

**LAPORAN PRAKTIKUM
MODUL 05
SINGLE LINKED LIST BAGIAN KEDUA**



Nama :

Tegar Kang Ageng Gilang (2311104018)

Dosen :

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

**PROGRAM STUDI S1 REKAYASA PERANGKAT
LUNAK**

**FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024**

TP (Tugas Pendahuluan)

1. Buatlah sebuah program yang memungkinkan pengguna memasukkan 6 elemen integer ke dalam sebuah list. Implementasikan fungsi `searchElement` untuk memeriksa apakah nilai tertentu terdapat dalam list tersebut."

Program :

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

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

// Inisialisasi list
struct List {
    Element* head;
};

void createList_23111804018(List &L) {
    L.head = NULL;
}

Element* alokasi_23111804018(int x) {
    Element* newElemen = new Element;
    newElemen->data = x;
    newElemen->next = NULL;
    return newElemen;
}

void insertFirst_23111804018(List &L, Element* P) {
    P->next = L.head;
    L.head = P;
}

void printInfo_23111804018(List L) {
    Element* current = L.head;
    int position = 1;
    while (current != NULL) {
        cout << "Elemen ke-" << position << ": " << current->data << endl;
        current = current->next;
        position++;
    }
}

void searchElement_23111804018(List L, int i) {
    Element* current = L.head;
    int position = 1;
    bool found = false;

    // Loop untuk mencari elemen dengan nilai i
    while (current != NULL) {
        if (current->data == i) {
            cout << "Nilai " << i << " ditemukan pada alamat: " << current << " dengan posisi urutan ke-" << position << endl;
            found = true;
            break;
        }
        current = current->next;
        position++;
    }

    // Jika elemen tidak ditemukan
    if (!found) {
        cout << "Elemen dengan nilai " << i << " tidak ada dalam list." << endl;
    }
}

int main() {
    List L;
    createList_23111804018(L);

    // Meminta pengguna untuk memasukkan 6 elemen
    cout << "Masukkan 6 elemen integer ke dalam list:" << endl;
    for (int j = 1; j <= 6; j++) {
        int value;
        cout << "Masukkan elemen ke-" << j + 1 << ": ";
        cin >> value;
        Element* newElemen = alokasi_23111804018(value);
        insertFirst_23111804018(L, newElemen);
    }

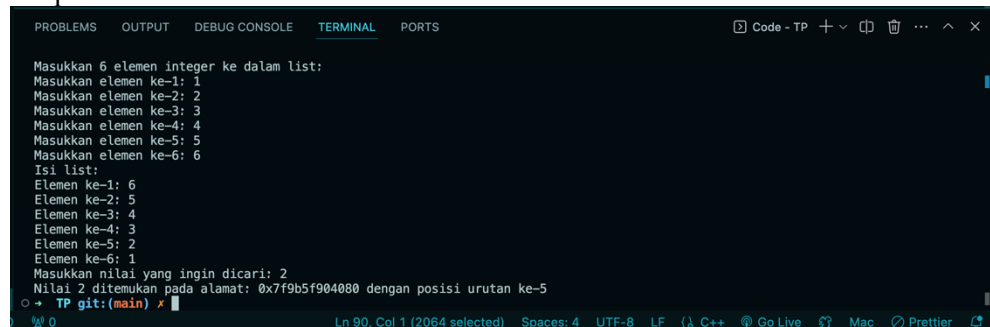
    // Menampilkan isi list
    cout << "Isi list:" << endl;
    printInfo_23111804018(L);

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

    // Mencari elemen dalam list
    searchElement_23111804018(L, cari);

    return 0;
}
```

Output :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan 6 elemen integer ke dalam list:
Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 4
Masukkan elemen ke-5: 5
Masukkan elemen ke-6: 6
Isi list:
Elemen ke-1: 6
Elemen ke-2: 5
Elemen ke-3: 4
Elemen ke-4: 3
Elemen ke-5: 2
Elemen ke-6: 1
Masukkan nilai yang ingin dicari: 2
Nilai 2 ditemukan pada alamat: 0x7f9b5f904080 dengan posisi urutan ke-5
TP git:(main) x
```

Penjelasan :

Kode ini merupakan implementasi linked list dengan fungsi `searchElement` untuk mencari nilai tertentu, lengkap dengan posisi dan alamat memorinya. Program juga menyediakan metode `tambahElemen` untuk menambahkan elemen di akhir list dan `tampilkanList` untuk menampilkan semua elemen dalam list.

Dalam program meminta pengguna untuk memasukkan 6 elemen integer ke dalam linked list, lalu menampilkan seluruh elemen yang ada. Pengguna kemudian dapat mencari nilai tertentu dalam list. sinya.

2. Buatlah program yang memungkinkan pengguna memasukkan 5 elemen integer ke dalam list. Implementasikan prosedur `bubbleSortList` untuk mengurutkan elemen-elemen dalam list dari nilai terkecil hingga terbesar.

Program :

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

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

// Inisialisasi list
struct List {
    Elemen* head;
};

void createList2311104018(List &L) {
    L.head = NULL;
}

Elemen* alokas2311104018(int x) {
    Elemen* newElemen = new Elemen;
    newElemen->data = x;
    newElemen->next = NULL;
    return newElemen;
}

void insertFirst2311104018(List &L, Elemen* P) {
    P->next = L.head;
    L.head = P;
}

void printInfo2311104018(List L) {
    Elemen* current = L.head;
    while (current != NULL) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

// Fungsi untuk mengurutkan elemen-elemen dalam list menggunakan Bubble Sort
void bubbleSortList2311104018(List &L) {
    if (L.head == NULL) return; // Jika list kosong, tidak perlu diurutkan

    bool swapped;
    do {
        swapped = false;
        Elemen* current = L.head;

        // Melakukan iterasi pada list
        while (current->next != NULL) {
            if (current->data > current->next->data) {
                // Pertukaran nilai data (bukan pointer)
                int temp = current->data;
                current->data = current->next->data;
                current->next->data = temp;

                swapped = true; // Menandakan bahwa ada pertukaran
            }
            current = current->next;
        } while (swapped); // Ulangi hingga tidak ada pertukaran
    }

}

int main() {
    List L;
    createList2311104018(L);

    // Meminta pengguna untuk memasukkan 5 elemen integer
    cout << "Masukkan 5 elemen integer ke dalam list:" << endl;
    for (int j = 0; j < 5; j++) {
        int value;
        cout << "Masukkan elemen ke-" << j + 1 << " : ";
        cin >> value;
        Elemen* newElemen = alokas2311104018(value);
        insertFirst2311104018(L, newElemen);
    }

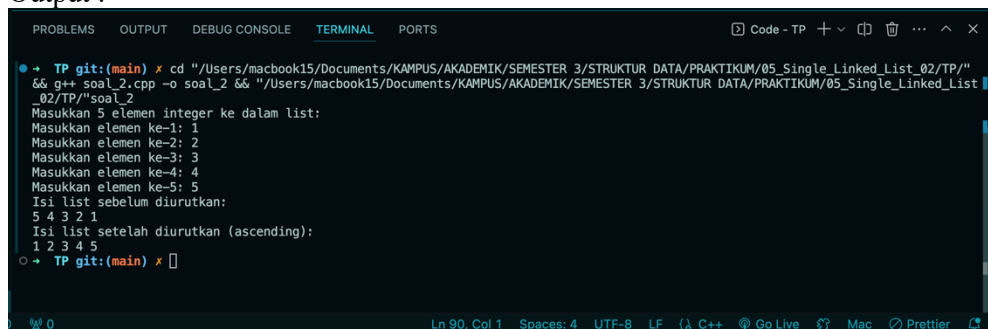
    // Menampilkan isi list sebelum diurutkan
    cout << "Isi list sebelum diurutkan:" << endl;
    printInfo2311104018(L);

    // Mengurutkan list menggunakan Bubble Sort
    bubbleSortList2311104018(L);

    // Menampilkan isi list setelah diurutkan
    cout << "Isi list setelah diurutkan (ascending):" << endl;
    printInfo2311104018(L);

    return 0;
}
```

Output :



```
TP git:(main) x cd "/Users/macbook15/Documents/KAMPUS/AKADEMIK/SEMESTER 3/STRUKTUR DATA/PRAKTIKUM/05_Single_Linked_List_02/TP/"
&& g++ soal_2.cpp -o soal_2 && "/Users/macbook15/Documents/KAMPUS/AKADEMIK/SEMESTER 3/STRUKTUR DATA/PRAKTIKUM/05_Single_Linked_List_02/TP/"soal_2
Masukkan 5 elemen integer ke dalam list:
Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 4
Masukkan elemen ke-5: 5
Isi list sebelum diurutkan:
5 4 3 2 1
Isi list setelah diurutkan (ascending):
1 2 3 4 5
TP git:(main) x
```

Penjelasan :

Kode ini merupakan implementasi linked list dengan prosedur `bubbleSort` untuk mengurutkan elemen-elemen dalam list secara ascending menggunakan algoritma bubble sort. Program juga memiliki metode `tambahElemen` untuk menambahkan elemen baru di akhir list dan `tampilkanList` untuk menampilkan seluruh elemen.

Pada program meminta pengguna memasukkan 5 elemen integer yang disimpan dalam linked list, kemudian menampilkan isi list sebelum dan sesudah pengurutan.

3. Buat program yang memungkinkan pengguna memasukkan 4 elemen integer ke dalam list secara manual. Setelah itu, minta pengguna memasukkan satu elemen tambahan yang akan ditempatkan di posisi yang sesuai agar list tetap terurut.

Program :

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

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

struct List {
    Elemen* head;
};

void createList2311104018(List& L) {
    L.head = NULL;
}

Elemen* alokas2311104018(int x) {
    Elemen* newElemen = new Elemen;
    newElemen->data = x;
    newElemen->next = NULL;
    return newElemen;
}

void insertFirst2311104018(List& L, Elemen* P) {
    P->next = L.head;
    L.head = P;
}

void printInfo2311104018(List L) {
    Elemen* current = L.head;
    while (current != NULL) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

// Prosedur untuk memasukkan elemen secara terurut
void insertSorted2311104018(List& L, Elemen* P) {
    Elemen* Q = L.head;
    Elemen* Prev = NULL;

    // Cari posisi di mana elemen harus dimasukkan
    while (Q != NULL && Q->data < P->data) {
        Prev = Q;
        Q = Q->next;
    }

    // Jika elemen harus dimasukkan di posisi head
    if (Prev == NULL) {
        P->next = L.head;
        L.head = P;
    }
    // Jika elemen harus dimasukkan di antara dua elemen atau di akhir list
    else {
        Prev->next = P;
        P->next = Q;
    }
}

int main() {
    List L;
    createList2311104018(L);

    // Minta pengguna untuk memasukkan 4 elemen integer
    cout << "Masukkan 4 elemen integer ke dalam list secara terurut:" << endl;
    for (int i = 0; i < 4; i++) {
        int value;
        cout << "Masukkan elemen ke-" << i + 1 << ": ";
        cin >> value;
        Elemen* newElemen = alokas2311104018(value);
        insertSorted2311104018(L, newElemen);
    }

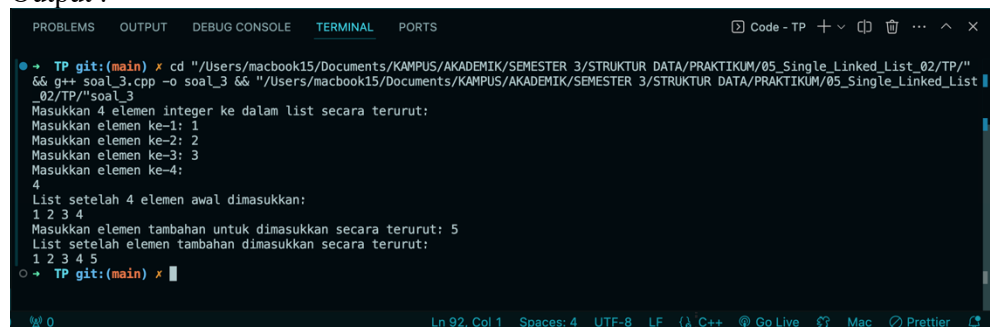
    // Menampilkan isi list setelah elemen awal dimasukkan
    cout << "List setelah 4 elemen awal dimasukkan:" << endl;
    printInfo2311104018(L);

    // Minta pengguna memasukkan elemen tambahan
    cout << "Masukkan elemen tambahan untuk dimasukkan secara terurut: ";
    int additionalValue;
    cin >> additionalValue;
    Elemen* additionalElemen = alokas2311104018(additionalValue);
    insertSorted2311104018(L, additionalElemen);

    // Menampilkan isi list setelah elemen tambahan dimasukkan
    cout << "List setelah elemen tambahan dimasukkan secara terurut:" << endl;
    printInfo2311104018(L);

    return 0;
}
```

Output :



```
TP git:(main) x cd "/Users/macbook15/Documents/KAMPUS/AKADEMIK/SEMESTER 3/STRUKTUR DATA/PRAKTIKUM/05_Single_Linked_List_02/TP/"
&& g++ soal_3.cpp -o soal_3 && "/Users/macbook15/Documents/KAMPUS/AKADEMIK/SEMESTER 3/STRUKTUR DATA/PRAKTIKUM/05_Single_Linked_List
02/TP/"soal_3
Masukkan 4 elemen integer ke dalam list secara terurut:
Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 4
List setelah 4 elemen awal dimasukkan:
1 2 3 4
Masukkan elemen tambahan untuk dimasukkan secara terurut: 5
List setelah elemen tambahan dimasukkan secara terurut:
1 2 3 4 5
TP git:(main) x
```

Penjelasan :

Kode ini merupakan implementasi linked list dengan prosedur `insertSorted` untuk menambahkan elemen baru ke dalam list yang sudah terurut. Metode `insertSorted` berfungsi menyisipkan elemen baru pada posisi yang tepat agar list tetap dalam urutan ascending. Jika `head` kosong atau elemen baru lebih kecil atau sama dengan `head`, elemen tersebut akan ditempatkan di depan list. Jika tidak, program akan mencari posisi yang sesuai di list.

Pada program meminta pengguna memasukkan 4 elemen integer yang langsung diurutkan dalam list. Setelah itu, pengguna dapat menambahkan elemen baru, dan list akan tetap terurut saat hasilnya ditampilkan.

