LAPORAN PRAKTIKUM

MODUL 5

“SINGLE LINKED LIST (BAGIAN KEDUA)”

Disusun Oleh:

MARSELLA DWI JULIANTI (2311104004)

SE 07-01

Dosen :

Yudha Islami Sulistya, S.kom, M.Cs.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

SOAL TP
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

1. Minta pengguna untuk memasukan nilai yang ingin dicari.
2. Jika nilai ditemukan, tampilan 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
```

Jawab:

Inputan file SOAL01.cpp:

TP > SOAL01.cpp > ...

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

// Struktur node untuk Single Linked List
struct Node {
    int data;
    Node* next;
};

// Kelas untuk mengelola Single Linked List
class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    // Fungsi untuk menambahkan elemen di akhir list
    void insert(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;
        }
    }

    // Fungsi untuk mencari elemen tertentu
    void searchElement(int value) {
        Node* current = head;
        int position = 1;

        while (current != nullptr) {
            if (current->data == value) {
                cout << "Elemen " << value << " ditemukan pada posisi ke-" << position << endl;
                return;
            }
            current = current->next;
            position++;
        }
        cout << "Elemen " << value << " tidak ditemukan dalam list." << endl;
    }
};
```

Pada program diatas, terdapat:

1. Header dan namespace:
    - '#include <iostream>' dimana untuk mengimpor library untuk input dan outputnya.
    - 'using namespace std;' dimana ini menggunakan namespace standar untuk menghindari penulisan 'std::' di depan setiap penggunaan fungsi dari library standar.
2. Struktur node:
    - 'struct Node', mendefiniskan struktru untuk node dalam linked list, yang berisi 'int data' untuk menyimpan data integer dan 'Node* next' dimana pointer yang menunjuk ke node berikutnya.
3. Kelas LinkedList:
    - 'LinkedList()' untuk mengelola operasi pada linked list.
    - 'Node* head' untuk pointer yang menunjuk ke node pertama (head) dari linked list.
4. Konstruktor:
    - 'LinkedList()' dimana ini menginsisalisasi objek 'LinkedList' dengan 'head' diatur ke 'nullptr' yang menunjukkan bahwa list kosong.
5. Fungsi 'insert(int value)' untuk membuat node baru dengan nilai yang diberkan. Jika list kosong ('head == nullptr') node baru menjadi head. Jika tidak, traverse akan menulurusuri list hingga akhir untuk menambahkan node baru diakhir.
6. Fungsi 'searchElement(int value)' dimana ini untuk memulai pencarian dari head dan menggunakan loop 'while' untuk menelurusi setiap node.

Jika data pada node saat ini (‘current->data’) sama dengan ‘value’ yang dicari, maka tampilkan posisi dan keluar dari fungsi. Jika tidak, pindah ke node berikutnya dan tingkatkan posisi. Dan jika lopp selesai dan elemen tidak ditemukan, makan tampilkan pesan bahwa elemen tidak ada dalam list.

Inputan file main.cpp:

```cpp
#include "SOAL01.cpp"

int main() {
    LinkedList list;

    // Minta pengguna memasukkan 6 elemen
    for (int i = 0; i < 6; i++) {
        int value;
        cout << "Masukkan elemen ke-" << (i + 1) << ": ";
        cin >> value;
        list.insert(value);
    }

    // Minta pengguna untuk memasukkan nilai yang ingin dicari
    int searchValue;
    cout << "Masukkan nilai yang ingin dicari: ";
    cin >> searchValue;

    // Mencari elemen
    list.searchElement(searchValue);

    return 0;
}
```

