

MODUL 12: PENCARIAN

12.1. Deskripsi Singkat

Proses pencarian adalah proses menemukan data tertentu di dalam sekumpulan data yang bertipe sama. Pencarian adalah salah satu hal yang fundamental dalam pemrograman. Jika kita membuka aplikasi pada komputer, fitur yang hampir pasti selalu ada adalah fitur pencarian. Entah itu mencari sebuah kata dalam kumpulan teks, mencari data tertentu pada spreadsheet, mencari mahasiswa dengan nama tertentu pada aplikasi database dan lain sebagainya.

Pada modul praktikum ini akan dibahas 2 metode pencarian dasar yaitu pencarian sekuensial dan pencarian biner. Metode pencarian tersebut dapat diterapkan pada struktur data array dan linked list

12.2. Tujuan Praktikum

- 1) Mahasiswa mampu menerapkan pencarian sekuensial dan biner pada array menggunakan bahasa C
- 2) Mahasiswa dapat melakukan operasi pencarian sekuensial dan biner pada linked list menggunakan bahasa C

12.3. Material Praktikum

Kegiatan pada modul ini memerlukan material berupa software editor dan compiler (atau IDE) untuk bahasa pemrograman C.

12.4. Kegiatan Praktikum

A. Pencarian Sekuensial

Pencarian sekuensial atau disebut juga pencarian berurutan atau pencarian linier merupakan metode pencarian yang paling sederhana. Pencarian sekuensial dilakukan dengan cara membandingkan nilai yang dicari pada sekumpulan data mulai dari awal data ke data berikutnya satu per satu hingga nilai yang dicari ditemukan atau tidak ditemukan pada sekumpulan data tersebut.

Dari deskripsi cara kerja pencarian sekuensial di atas, maka perulangan dilakukan mulai dari 1 hingga sejumlah n data. Jika beruntung maka nilai yang dicari ditemukan pada data pertama yang dicek. Jika tidak maka nilai yang dicari tidak ditemukan hingga akhir data. Proses pencarian dihentikan jika data yang dicari ditemukan, atau setelah pengecekan dilakukan hingga akhir data. Runtime pencarian sekuensial secara umum adalah $O(n)$.

A.1. Pencarian Sekuensial Pada Data Tidak Terurut (Array)

Berikut adalah listing program pencarian sekuensial pada data tidak terurut (array). Anda bisa menyimpannya pada file **modul12a1.c**.

```

#include <stdio.h>
#define MAX 100 //ukuran maksimum array

void fill_data(int data[], int *size){ //mengisi data
    printf("Input ukuran array (max 100): ");
    scanf("%d", size);
    for(int i=0;i<*size;i++)
        printf("input data ke-%d :",i+1);
        scanf("%d",&data[i]);
    }
}

int seq_search(int data[], int size, int x){
    for (int i=0;i<size;i++){
        if(data[i]==x) return i;
    }
    return -1;
}

void main(){
    int data[MAX];
    int size; //ukuran array
    int x;
    fill_data(data,&size);
    printf("nilai yang mau dicari: ");
    scanf("%d", &x);
    if(seq_search(data,size,x)==-1) printf("tidak ditemukan");
    else printf("ditemukan pada indeks ke-
%d",seq_search(data,size,x));
}

```

Ketika dijalankan akan menghasilkan output sebagai berikut:

```
C:\Users\Panasonic\Documents\codeblocks\arraysearch.exe
Input ukuran array (max 100): 6
Input data: 2 4 3 6 5 1
nilai yang mau dicari: 6
ditemukan pada indeks ke-3
Process returned 26 (0x1A)   execution time : 22.368 s
Press any key to continue.
```

A.2. Pencarian Sekuensial Pada Data Tidak Terurut (Linked List)

Berikut adalah contoh listing program pencarian sekuensial pada data tidak terurut (linked list). Anda bisa menyimpannya pada file **modul12a2.c**.

