LAPORAN PRAKTIKUM

MODUL 5

ABSTRCSINGLE LINKED LIST (BAGIAN KEDUA)

Nama :

Novita Syahwa Tri Hapsari (2311104007)

Dosen : Yudha Islami

Sulistya,S.Kom,M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

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

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.

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

struct SearchResult_2311104007 {
    int* address;
    int position;
    bool found;
};

SearchResult_2311104007 searchElement_2311104007(int L[], int
size, int target) {
    SearchResult_2311104007 result;
    result.found = false;
    int* current = &L[0];
    int position = 1;

    while (position <= size) {
        if (*current == target) {
            result.address = current;
            result.position = position;
            result.found = true;
            return result;
        }
        if (position < size) {
            current = &L[position];
        }
        position++;
    }

    return result;
}
void inputData_2311104007(int arr[], int size) {
    cout << "Masukkan " << size << " angka integer:\n";
    for (int i = 0; i < size; i++) {
        while (true) {
            cout << "Angka ke-" << (i + 1) << ": ";
            if (cin >> arr[i]) {
                break;
            } else {
                cout << "Mohon masukkan angka integer yang
valid!\n";
                cin.clear();
                cin.ignore(10000, '\n');
            }
        }
    }
}
int inputSearchValue_2311104007() {
    int searchValue;
    while (true) {
        cout << "\nMasukkan nilai yang ingin dicari: ";
        if (cin >> searchValue) {
            return searchValue;
        } else {
            cout << "Mohon masukkan angka integer yang
valid!\n";
            cin.clear();
            cin.ignore(10000, '\n');
        }
    }
}
void displayResult_2311104007(SearchResult_2311104007 result,
int searchValue) {
    if (result.found) {
        cout << "\nNilai " << searchValue << " ditemukan:" <<
endl;
        cout << "Alamat: " << result.address << endl;
        cout << "Posisi: urutan ke-" << result.position <<
endl;
    } else {
        cout << "\nNilai " << searchValue << " tidak
ditemukan dalam array!" << endl;
    }
}
int main() {
    const int SIZE = 6;
    int numbers[SIZE];
    inputData_2311104007(numbers, SIZE);
        int searchValue = inputSearchValue_2311104007();

    SearchResult_2311104007 result =
searchElement_2311104007(numbers, SIZE, searchValue);
        displayResult_2311104007(result, searchValue);
    return 0;
}
```

```
Masukkan 6 angka integer:
Angka ke-1: 2
Angka ke-2: 3
Angka ke-3: 4
Angka ke-4: 5
Angka ke-5: 6
Angka ke-6: 1

Masukkan nilai yang ingin dicari: 6

Nilai 6 ditemukan:
Alamat: 0x5ffe90
Posisi: urutan ke-5
```

Penjelasan:

Pada code tersebut saya sudah menyertakan nim di setiap fungsinya. Program ini mencari angka dalam array dan menunjukkan alamat dan posisi memori jika ditemukan. Pertama, fungsi inputData_2311104007 meminta pengguna memasukkan enam angka untuk mengisi array. Kemudian, fungsi inputSearchValue_2311104007 memasukkan setiap elemen dalam array, dan jika angka ditemukan, alamat dan posisinya disimpan. Parameter displayResult_2311104007 digunakan untuk menampilkan hasil; ini mencetak posisi dan alamat jika ada angka, dan mencetak pesan "tidak ditemukan" jika tidak ada.

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.

2. **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).

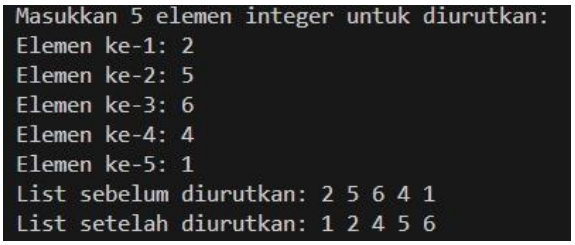
TP > soal2.cpp > main()

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

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

void append_2311104007(Node*& head, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
void bubbleSort_2311104007(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);
}
void printList_2311104007(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
int main() {
    Node* head = nullptr;
    int value;
    cout << "Masukkan 5 elemen integer untuk diurutkan:\n";
    for (int i = 0; i < 5; i++) {
        cout << "Elemen ke-" << i + 1 << ": ";
        cin >> value;
        append_2311104007(head, value);
    }
    cout << "List sebelum diurutkan: ";
    printList_2311104007(head);
    bubbleSort_2311104007(head);
    cout << "List setelah diurutkan: ";
    printList_2311104007(head);
    return 0;
}
```

Penjelasan:

Program ini membuat daftar berantai (link list) dan menggunakan algoritma bubble short untuk mengurutkan elemennya. Pertama, elemen baru ditambahkan ke akhir daftar melalui fungsi append_2311104007. Pengguna diminta untuk mengetik lima angka untuk disimpan dalam daftar. Elemen daftar sebelum dan sesudah pengurutan kemudian dicetak dengan menggunakan fungsi printList_2311104007. Fungsi bubbleSort_2311104007 melakukan pengurutan dengan membandingkan setiap pasangan elemen berurutan dan mengubah posisi mereka jika elemen pertama lebih besar dari elemen kedua. Sampai semua elemen tersusun dari kecil ke besar.

Hasil running



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

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

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

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

void insertSorted_2311104007(Node*& head, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = nullptr;

    if (head == nullptr || head->data >= newData) {
        newNode->next = head;
        head = newNode;
    } else {
        Node* current = head;

        while (current->next != nullptr && current->next->data < newData) {
            current = current->next;
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}
void printList_2311104007(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
int main() {
    Node* head = nullptr;
    int value;
    cout << "Masukkan 4 elemen integer:\n";
    for (int i = 0; i < 4; i++) {
        cout << "Elemen ke-" << i + 1 << ": ";
        cin >> value;
        insertSorted_2311104007(head, value);
    }
    cout << "\nMasukkan elemen tambahan: ";
    cin >> value;
    insertSorted_2311104007(head, value);
    cout << "List setelah elemen tambahan dimasukkan: ";
    printList_2311104007(head);
    return 0;
}
```

Hasil running

```
Masukkan 4 elemen integer:
Elemen ke-1: 2
Elemen ke-2: 1
Elemen ke-3: 6
Elemen ke-4: 4

Masukkan elemen tambahan: 7
List setelah elemen tambahan dimasukkan: 1 2 4 6 7
```

Penjelasan:

Dengan menggunakan fungsi "insertSorted_2311104007", elemen baru ditambahkan ke daftar dan ditempatkan pada posisi yang tepat sehingga daftar tetap terurut. Jika elemen yang ditambahkan lebih kecil atau sama dengan elemen pertama, elemen tersebut dianggap sebagai kepala baru. Jika tidak, fungsi akan mencari posisi yang tepat dan menyisipkan elemen di antara dua elemen lain. Untuk membuat daftar awal, program meminta pengguna memasukkan empat angka, lalu memasukkan satu elemen tambahan. Setelah elemen dimasukkan dan diurutkan, fungsi "printList_2311104007" digunakan untuk menampilkan daftar.