

**TUGAS PENDAHULUAN**  
**MODUL 5**  
**“Single Linked List (Bagian Kedua)”**



**Disusun Oleh:**  
**Isabelle Putri Ardini - 2311104030**  
**SE-07-01**

**Dosen:**  
**Yudha Islami Sulistya**

**PROGRAM STUDI SOFTWARE ENGINEERING**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2024**

## Soal Tugas Pendahuluan

### Soal 1: Mencari Elemen Tertentu dalam SLL

**Deskripsi Soal:** Buatlah program yang mengizinkan pengguna memasukkan 6 elemen interger ke dalam list. Implementasikan funtion **searchElement** untuk mencari apakah sebuah nilai tertentu ada dalam list.

#### Instruksi

1. Minta pengguna untuk memasukkan nilai yang ingin dicari.
2. Jika nilai ditemukan, tamplkan alamat dan posisi dalam angka (contoh: urutan ke 4) pada list tersebut.
3. Jika nilai tidak ditemukan, tampilkan pesan bahwa elemen tersebut tidak ada dalam list tersebut.

#### NB:

1. Gunakan pendekatan linier search untuk mencari elemen.

#### Sub-Program:

```
Function searchElement( L : list, i : integer)
{ I.S. List tidak kosong.
  F.S. Menampilkan alamat dan posisi elemen i jika ditemukan}
Dictionary
    current: address
    position: int
Algorithms
    current <- L.head
    position <- 1

    //melakukan perulangan selama i belum ditemukan dan posisi
    current belum berada pada
    akhir list
    While .....
        //seiring pointer (current) bergerak, position bertambah
        . . . . .
        //lakukan perpindahan current
        . . . . .
    endwhile
    //jika i ditemukan maka tampilkan alamat dan posisi
    if....
        output(...)
    //jika tidak ditemukan maka tampilkan pesan yang menyatakan
    hal tsb
    else...
        output(...)
    endif
endfunction
```

**Program:**

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambah elemen baru di akhir list
void append(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;

    if (*head_ref == nullptr) {
        *head_ref = new_node;
        return;
    }

    Node* last = *head_ref;
    while (last->next != nullptr) {
        last = last->next;
    }
    last->next = new_node;
}
```

```

// Procedure untuk menambahkan elemen secara terurut ke dalam list
void insertSorted(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;

    // Jika list kosong atau elemen baru lebih kecil dari elemen head
    if (*head_ref == nullptr || (*head_ref)->data >= new_data) {
        new_node->next = *head_ref;
        *head_ref = new_node;
        return;
    }

    // Temukan posisi yang sesuai untuk elemen baru
    Node* current = *head_ref;
    while (current->next != nullptr && current->next->data < new_data) {
        current = current->next;
    }

    new_node->next = current->next;
    current->next = new_node;
}

// Fungsi untuk mencetak list
void printList(Node* node) {
    while (node != nullptr) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

```

```

int main() {
    Node* head = nullptr;
    int input;

    // Memasukkan 4 elemen integer ke dalam list
    cout << "Masukkan 4 elemen integer secara terurut ke dalam list:" << endl;
    for (int i = 0; i < 4; i++) {
        cout << "Elemen ke-" << (i + 1) << ": ";
        cin >> input;
        insertSorted(&head, input);
    }

    cout << "\nList setelah memasukkan 4 elemen: ";
    printList(head);

    // Meminta pengguna memasukkan elemen tambahan untuk disisipkan
    cout << "Masukkan elemen tambahan yang akan disisipkan secara terurut: ";
    cin >> input;

    // Menyisipkan elemen baru secara terurut
    insertSorted(&head, input);

    cout << "List setelah menambahkan elemen baru: ";
    printList(head);

    return 0;
}

```

#### Hasil Running:

```

Masukkan 6 elemen integer ke dalam list:
Elemen ke-1: 4
Elemen ke-2: 6
Elemen ke-3: 15
Elemen ke-4: 2
Elemen ke-5: 78
Elemen ke-6: 33
Masukkan elemen yang ingin dicari: 33

```

#### Soal 2: Menggunakan List Menggunakan Bubble Sort

**Deskripsi Soal:** Buatlah program yang mengizinkan pengguna memasukkan 5 elemen integer ke dalam list. Implementasikan procedure bubbleSortList untuk mengurutkan elemen-elemen dalam list dari nilai terkecil ke terbesar.

#### Instruksi

1. Setelah mengurutkan, tampilkan elemen-elemen list urutan yang benar.

## Langkah-langkah Bubble Sort pada SLL

1. Inisialisasi:
  - Buat pointer current yang akan digunakan untuk menelusuri list.
  - Gunakan variabel boolean swapped untuk mengawasi apakah ada pertukaran yang dilakukan pada iterasi saat ini.
2. Traversing dan Pertukaran:
  - Lakukan iterasi berulang sampai tidak ada pertukaran yang dilakukan:
    - o Atur swapped ke false di awal setiap iterasi.
    - o Set current ke head dari list.
    - o Selama current.next tidak null (masih ada node berikutnya):
      - Bandingkan data pada node current dengan data pada node current.next.
      - Jika data pada current lebih besar dari data pada current.next, lakukan pertukaran:
        - Tukar data antara kedua node (bukan pointer).
        - Set swapped menjadi true untuk menunjukkan bahwa ada pertukaran yang dilakukan.
      - Pindahkan current ke node berikutnya (current = current.next).
3. Pengulangan:
  - Ulangi langkah 2 sampai tidak ada lagi pertukaran yang dilakukan (artinya list sudah terurut).

## Contoh Proses Bubble Sort

- List awal : 4 – 2 – 3 – 1 dan akan melakukan sorting membesar / ascending
- Iterasi pertama:
  - Bandingkan 4 dan 2:  $4 > 2$ , lakukan penukaran, 2 – 4 – 3 – 1
  - Bandingkan 4 dan 3:  $4 > 3$ , lakukan penukaran, 2 – 3 – 4 – 1
  - Bandingkan 4 dan 1:  $4 > 1$ , lakukan penukaran, 2 – 3 – 1 – 4
  - Kondisi list di akhir iterasi: 2 – 3 – 1 – 4
- Iterasi kedua:
  - Bandingkan 2 dan 3:  $2 < 3$ , tidak terjadi penukaran
  - Bandingkan 3 dan 1:  $3 > 1$ , lakukan penukaran, 2 – 1 – 3 – 4
  - Bandingkan 3 dan 4:  $3 < 4$ , tidak terjadi penukaran
  - Kondisi list di akhir iterasi: 2 – 1 – 3 – 4
- Iterasi ketiga:
  - Bandingkan 2 dan 1:  $2 > 1$ , lakukan penukaran, 1 – 2 – 3 – 4
  - Bandingkan 2 dan 3:  $2 < 3$ , tidak terjadi penukaran
  - Bandingkan 3 dan 4 :  $3 < 4$ , tidak terjadi penukaran
  - Kondisi list di akhir iterasi: 1 – 2 – 3 – 4

## Sub-Program:

```
Procedure bubbleSort( in/out L : list )  
{ I.S. List tidak kosong.
```

F.S. elemen pada listurut membesar berdasarkan infonya}

### Program:

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambah elemen baru di akhir list
void append(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;

    if (*head_ref == nullptr) {
        *head_ref = new_node;
        return;
    }

    Node* last = *head_ref;
    while (last->next != nullptr) {
        last = last->next;
    }
    last->next = new_node;
}
```

```
// Procedure untuk mengurutkan elemen-elemen dalam list menggunakan Bubble Sort
void bubbleSortlist(Node* head) {
    bool swapped;
    Node* current;
    Node* last = nullptr;

    // Ulangi proses sorting sampai tidak ada pertukaran
    do {
        swapped = false;
        current = head;

        while (current->next != last) {
            if (current->data > current->next->data) {
                swap(current->data, current->next->data);
                swapped = true;
            }
            current = current->next;
        }
        last = current; // Setelah iterasi, elemen terbesar akan berada di akhir list
    } while (swapped);
}
```

```

// Fungsi untuk mencetak list
void printList(Node* node) {
    while (node != nullptr) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    int input;

    // Memasukkan 5 elemen integer ke dalam list
    cout << "Masukkan 5 elemen integer ke dalam list:" << endl;
    for (int i = 0; i < 5; i++) {
        cout << "Elemen ke-" << (i + 1) << ": ";
        cin >> input;
        append(&head, input);
    }

    cout << "\nList sebelum diurutkan: ";
    printList(head);

    // Mengurutkan list menggunakan Bubble Sort
    bubbleSortList(head);

    cout << "List setelah diurutkan: ";
    printList(head);

    return 0;
}

```

### Hasil Running:

```

Masukkan 5 elemen integer ke dalam list:
Elemen ke-1: 4
Elemen ke-2: 8
Elemen ke-3: 13
Elemen ke-4: 27
Elemen ke-5: 7

List sebelum diurutkan: 4 8 13 27 7
List setelah diurutkan: 4 7 8 13 27

```

### Soal 3: Menambahkan Elemen Secara Terurut

**Deskripsi Soal:** Buatlah program yang mengizinkan pengguna memasukkan 4 elemen integer ke dalam list secara manual. Kemudian, minta pengguna memasukkan elemen



tambahan yang harus ditempatkan di posisi yang sesuai sehingga list tetap terurut.

### Instruksi

1. Implementasikan procedure insertSorted untuk menambahkan elemen baru ke dalam list yang sudah terurut.
2. Tampilkan list setelah elemen baru dimasukkan.

### Sub-Program:

**Procedure** insertSorted( **in/out** L : **list**, **in** P : **address**)

{ I.S. List tidak kosong.

F.S. Menambahkan elemen secara terurut}

Dictionary

Q, Prev: address

found: bool

Algorithms

Q <- L.head

found <- false

//melakukan perulangan selama found masih false dan Q masih menunjuk elemen pada list

While .....

    //melakukan pengecekan apakah info dari elemen yang ditunjuk memiliki nilai lebih kecil dari pada P

    if ....

        //jika iya maka Prev diisi elemen Q, dan Q diisi elemen setelahnya

        ....

        //jika tidak maka isi found dengan nilai 'true'

    else

        . . .

    Endif

    //lakukan perpindahan Q

    ....

endwhile

//melakukan pengecekan apakah Q elemen head

if ....

    //jika iya, maka tambahkan P sebagai head

    ....

//melakukan pengecekan apakah Q berisi null (sudah tidak menunjuk elemen pada list

else if ...

    //jika iya, maka tambahkan P sebagai elemen terakhir

    ...

//jika tidak keduanya, maka tambahkan P pada posisi diantara Prev dan Q

```
        else
            ....
        endif
    endprocedure
```

### Program:

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambah elemen baru di akhir list
void append(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;

    if (*head_ref == nullptr) {
        *head_ref = new_node;
        return;
    }

    Node* last = *head_ref;
    while (last->next != nullptr) {
        last = last->next;
    }
    last->next = new_node;
}
```

```

// Procedure untuk menambahkan elemen secara terurut ke dalam list
void insertSorted(Node** head_ref, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;

    // Jika list kosong atau elemen baru lebih kecil dari elemen head
    if (*head_ref == nullptr || (*head_ref)->data >= new_data) {
        new_node->next = *head_ref;
        *head_ref = new_node;
        return;
    }

    // Temukan posisi yang sesuai untuk elemen baru
    Node* current = *head_ref;
    while (current->next != nullptr && current->next->data < new_data) {
        current = current->next;
    }

    new_node->next = current->next;
    current->next = new_node;
}

// Fungsi untuk mencetak list
void printList(Node* node) {
    while (node != nullptr) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

```

```

int main() {
    Node* head = nullptr;
    int input;

    // Memasukkan 4 elemen integer ke dalam list
    cout << "Masukkan 4 elemen integer secara terurut ke dalam list:" << endl;
    for (int i = 0; i < 4; i++) {
        cout << "Elemen ke-" << (i + 1) << ": ";
        cin >> input;
        insertSorted(&head, input);
    }

    cout << "\nList setelah memasukkan 4 elemen: ";
    printList(head);

    // Meminta pengguna memasukkan elemen tambahan untuk disisipkan
    cout << "Masukkan elemen tambahan yang akan disisipkan secara terurut: ";
    cin >> input;

    // Menyisipkan elemen baru secara terurut
    insertSorted(&head, input);

    cout << "List setelah menambahkan elemen baru: ";
    printList(head);

    return 0;
}

```

### Hasil Running:

```

Masukkan 4 elemen integer secara terurut ke dalam list:
Elemen ke-1: 3
Elemen ke-2: 12
Elemen ke-3: 9
Elemen ke-4: 20

List setelah memasukkan 4 elemen: 3 9 12 20
Masukkan elemen tambahan yang akan disisipkan secara terurut: 6
List setelah menambahkan elemen baru: 3 6 9 12 20

```