TUGAS PENDAHULUAN MODUL 5
STRUKTUR DATA

Disusun Oleh :

Nurul Maulina Nainggolan

21104053/SE07-01

PROGRAM STUDI REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERAITY PURWOKERTO

2024

MODUL 5

TUGAS PENDAHULUAN MODUL 5

STRUKTUR DATA- GANJIL 2024/2025

"Single Linked List (Bagian Kedua)"

1. **Soal 1 : Mencari Elemen Tertentu dalam SLL**

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

   Instruksi :

   a. Minta pengguna untuk memasukkan nilai yang ingin dicari
   b. Jika nilai ditemukan, tampilan alamat dan posisi dalam angka (contoh : urutan ke-4) Pada list tersebut
   c. Jika nilai ditemukan, tampilkan pesan bahwa elemen tersebut tidak ada dalam list tersebut.

NB : 1. Gunakan pendekatan linear search untuk mencati 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
```

Pendeklarasian :

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

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

//menambahkan elemen ke dalam linked list
void append(Node*&head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next=nullptr;

    if(head == nullptr) {
        head = newNode;
    } else{
        Node* temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next=newNode;

    }
}

//mencari elemen
void searchElement(Node* head, int cari_elmn) {
    Node* current = head;
    int position= 1;
    bool found = false;

    while(current != nullptr) {
        if(current->data == cari_elmn) {
            found = true;
            cout << "Elemen ditemukan di alamat: " << current << "pada urutan ke- " << position << endl;
            break;
        }
        current = current->next;
        position++;
    }

    if (!found) {
        cout << "Elemen" << cari_elmn << "tidak ditemukan dalam list!!" << endl;
    }

}



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

    //memasukkan 6 elemen
    cout << "Masukkan 6 Elemen ke Dalam List: " << endl;
    for(int i=0; i<6; i++) {
        cout << "Elemen Ke- " << i + 1 << ": ";
        cin >> value;
        append(head, value);
    }

    //memasukkan elemen
    int cari_elm;
    cout << "Masukkan Elemen yang dicari: ";
    cin >> cari_elm;


    //cari elemen dalam linked list
    searchElement(head, cari_elm);

    return 0;
}
```

Output :

```
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2> cd TP
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP> g++ Latihan1.cpp -o Latihan1
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP> ./Latihan1
Masukkan 6 Elemen ke Dalam List:
Elemen Ke- 1: 1
Elemen Ke- 2: 2
Elemen Ke- 3: 3
Elemen Ke- 4: 4
Elemen Ke- 5: 5
Elemen Ke- 6: 6
Masukkan Elemen yang dicari: 4
Elemen ditemukan di alamat: 0xbd6f38pada urutan ke- 4
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP>
```

Pada soal ini mengarahkan untuk membuat list berisi 6 elemen, dengan tugas mencari elemen diurutan keberapa dan di alamat mana. Pada sintak ini menggunakan seacrhElement untuk mencari elemen yang diminta. Jika ditemukan, akan menampilkan alamat memori dan urutan, namun jika tidak ditemuakn akan menampilkan pesan tidak ditemukan.

2. **Soal 2 : Mengurutkan 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 : Setelah mengurutkan, tampilkan elemen-elemen list dalam urutan yang benar

Langkah – Langkah Buble Sort pada SLL

a. 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.

b. Traversing dan Pertukaran
   - Lakukan iterasi berulang sampai tidak ada pertukaran yang dilakukan:
     - Atur swapped ke false di awal setiap iterasi.
     - Set current ke head dari list.
     - 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).

c. 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 list urut membesar berdasarkan infonya}
```

Pendeklarasian :

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

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

//menambahkan elemen ke dalam linked list
void append(Node*&head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next=nullptr;

    if(head == nullptr) {
        head = newNode;
    } else{
        Node* temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next=newNode;

    }
}

//menampilkan elemen
void displayList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}


//mengurutkan linked list
void bubbleSortList(Node* head) {
    if (head ==nullptr)
    return;

    bool swapped;
    Node* current;
    Node* lastPtr = nullptr;

    do {
        swapped = false;
        current = head;

        while(current->next !=lastPtr) {
            if(current->data > current->next->data) {
                int temp = current->data;
                current->data = current->next->data;
                current->next->data = temp;
                swapped = true;
            }
            current = current->next;
        }
        lastPtr = current;
    }while (swapped);
}




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

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

    // Menampilkan list sebelum sorting
    cout << "List sebelum diurutkan: ";
    displayList(head);

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

    // Menampilkan list setelah sorting
    cout << "List setelah diurutkan: ";
    displayList(head);

    return 0;
}
```

Output :

```
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP> g++ Latihan2.cpp -o Latihan2
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP> ./Latihan2
Masukkan 5 elemen integer ke dalam list:
Elemen ke-1: 2
Elemen ke-2: 6
Elemen ke-3: 4
Elemen ke-4: 9
Elemen ke-5: 10
List sebelum diurutkan: 2 6 4 9 10
List setelah diurutkan: 2 4 6 9 10
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP>
```

Pada soal ini mengajarkan bagaimana cara mengurutkan elemen menggunakan bubble sort. Pada sintak ini menggunakan perintah append untuk menambahkan elemen baru, displayList untuk menampilkan elemet linked list dan bubblesortList untuk mengimplementasikan elemen dengan mengurutkannya. Jika elemen yang sedang diperiksa lebih besar dari elemen berikutnya, maka data akan ditukar, proses ini akan berlangsung sampai tidak ada lagi elemen yang di tukar.

3. **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 :

a. Implementasikan procedure **insertSorted** untuk menambahkan elemen baru ke dalam list yang sudah terurut.
b. 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
```

Pendeklarasian :

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

// Definisi struktur node
struct Node {
    int data;
    Node* next;
};

// menambahkan elemen
void insertSorted(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;

    // Jika list kosong atau nilai yang dimasukkan lebih kecil dari head
    if (head == nullptr || head->data >= value) {
        newNode->next = head;
        head = newNode;
    } else {
        Node* current = head;
        Node* prev = nullptr;

        // menyisipkan elemen baru
        while (current != nullptr && current->data < value) {
            prev = current;
            current = current->next;
        }

        // Menyisipkan elemen baru di posisi yang sesuai
        prev->next = newNode;
        newNode->next = current;
    }
}

//  menampilkan elemen-elemen dalam linked list
void displayList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

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

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

    // Menampilkan list setelah input awal
    cout << "List setelah input awal: ";
    displayList(head);

    // memasukkan elemen tambahan
    cout << "Masukkan elemen tambahan yang ingin dimasukkan secara terurut: ";
    cin >> value;
    insertSorted(head, value);

    // Menampilkan list setelah elemen tambahan dimasukkan
    cout << "List setelah elemen tambahan dimasukkan: ";
    displayList(head);

    return 0;
}
```

Output :

```
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP> g++ Latihan3.cpp -o Latihan3
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP> ./Latihan3
Masukkan 4 elemen integer ke dalam list (urutkan secara manual):
Elemen ke-1: 4
Elemen ke-2: 78
Elemen ke-3: 12
Elemen ke-4: 0
List setelah input awal: 0 4 12 78
Masukkan elemen tambahan yang ingin dimasukkan secara terurut: 34
List setelah elemen tambahan dimasukkan: 0 4 12 34 78
PS C:\Users\nurul\OneDrive - ypt.or.id\STRUKTUR DATA\PRAKTIKUM\05_Single_Linked_List_Bagian_2\TP>
```

Pada soal terakhir, mengajarkan tentang bagaimana menyisipkan elemen pada list. Pada bagian ini pengguna disuruh memasukkan 4 elemen kemudian menambahkan elemen baru dengan menggunakan fungsi insertSorted yang digunakan untuk menyisipkan elemen tambahan pada posisi yang sesuai agar list tetap berurut.