Pada program diatas, fungsi ‘main()’ disini untuk membuat objek ‘LinkedList’ yang menggunakan lopp untuk meminta pengguna memasukkan 6 elemen ke dalam list menggunakan fungsi ‘insert()’.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\05_Single_Linked_List_Bagian_2> cd 'd:\SEMESTER
R DATA\05_Single_Linked_List_Bagian_2\TP\output'
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\05_Single_Linked_List_Bagian_2\TP\output> & .\'
Masukkan elemen ke-1: 3
Masukkan elemen ke-2: 4
Masukkan elemen ke-3: 5
Masukkan elemen ke-4: 2
Masukkan elemen ke-5: 6
Masukkan elemen ke-6: 8
Masukkan nilai yang ingin dicari: 6
Elemen 6 ditemukan pada posisi ke-5
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\05_Single_Linked_List_Bagian_2\TP\output>
```

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 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:
     - 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).
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 list urut membesar berdasarkan infonya}
```

Jawab:

Inputan file SOAL02.cpp:

TP > SOAL02.cpp > ...

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

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

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void insert(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;
        }
    }
```

```
33
34      void bubbleSort() {
35          if (head == nullptr) return;
36
37          bool swapped;
38          do {
39              swapped = false;
40              Node* current = head;
41
42              while (current->next != nullptr) {
43                  if (current->data > current->next->data) {
44                      // Tukar data
45                      int temp = current->data;
46                      current->data = current->next->data;
47                      current->next->data = temp;
48                      swapped = true;
49                  }
50                  current = current->next;
51              }
52          } while (swapped);
53      }
54
55      void display() {
56          Node* temp = head;
57          while (temp != nullptr) {
58              cout << temp->data << " ";
59              temp = temp->next;
60          }
61          cout << endl;
62      }
63  };
```

Pada program diatas, terdapat:

1. Header dan namespace:
    - ‘#include <iostream>’ dimana untuk mengimpor library untuk input dan outputnya.
    - ‘using namespace std;’ dimana ini menggunakan namespace standar untuk menghindari penulisan ‘std::’ di depan setiap penggunaan fungsi dari library standar.
2. Struktur node:
    - ‘struct Node’ ini mendefinisikan struktur untuk node dalam linked list, yang berisi ‘int data’ yang menyimpan data integer dan ‘Node* next’ untuk pointer yang menunjuk ke node berikutnya.
3. Kelas LinkedList, dimana terdapat ‘class LinkedList’ yang mengelola operasi pada linked list dan ‘Node* head’ untuk pointer yang menunjuk ke node pertama (head) dari linked list.
4. Konstruktor, terdapat ‘LinkedList()’ yang menginsialisasi objek ‘LinkedList’ dengan ‘head diatur ke ‘nullptr’ yang menunjuk bahwa list kosong.
5. Fungsi ‘insert(int value)’, dimana membuat node baru dengan nilai yang diberikan. Jika kosong (‘head == nullptr’) node baru akan menjadi head. Jika tidak, akan menelusuri list hingga akhir untuk menambahkan node baru di akhir.
6. Fungsi ‘bubbleSort()’ menginsialisasi ‘swapped’ yang diatur ke ‘false’ di

awal setiap iterasi untuk menandakan bahwa belum ada pertukaran yang dilakukan. Dan ‘Node* current = head;’ digunakan untuk menelusuri list dari node head.

- Looping: ‘while (current->next != nullptr)’ selama ‘current’ tidak menunjuk ke node terakhir, maka perbandingan antara data node saat ini dan node selanjutnya.
- ‘if (current->data > current->next->data)’ dimana jika data pada node ‘current’ lebih besar dari data pada node berikutnya, maka terjadi tukar data yang dimana simpan data dari ‘current’ ke dalam variabel sementara ‘temp’. Lalu set data dari ‘current’ dengan data dari node berikutnya. Lalu set data dari node berikutnya dengan nilai yang disimpan dalam ‘temp’. Dan tandai pertukaran, dimana set ‘swapped’ menjadi ‘true’ untuk menunjukkan bahwa telah terjadi pertukaran. Dan ‘current = current->next;’ dimana oindah ke node berikutnya untuk melanjutkan perbandingan.
- Pengulangan, ‘do...while (swapped);’ untuk ulangi proses ini sampai tidak ada pertukaran yang dilakukan, yang berarti list sudah terurut.

7. Fungsi ‘display()’, yang digunakan untuk menampilkan semua elemen dalam linked list. Dan menggunakan pointer ‘temp’ untuk menelusuri list dari head hingga akhir. Lalu selama ‘temp’ tidak null, maka tampilan adat dari setiap node, diikuti dengan spasi. Setelah semua elemen ditampilkan, tambahkan baris baru untuk kejelasan output.

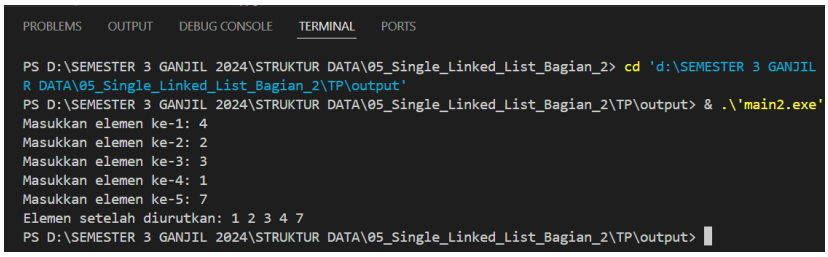
Inputan file main.cpp:

TP > main2.cpp > main()