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

struct node{
    int value;
    struct node *next;
};

typedef struct node *ptrnode;

ptrnode head = NULL;
int jumnode; //jumlah node

ptrnode insert(int nilai){
    ptrnode p,q;
    p = (ptrnode)malloc(sizeof(struct node));
    p->value = nilai;
    p->next = NULL;
    if(head==NULL) {
        head=p;
    }
    else {
        tmp = head;
        while(q->next!=NULL) {
            tmp = tmp->next;
        }
    }
}
```

```

        tmp->next = p;
    }
    return(head);
}

void isi_data(){
    int k;
    printf("input jumlah node: ");
    scanf("%d",&jumnode);
    for(int j=1;j<=jumnode;j++){
        printf("input data ke-%d :",j);
        scanf("%d", &k);
        head=insert(k);
    }
}

int search(int x){ //x adalah nilai yang dicari
    int j = 1;
    ptrnode tmp=head;
    while(tmp != NULL){
        if(x==tmp->value){
            return j;
        }
        else{
            tmp = tmp->next;
            j++;
        }
    }
    return -1; //jika tidak ada yang dicari return -1
}

void bersihkan_memori(){
    while(head != NULL){
        ptrnode tmp = head;
        head = head->next;
    }
}

```

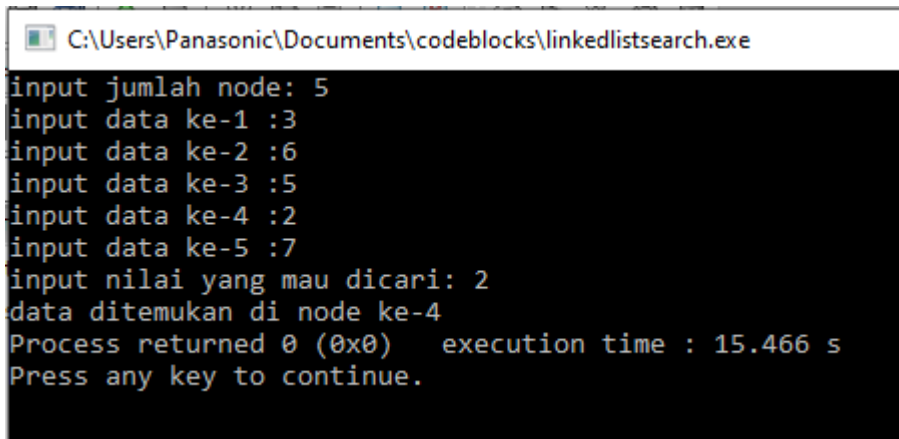
```

        tmp->next = NULL;
        free(tmp);
    }
}

void main(){
    isi_data();
    int x;
    printf("input nilai yang mau dicari: ");
    scanf("%d",&x);
    if (search(x) == -1) printf("data tidak ditemukan");
    else printf("data ditemukan di node ke-%d",search(x));
    bersihkan_memori();
}

```

Jika kode di atas dijalankan maka outputnya sebagai berikut:



```

C:\Users\Panasonic\Documents\codeblocks\linkedlistsearch.exe
input jumlah node: 5
input data ke-1 :3
input data ke-2 :6
input data ke-3 :5
input data ke-4 :2
input data ke-5 :7
input nilai yang mau dicari: 2
data ditemukan di node ke-4
Process returned 0 (0x0)   execution time : 15.466 s
Press any key to continue.

```

A.3. Pencarian Sekuensial Pada Data Terurut

Pada pencarian sekuensial pada data yang terurut, proses pencarian adalah proses iterasi yang dimulai dari data indeks pertama pada array sampai data yang dicari ditemukan atau nilai yang dicari sudah melebihi atau kurang dari nilai data pada indeks array yang dikunjungi atau sampai data pada indeks terakhir. Jika diterapkan pada linked list, maka pencarian dimulai dari head atau node pertama hingga node terakhir atau tail, sampai data yang dicari ditemukan atau nilai yang dicari sudah melebihi atau kurang dari nilai data pada node yang dikunjungi atau sampai data pada node tail.

Dari pernyataan di atas, terdapat 3 kondisi dimana proses pencarian berhenti yaitu:

1. Jika data yang dicari ditemukan
2. Jika nilai data pada indeks array atau node yang sedang dikunjungi sudah melebihi atau kurang dari nilai yang dicari (tergantung data terurut menaik atau menurun).

3. Jika sudah sampai di data indeks array atau node terakhir.

Modifikasilah file **modul12a1.c** dan file **modul12a2.c** supaya dapat mengakomodasi pencarian sekuensial pada data yang terurut, baik urut menaik atau menurun.

B. Pencarian Biner

Pencarian biner atau pencarian bagi dua hanya bisa dilakukan pada data yang terurut. Pencarian dilakukan dengan cara mengecek elemen tengah. Jika data di elemen tengah sama dengan yang dicari, maka ditemukan. Jika data di elemen tengah lebih besar dari data yang dicari, maka pencarian dilanjutkan di bilah kiri, jika data di elemen tengah lebih kecil dari data yang dicari, maka pencarian dilanjutkan di bilah kanan. Begitu seterusnya hingga data yang dicari ditemukan atau semua elemen sudah diperiksa.

Performa dari pencarian biner $O(2^{\log n})$ jauh lebih cepat dibandingkan pencarian sekuensial $O(n)$. Fungsi pencarian biner dapat dinyatakan sebagai fungsi rekursif atau iteratif. berikut ini adalah fungsi pencarian biner secara iteratif pada array. Simpanlah pada file **modul12b1.c**

```
#include <stdio.h>
#define MAX 100 //ukuran maksimum array

void fill_data(int data[], int *size){ //mengisi data
    printf("Input ukuran array (max 100): ");
    scanf("%d", size);
    printf("Input data ascending: ");
    for(int i=0;i<*size;i++){
        scanf("%d",&data[i]);
    }
}

int binary_search(int data[], int size, int x){
    int L = 0;
    int H = size-1;
    int M = -1;
    int index = -1;

    while(L<=H){
        M = (L+H)/2;
        if(data[M]==x) return M;
        else{
```

```

        if(data[M] < x) L = M + 1;
        else H = M - 1;
    }
}
return -1;
}

void main(){
    int data[MAX];
    int size; //ukuran array
    int x;
    fill_data(data,&size);
    printf("nilai yang mau dicari: ");
    scanf("%d", &x);
    if(binary_search(data,size,x)==-1) printf("tidak
ditemukan");
    else printf("ditemukan pada indeks ke-
%d",binary_search(data,size,x));
}

```

Jika program di atas dijalankan maka kurang lebih outputnya seperti ini:

```

C:\Users\Panasonic\Documents\codeblocks\arraysearchbinary.exe
Input ukuran array (max 100): 6
Input data ascending: 9 13 15 17 21 29
nilai yang mau dicari: 29
ditemukan pada indeks ke-5
Process returned 26 (0x1A) execution time : 32.851 s
Press any key to continue.

```

Pada linked list, pencarian biner dapat diterapkan juga namun perlu runtime yang lebih lama pada pencarian node tengah. Berikut adalah contoh listing program untuk pencarian biner pada single linked list. Simpanlah pada file **modul12b2.c**

```

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

struct node{
    int value;
    struct node *next;
};

```

```

typedef struct node *ptrnode;

ptrnode head,tail = NULL;
int jumnode;

ptrnode insert(int nilai){
    ptrnode p;
    p = (ptrnode)malloc(sizeof(struct node));
    p->value = nilai;
    p->next = NULL;
    if(head==NULL) {
        head=p;
        tail=head;
    }
    else {
        tail = head;
        while(tail->next!=NULL){
            tail = tail->next;
        }
        tail->next = p;
        tail = p;
    }
    return(head);
}

void isi_data(){
    int j,k;
    printf("input jumlah node (data harusurut ascending): ");
    scanf("%d",&jumnode);
    for(j=1;j<=jumnode;j++){
        printf("input data ke-%d :",j);
        scanf("%d", &k);
        head=insert(k);
    }
}

```



```

}

ptrnode middle(ptrnode start, ptrnode last){
//untuk mendapatkan node tengah
    if (start == NULL) return NULL;
    ptrnode slow = start;
    ptrnode fast = start -> next;
    while (fast != last){
        fast = fast -> next;
        if (fast != last){
            slow = slow -> next;
            fast = fast -> next;
        }
    }
    return slow;
}

ptrnode binarySearch(int x){
    ptrnode start = head;
    ptrnode last = NULL;
    do{
        // temukan node tengah
        ptrnode mid = middle(start, last);
        // jika node tengah NULL
        if (mid == NULL) return NULL;
        // Jika x ditemukan di node tengah
        if (mid -> value == x) return mid;
        // Jika nilai x lebih dari node tengah
        else if (mid -> value < x) start = mid -> next;
        // Jika nilai x kurang dari node tengah
        else last = mid;
    } while (last == NULL || last != start);
    // jika tidak ditemukan
    return NULL;
}

```

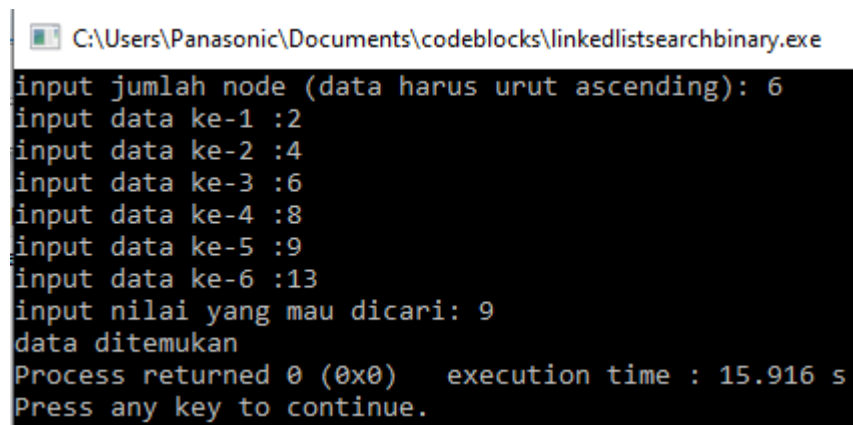
```

void bersihkan_memori() {
    while(head != NULL) {
        ptrnode tmp = head;
        head = head->next;
        tmp->next = NULL;
        free(tmp);
    }
}

void main() {
    isi_data();
    int x;
    printf("input nilai yang mau dicari: ");
    scanf("%d", &x);
    if(binarySearch(x) == NULL) printf("data tidak ditemukan");
    else printf("data ditemukan");
    bersihkan_memori();
}

```

Jika kode di atas dijalankan maka outputnya seperti ini:



```

C:\Users\Panasonic\Documents\codeblocks\linkedlistsearchbinary.exe
input jumlah node (data harus urut ascending): 6
input data ke-1 :2
input data ke-2 :4
input data ke-3 :6
input data ke-4 :8
input data ke-5 :9
input data ke-6 :13
input nilai yang mau dicari: 9
data ditemukan
Process returned 0 (0x0)   execution time : 15.916 s
Press any key to continue.

```

Performa pencarian biner pada linked list tidak bisa secepat pada array karena penentuan node yang akan dikunjungi tidak menggunakan random access seperti array. Runtime pencarian biner pada doubly-linked list adalah $O(n^2 \log n)$, dibandingkan runtime pencarian sekuensial pada array sebesar $O(n)$. Dengan sedikit modifikasi, performa pencarian biner pada doubly linked list dapat ditingkatkan menjadi $O(2 \log n)$.