ketemu = false;

    while (L <= R && !ketemu){
        hitung++;
        m = (L + R) / 2;
        if(strcmp(mode,"nrp") == 0){
            if(strcmp(arr[m].nrp, key) == 0){
```

```
                ketemu = true;
            }else if (strcmp(arr[m].nrp, key) > 0){
                R = m - 1;
            }else{
                L = m + 1;
            }
        }else if(strcmp(mode,"nama") == 0){
            if(strcmp(arr[m].nama, key) == 0){
                ketemu = true;
            }else if (strcmp(arr[m].nama, key) > 0){
                R = m - 1;
            }else{
                L = m + 1;
            }
        }
    }

    if(ketemu){
        return hitung;
    }else{
        return -1;
    }
}

int main(){
    struct Mahasiswa data_awal[MAX];
    int n, ketemu, pencarian;
    char kunci[50], mode[10];

    printf("Masukkan jumlah mahasiswa: ");
    scanf("%d", &n);

    for(int i = 0; i < n; i++){
        printf("Masukkan Nama: ");
        scanf("%s", data_awal[i].nama);
        printf("Masukkan NRP: ");
        scanf("%s", data_awal[i].nrp);
        printf("\n");
    }


    printf("Pencarian berdasarkan NRP atau Nama? (nrp/nama) : ");
    scanf("%s", mode);
    printf("Kunci : ");
    scanf("%s", kunci);
    printf("Pencarian menggunakan Sequiesntial atau Binary(1/2)? ");
    scanf("%d", &pencarian);
    if(pencarian==1){
        ketemu=SequentialSearch(data_awal, n, kunci, mode);
    }else if(pencarian==2){
        QuickSort(data_awal, 0, n-1);
        ketemu=ketemu = BinarySearch(data_awal, n, kunci, mode);
    }

    if(ketemu>=0){
```

```
        printf("Data ditemukan pada indeks ke-%d\n", ketemu);
        printf("NRP: %s\n", data_awal[ketemu].nrp);
        printf("Nama: %s\n", data_awal[ketemu].nama);
    }else{
        printf("Data tidak ditemukan\n");
    }

    return 0;
}
```

Output:

```
Masukkan jumlah mahasiswa: 4
Masukkan Nama: Mirta
Masukkan NRP: 3122500009

Masukkan Nama: Zahrotul
Masukkan NRP: 3122500004

Masukkan Nama: Arsyita
Masukkan NRP: 3122500008

Masukkan Nama: Nadila
Masukkan NRP: 3122500002

Pencarian berdasarkan NRP atau Nama? (nrp/nama) : nama
Kunci : Arsyita
Pencarian menggunakan Sequiesntial atau Binary(1/2)? 1
Data ditemukan pada indeks ke-2
NRP: 3122500008
Nama: Arsyita
```

```
Masukkan jumlah mahasiswa: 4
Masukkan Nama: Mirta
Masukkan NRP: 3122500009

Masukkan Nama: Zahrotul
Masukkan NRP: 3122500002

Masukkan Nama: Arsyita
Masukkan NRP: 3122500008

Masukkan Nama: Nadila
Masukkan NRP: 3122500002

Pencarian berdasarkan NRP atau Nama? (nrp/nama) : nrp
Kunci : 3122500008
Pencarian menggunakan Sequiesntial atau Binary(1/2)? 2
Data ditemukan pada indeks ke-2
NRP: 3122500008
Nama: Arsyita
```

Analisa:

Program ini merupakan sebuah program C yang mengimplementasikan dua metode pencarian data dalam sebuah array of struct Mahasiswa. Metode pertama adalah Sequential Search, yaitu metode pencarian data secara linear dengan melakukan perbandingan satu per satu dari data awal hingga data yang dicari ditemukan atau sampai akhir array. Metode

kedua adalah Binary Search, yaitu metode pencarian data yang memanfaatkan sifat dari array yang telah terurut secara ascending. Program ini menerima input dari pengguna berupa jumlah data mahasiswa, nama, dan NRP, lalu melakukan pencarian berdasarkan NRP atau Nama yang diinginkan pengguna. Pencarian menggunakan metode Sequential atau Binary tergantung pada input pengguna. Jika menggunakan metode Binary Search, maka array data akan diurutkan terlebih dahulu menggunakan metode QuickSort sebelum melakukan pencarian. Program ini akan menampilkan indeks data yang ditemukan beserta NRP dan Nama mahasiswa tersebut jika data ditemukan, atau pesan "Data tidak ditemukan" jika tidak ditemukan.

6. Guessing Game. User memikirkan suatu angka yang berada pada range 1 – n (n diinputkan oleh user). Buat program untuk menebak angka tersebut, tebakan berdasarkan informasi dari user

```
> (guessing-game 100)
Is your number less than 51? yes
Is your number less than 26? no
Is your number less than 38? no
Is your number less than 44? no
Is your number less than 47? yes
Is your number less than 45? no
Is your number less than 46? no
Since your number is less than 47 but not less than 46, it must be
46.
```

Listing Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int guessing_game(int n) {
    srand(time(NULL));
    int batas_bawah = 1;
    int batas_atas = n;

    while (1) {
        int tebakan = (batas_bawah + batas_atas) / 2; // tebakan di
tengah-tengah rentang

        printf("\nApakah angka yang anda masukkan kurang dari %d ? ",
tebakan);

        char jawab[6];
        scanf("%s", jawab);

        if (jawab[0] == 'y' || jawab[0] == 'Y') {
            batas_atas = tebakan - 1;
        } else {
            batas_bawah = tebakan + 1;
        }
```

```
        if (batas_bawah == batas_atas) {
            return batas_bawah;
        }
    }
}

int main() {
    int n;
    printf("Masukkan batas angka terbesar: ");
    scanf("%d", &n);

    printf("\n-------- Tebak Angka --------\n");

    int jawab = guessing_game(n);

    printf("\nAngka yang anda masukkan\t\t: %d\n", jawab);

    return 0;
}
```

Output:

```
Masukkan batas angka terbesar: 100

-------- Tebak Angka --------

Apakah angka yang anda masukkan kurang dari 50 ? t

Apakah angka yang anda masukkan kurang dari 75 ? t

Apakah angka yang anda masukkan kurang dari 88 ? t

Apakah angka yang anda masukkan kurang dari 94 ? t

Apakah angka yang anda masukkan kurang dari 97 ? t

Apakah angka yang anda masukkan kurang dari 99 ? t

Angka yang anda masukkan                : 100
```

Analisa:

Program ini menggunakan algoritma pembagian dua (binary search) untuk menebak angka dengan efisien. Setiap kali program melakukan tebakan, rentang angka yang mungkin akan menyempit menjadi setengah dari rentang sebelumnya. Hal ini memungkinkan program untuk menebak angka dengan jumlah tebakan yang relatif sedikit, terutama jika rentang angka yang dimasukkan oleh pemain cukup besar.

Namun, perlu diperhatikan bahwa program ini asumsi pemain memberikan jawaban yang konsisten dan benar. Jika pemain memberikan jawaban yang tidak sesuai dengan angka yang dipilih, program mungkin akan memberikan hasil yang tidak akurat.

Selain itu, program ini belum melakukan validasi input dari pemain. Jadi jika pemain memasukkan input yang tidak valid, program mungkin tidak berperilaku sebagaimana yang diharapkan.