```cpp
#include "SOAL02.cpp"

int main() {
    LinkedList list;

    // Minta pengguna memasukkan 5 elemen
    for (int i = 0; i < 5; i++) {
        int value;
        cout << "Masukkan elemen ke-" << (i + 1) << ": ";
        cin >> value;
        list.insert(value);
    }

    // Mengurutkan list
    list.bubbleSort();

    // Menampilkan elemen-elemen list setelah diurutkan
    cout << "Elemen setelah diurutkan: ";
    list.display();

    return 0;
}
```

Pada program diatas, fungsi ‘main()’ yaitu membuat objek ‘LinkedList’ bernama ‘list’ dan menggunakan loop untuk memeinta pengguna memasukkan 5 elemen ke dalam list menggunakan fungsi ‘insert()’. Lalau untuk memanggil fungsi ‘bubbleSort()’ untuk mengurutkan elemen dalam list. Dan setelah diurutkan, memanggil fungsi ‘display()’ untuk menampilkan elemen-elemen list yang telah diururtkan.

Output:



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
```

Jawab:

Inputan SOAL03.cpp:

```cpp
TP > SOAL03.cpp > ...
#include <iostream>
using namespace std;

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

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    // Fungsi untuk menambahkan elemen secara terurut
    void insertSorted(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;

        // Jika list kosong atau elemen baru lebih kecil dari head
        if (head == nullptr || head->data >= newNode->data) {
            newNode->next = head;
            head = newNode;
        } else {
            Node* current = head;
            // Temukan posisi untuk elemen baru
            while (current->next != nullptr && current->next->data < newNode->data) {
                current = current->next;
            }
            newNode->next = current->next;
            current->next = newNode;
        }
    }

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

Pada program diatas, terdapat:

1. Header dan namespace, dimana ‘#include <iostream> untuk mengimpor library untuk input dan output. Lalu ‘using namespace std;’ menggunakan namespace standar untuk menghindari oenulisan ‘std::’ didepan setiap penggunaan fungsi dari library standar.
2. Struktur node, terdapat ‘int data’ untuk menyimpan data integer dan ‘Node* head’ untuk pointer yang menunjuk ke node berikutnya.
3. Kelas LinkedList, terdapat ‘class LinkedList’ yang mengelola operasi pada linked list dan ‘Node* head’ untuk pointer yang menunjuk ke node pertama (head) dari linked list.
4. Konstruktor, terdapat ‘LinkedList()’ untuk menginsialisasikan objek ‘LinkedList’ dengan ‘head’ yang diatur ke ‘nullptr’ yang menunjukkan bahwa list kosong

