LAPORAN PRAKTIKUM
MODUL 5

SINGLE LINKED LIST BAGIAN KEDUA

Disusun Oleh:

Prajna paramitha - 2311104016

SE 07 01

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
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.

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

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

// Struktur untuk linked list
struct List {
    Node* head;
};

// Fungsi membuat list kosong
void createList_2311104016(List &L) {
    L.head = NULL;
}

// Fungsi alokasi node baru
Node* allocate_2311104016(int x) {
    Node* newNode = new Node();
    newNode->data = x;
    newNode->next = NULL;
    return newNode;
}

// Fungsi untuk menambah node di depan
void insertFirst_2311104016(List &L, Node* newNode) {
    if(L.head == NULL) {
        L.head = newNode;
    } else {
        newNode->next = L.head;
        L.head = newNode;
    }
}

// Fungsi untuk mencari elemen
void searchElement_2311104016(List L, int x) {
    Node* current = L.head;
    int position = 1;
    bool found = false;

    // Melakukan perulangan selama i belum ditemukan dan posisi current belum berada pada akhir list
    while(current != NULL && !found) {
        if(current->data == x) {
            found = true;
        } else {
            current = current->next;
            position++;
        }
    }
```

```
    // Jika i ditemukan maka tampilkan alamat dan posisi
    if(found) {
        cout << "\nElemen " << x << " ditemukan pada: " << endl;
        cout << "Alamat: " << current << endl;
        cout << "Posisi urutan ke-" << position << endl;
    }
    // Jika tidak ditemukan maka tampilkan pesan
    else {
        cout << "\nElemen " << x << " tidak ditemukan dalam list" << endl;
    }
}

// Fungsi untuk menampilkan list
void printList_2311104016(List L) {
    cout << "\nIsi List: ";
    Node* current = L.head;
    while(current != NULL) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

int main() {
    List L;
    createList_2311104016(L);

    // Memasukkan 6 elemen ke dalam list
    cout << "Masukkan 6 bilangan integer:\n";
    for(int i = 0; i < 6; i++) {
        int num;
        cout << "Bilangan ke-" << (i+1) << ": ";
        cin >> num;
        Node* newNode = allocate_2311104016(num);
        insertFirst_2311104016(L, newNode);
    }

    // Menampilkan isi list
    printList_2311104016(L);

    // Mencari elemen
    int searchNum;
    cout << "\nMasukkan nilai yang ingin dicari: ";
    cin >> searchNum;
    searchElement_2311104016(L, searchNum);

    return 0;
}
```

```
Masukkan 6 bilangan integer:
Bilangan ke-1: 1
Bilangan ke-2: 2
Bilangan ke-3: 4
Bilangan ke-4: 5
Bilangan ke-5: 6
Bilangan ke-6: 7

Isi List: 7 6 5 4 2 1

Masukkan nilai yang ingin dicari: 1

Elemen 1 ditemukan pada:
Alamat: 0x747170
Posisi urutan ke-6
PS C:\Users\ZBooK\OneDrive\Dokumen\Struktur data>
```

Penjelasan :

- Struct node : struktur untuk node yang berisi data tipenya integer.
- Struct List : struktur untuk linked listnya
- Void createList : fungsi untuk membuat list kosong
- Node* allocate : fungsi untuk alokasi node baru
- Void insertFirst : fungsi untuk menambahkan node didepan
- Void search element : untuk mencari elemen
  Terdapat perulangan while untuk mencari i.
  Dan terdapat if(found) : jika inya ditemukan pada list maka akan ditampilkan pesan. Jika tidak maka else dan akan menampilkan pesan juga
- Void printList : untuk menampilkan isi listnya
- Int main() : dimana programnya itu berjalan dan memanggil fungsi yg sudah diibuat sebelumnya
- For (…) : merupakan perulangan untuk menginputan 6 data dari user, jadi selama i kurang dari 6 makan perulangan akan terus berjalan yang selanjutnya data akan disimpan di num yang nantinya num ini akan menjadi parameter dalam fungsi allocate untuk menambahkan data yang diinputkan user kedalam list.

- Dibawahnya saya panggil printList untuk menampilkan isi list setelah user menginputkan 6 data.
- User memasukkan bilangan yang ingin dicari, lalu inputan disimpan di searchNum yang sebelumnya kita sudah buat. Yang selanjutnya akan dijadikan parameter pada fungsi searchElement untuk mencari data yang diinputkan.

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.

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

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

// Struktur untuk linked list
struct List {
    Node* head;
};

// Fungsi membuat list kosong
void createList_2311104016(List &L) {
    L.head = NULL;
}

// Fungsi alokasi node baru
Node* allocate_2311104016(int x) {
    Node* newNode = new Node();
    newNode->data = x;
    newNode->next = NULL;
    return newNode;
}

// Fungsi untuk menambah node di depan
void insertFirst_2311104016(List &L, Node* newNode) {
    if(L.head == NULL) {
        L.head = newNode;
    } else {
        newNode->next = L.head;
        L.head = newNode;
    }
}

// Fungsi untuk menampilkan list
void printList_2311104016(List L) {
    cout << "Isi List: ";
    Node* current = L.head;
    while(current != NULL) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}
```

```cpp
// Fungsi bubble sort
void bubbleSortList_2311104016(List &L) {
    if(L.head == NULL || L.head->next == NULL)
        return;

    bool swapped;
    Node* current;
    Node* last = NULL;

    do {
        swapped = false;
        current = L.head;

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

int main() {
    List L;
    createList_2311104016(L);

    // Memasukkan 5 elemen ke dalam list
    cout << "Masukkan 5 bilangan integer:\n";
    for(int i = 0; i < 5; i++) {
        int num;
        cout << "Bilangan ke-" << (i+1) << ": ";
        cin >> num;
        Node* newNode = allocate_2311104016(num);
        insertFirst_2311104016(L, newNode);
    }

    cout << "\nSebelum diurutkan:\n";
    printList_2311104016(L);

    // Mengurutkan list
    bubbleSortList_2311104016(L);
```

```cpp
int main() {
    List L;
    createList_2311104016(L);

    // Memasukkan 5 elemen ke dalam list
    cout << "Masukkan 5 bilangan integer:\n";
    for(int i = 0; i < 5; i++) {
        int num;
        cout << "Bilangan ke-" << (i+1) << ": ";
        cin >> num;
        Node* newNode = allocate_2311104016(num);
        insertFirst_2311104016(L, newNode);
    }

    cout << "\nSebelum diurutkan:\n";
    printList_2311104016(L);

    // Mengurutkan list
    bubbleSortList_2311104016(L);

    cout << "\nSetelah diurutkan (ascending):\n";
    printList_2311104016(L);

    return 0;
}
```

```
Masukkan 5 bilangan integer:
Bilangan ke-1: 3
Bilangan ke-2: 4
Bilangan ke-3: 5
Bilangan ke-4: 6
Bilangan ke-5: 4

Sebelum diurutkan:
Isi List: 4 6 5 4 3

Setelah diurutkan (ascending):
Isi List: 3 4 4 5 6
PS C:\Users\ZBooK\OneDrive\Dok
```

Penjelasan :

- Struct Node : struktur untuk nodenya didalam linked list
- Int data : artinya data dlam node tipe datanya int
- Node* next : pointer yg nunjuk ke node berikutnya
- Struct list : struktur untuk linked list
- Node* head : pointer yg menunjuk ke node pertama dalam list
- Void createList : untuk nginialisasi list kosong
- Node* allocate : untuk mengalokasikan node baru, x sebagai parameter yg akan disimpan ke node barunya
- Node* newNode : buat mengalokasi memori baru
- Void insertfirst : untuk menambahan node diawal list
- Didalam insertfirst terdapat if else pengecekan apakah listnya kosong atau tidak, jika kodong maka node baru yang ditambahkan akan menjadi yang pertama
- Void printList : untuk menampilkan semua elemen didalam list. