PRAKTIKUM STRUKTUR DATA

TUGAS PENDAHULUAN 5

Nama :

Aulia Ahmad Ghaus Adzam (2311104028)

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

B. Soal Tugas Pendahuluan

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.

SOAL_01.cpp

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

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

Node* createNode(int data) {
   Node* newNode = new Node();
   newNode ->data = data;
   newNode ->next = nullptr;
   return newNode;
}

void insertNode(Node*& head, int data) {
   Node* newNode = createNode(data);
   if (head == nullptr) {
      head = newNode;
   } else {
      Node* temp = head;
      while (temp->next != nullptr) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}

void searchElement_2311104028(Node* head, int target)
{
   Node* current = head;
   int position = 1;
   bool found = false;

   while (current != nullptr) {
      if (current->data == target) {
         cout << "Element " << target << " Ditemukan
di Posisi " << position << " Dengan alamat " <<
current << endl;
         found = true;
         break;
      }

      current = current->next; position++;
   }
   if (!found) {
      cout << "Elemen " << target << " Tidak
ditemukan di dalam list." << endl;
   }
}

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

   cout << "Masukan 6 Elemen untuk List: " << endl;
   for (int i = 0; i < 6; i++) {
      cin >> element;
      insertNode(head, element);
   }

   cout << "masukan elemen yang dicari: ";
   cin >> element ;
   searchElement_2311104028(head, element);

   return 0;
}
```

Penjelasan:
Memasukan Node ke dalam List memakai fungsi inserNode, lalu pencarian elemen dengan linear search dengan fungsi searchElement_2311104028 Sesuai NIM saya maka akan dilakukan pencarian dari head hingga akhir list sampai ditemukan jika tidak maka akan memberi pesan kepada pengguna jika ditemukan pesan yang akan disampaikan adalah posisi elemen tersebut dan alamt dari variable nya.

Output:

```
Masukan 6 Elemen untuk List:
1
3
2
5
6
7
masukan elemen yang dicari: 5
Element 5 Ditemukan di Posisi 4 Dengan alamat 0xd1a80
```

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: 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:
o Bandingkan 4 dan 2: 4 > 2, lakukan penukaran, 2 – 4 – 3 – 1
o Bandingkan 4 dan 3: 4 > 3, lakukan penukaran, 2 – 3 – 4 – 1
o Bandingkan 4 dan 1: 4 > 1, lakukan penukaran, 2 – 3 – 1 – 4
o Kondisi list di akhir iterasi: 2 – 3 – 1 – 4 -

Iterasi kedua:
o Bandingkan 2 dan 3: 2 < 3, tidak terjadi penukaran
o Bandingkan 3 dan 1: 3 > 1, lakukan penukaran, 2 – 1 – 3 – 4
o Bandingkan 3 dan 4: 3 < 4, tidak terjadi penukaran
o Kondisi list di akhir iterasi: 2 – 1 – 3 – 4 -

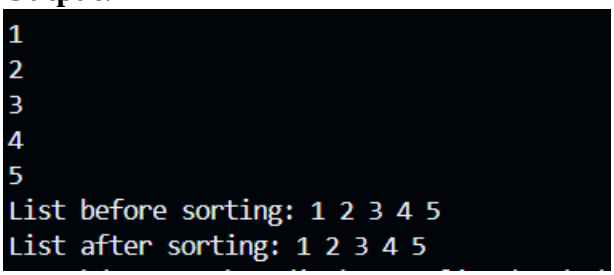
Iterasi ketiga:
o Bandingkan 2 dan 1: 2 > 1, lakukan penukaran, 1 – 2 – 3 – 4
o Bandingkan 2 dan 3: 2 < 3, tidak terjadi penukaran
o Bandingkan 3 dan 4 : 3 < 4, tidak terjadi penukaran
o Kondisi list di akhir iterasi: 1 – 2 – 3 – 4.

SOAL_02.cpp

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

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

Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

void insertNode(Node*& head, int data) {
    Node* newNode = createNode(data);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void bubbleSortList_2311104028(Node*& head) {
    if (head == nullptr) return;

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

    do {
        swapped = false;
        current = head;

        while (current->next != lastSorted) {
            if (current->data > current->next->data)
{
                swap(current->data, current->next-
>data);
                swapped = true;
            }
            current = current->next;
        }
        lastSorted = current;
    } while (swapped);
}

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

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

    cout << "Enter 5 elements for the list: " <<
endl;
    for (int i = 0; i < 5; i++) {
        cin >> element;
        insertNode(head, element);
    }

    cout << "List before sorting: ";
    displayList(head);

    bubbleSortList_2311104028(head);

    cout << "List after sorting: ";
    displayList(head);

    return 0;
}
```

Penjelasan
Node struktur data menyimpan data untuk menyimpan nilai (data) dan pointer ke node berikutnya (next), createNode untuk membuat node baru dengan nilai tertentu, insertNode nambahkan node baru di akhir linked list, bubblesortList 2311104028 untuk menerapkan bubble sorting list

Output:



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.

SOAL_03.cpp

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

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

Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

void insertNodeSorted_2311104028(Node*& head, int data) {
    Node* newNode = createNode(data);
    if (head == nullptr || head->data >= data) {
        newNode->next = head;
        head = newNode;
        return;
    }

    Node* current = head;
    while (current->next != nullptr && current->next->data < data) {
        current = current->next;
    }
    newNode->next = current->next;
    current->next = newNode;
}

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

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

    cout << "Enter 4 elements for the sorted list: " << endl;
    for (int i = 0; i < 4; i++) {
        cin >> element;
        insertNodeSorted_2311104028(head, element);
    }

    cout << "Enter an element to insert into the sorted list: ";
    cin >> element;
    insertNodeSorted_2311104028(head, element);

    cout << "List after insertion: ";
    displayList(head);

    return 0;
}
```

Penjelasan
Node Struktur data berisi nilai dan pointer ke node berikutnya dalam linked list, createNode Mmebuat node baru, inserNodeSorted_2311104028 agar elemen baru di posisi yang tepat terurut, display List menampilkan elemen-elemen list, main memasukan 4 elemen awal yang terurut lalu menabahkan elemeb baru secara terurut.

Output:

```
Enter 4 elements for the sorted list:
10
20
30
40
Enter an element to insert into the sorted list: 100
List after insertion: 10 20 30 40 100
```

C. Latihan Unguided

Modul 5 Single Linked List

Buatlah sebuah program di C++ yang mengimplementasikan Single Linked List untuk menyimpan data mahasiswa. Setiap node dalam linked list menyimpan NIM (Nomor Induk Mahasiswa) dan Nama mahasiswa. Program tersebut harus memiliki fungsi untuk menambahkan data mahasiswa ke dalam linked list dan juga fungsi untuk mencari mahasiswa berdasarkan NIM.

Spesifikasi Program:
1. Buatlah Single Linked List dengan struktur data yang menyimpan:
○ NIM (integer)
○ Nama (string)

2. Implementasikan fungsi untuk:
○ Menambahkan data mahasiswa ke dalam linked list.
○ Mencari mahasiswa berdasarkan NIM.
○ Menampilkan pesan jika mahasiswa ditemukan atau tidak ditemukan dalam list.

3. Jika mahasiswa ditemukan, tampilkan Nama mahasiswa tersebut.

4. Jika tidak ditemukan, ditampikan dengan “Mahasiswa dengan NIM(nim yang dicari) tidak ditemukan.

```cpp
#include <iostream>
#include <string>

using namespace std;

struct Mahasiswa {
    int NIM;
    string nama;
    Mahasiswa* next;
};

void inputMahasiswa(Mahasiswa*& head, int NIM, string nama) {
    Mahasiswa* newNode = new Mahasiswa();
    newNode ->NIM = NIM;
    newNode ->nama = nama;
    newNode ->next = nullptr;

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

Mahasiswa* cariMahasiswa(Mahasiswa* head, int NIM) {
    Mahasiswa* temp = head;
    while (temp != nullptr) {
        if (temp ->NIM == NIM) {
            return temp;
        }
        temp = temp->next;
    }
    return nullptr;
}

void outputMahasiswa(Mahasiswa* head) {
    Mahasiswa* temp = head;
    while (temp != nullptr) {
        cout << "Nama: " << temp->nama << " NIM: " << temp->NIM << endl;
        temp = temp->next;
    }
```

```cpp
int main() {
    Mahasiswa* head = nullptr;
    int NIM, pilihan;
    string nama;

    do {
        cout << "Menu:\n";
        cout << "1. Tambah Mahasiswa\n";
        cout << "2. Cari Mahasiswa Berdasarkan NIM\n";
        cout << "3. Tampilkan Semua Mahasiswa\n";
        cout << "4. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            cout << "Masukan NIM: ";
            cin >> NIM;
            cin.ignore();
            cout << "Masukan Nama: ";
            getline(cin, nama);
            inputMahasiswa(head, NIM, nama);
            break;

        case 2:
            cout << "Masukan NIM Mahasiwa Yang Ingin dicari: ";
            cin >> NIM;
            Mahasiswa* hasil;
            hasil = cariMahasiswa(head, NIM);
            if (hasil != nullptr) {
                cout << "Mahasiswa Dengan NIM: " << NIM << hasil->nama << " Ditemukan " << endl;
            } else {
                cout << "Mahasiswa Dengan NIM: " << NIM << " Tidak Ditemukan" << endl;
            }
            break;

        case 3:
            cout << "Data Mahasiswa\n";
            outputMahasiswa(head);
            break;

        case 4:
            cout << "Anda Telah Keluar dari Program" << endl;
            break;

        default:
            cout << "Pilihan Tidak Valid" << endl;
            break;
        }
    } while (pilihan != 4);

    return 0;
}
```

Penjelasan

Struct Mahasiswa untuk menyimpan variable nama, NIM dan pointer nex yang menunjuk ke node yang selanjutnya, inputMahasiwa() program function untuk memasukan data Mahasiswa, cariMahasiswa() program function untuk mencari

Mahasiswa melalui NIM, outputMahasiswa() adalah program function untuk menampilkan seluruh data Mahasiwa yang telalah di save ke dalam.

Output:

```
Menu:
1. Tambah Mahasiswa
2. Cari Mahasiswa Berdasarkan NIM
3. Tampilkan Semua Mahasiswa
4. Keluar
Pilih menu: 1
Masukkan NIM: 12
Masukkan Nama: Aulia Ahmad Ghaus Adzam
```

```
Menu:
1. Tambah Mahasiswa
2. Cari Mahasiswa Berdasarkan NIM
3. Tampilkan Semua Mahasiswa
4. Keluar
Pilih menu: 2
Masukkan NIM yang dicari: 12
Mahasiswa ditemukan: Aulia Ahmad Ghaus Adzam
```

```
Menu:
1. Tambah Mahasiswa
2. Cari Mahasiswa Berdasarkan NIM
3. Tampilkan Semua Mahasiswa
4. Keluar
Pilih menu: 3
Daftar Mahasiswa:
NIM: 12, Nama: Aulia Ahmad Ghaus Adzam
```