5. Fungsi ‘insert(int value)’, dimana jika list kosong, menggunakan pointer ‘temp’ untuk menelusuri list hingga mencapai node terakhir. Setelah mencapai node terakhir, node baru ditambahkan di akhir list dengan mengatur ‘temp->next’ ke node baru.
6. Fungsi ‘insertSorted(int value)’, dimana ini terdapat ‘Node* newNode = new Node();’ untuk membuat node baru. Lalu ‘newNode->data = value;’ untuk menetapkan data untuk node baru. Dan ‘newMode->next = nullptr;’ untuk menandakan bahwa node baru tidak memiliki next.
7. Fungsi ‘display()’ yang digunakan untuk menampilkan semua elemen dalam linked list. Dan menggunakan pointer ‘temp’ untuk menelusuri list dari head hingga akhir. Lalu selama ‘temp’ tidak null, maka tampilan adat dari setiap node, diikuti dengan spasi. Setelah semua elemen ditampilkan, tambahkan baris baru untuk kejelasan output.

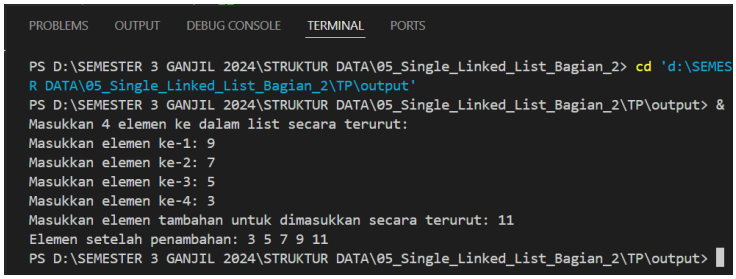
Inputan main.cpp:

TP > main3.cpp > main()

```cpp
#include "SOAL03.cpp"

int main() {
    LinkedList list;

    // Minta pengguna memasukkan 4 elemen
    cout << "Masukkan 4 elemen ke dalam list secara terurut:" << endl;
    for (int i = 0; i < 4; i++) {
        int value;
        cout << "Masukkan elemen ke-" << (i + 1) << ": ";
        cin >> value;
        list.insertSorted(value);
    }

    // Minta pengguna memasukkan elemen tambahan
    int newValue;
    cout << "Masukkan elemen tambahan untuk dimasukkan secara terurut: ";
    cin >> newValue;
    list.insertSorted(newValue);

    // Menampilkan elemen setelah penambahan
    cout << "Elemen setelah penambahan: ";
    list.display();

    return 0;
}
```

Pada program diatas, fungsi ‘main()’ untuk Membuat objek ‘LinkedList’ bernama list. Lalu Menggunakan loop untuk meminta pengguna memasukkan 4 elemen ke dalam list menggunakan fungsi ‘insert()’. Setelah itu, meminta pengguna untuk memasukkan elemen tambahan yang ingin disisipkan. Lalu Memanggil fungsi ‘insertSorted(newValue)’ untuk menambahkan elemen baru ke dalam list secara terurut. Setelah penambahan, memanggil fungsi ‘display()’ untuk menampilkan elemen-elemen list yang telah diperbarui.

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\05_Single_Linked_List_Bagian_2> cd 'd:\SEMES
R DATA\05_Single_Linked_List_Bagian_2\TP\output'
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\05_Single_Linked_List_Bagian_2\TP\output> &
Masukkan 4 elemen ke dalam list secara terurut:
Masukkan elemen ke-1: 9
Masukkan elemen ke-2: 7
Masukkan elemen ke-3: 5
Masukkan elemen ke-4: 3
Masukkan elemen tambahan untuk dimasukkan secara terurut: 11
Elemen setelah penambahan: 3 5 7 9 11
PS D:\SEMESTER 3 GANJIL 2024\STRUKTUR DATA\05_Single_Linked_List_Bagian_2\TP\output>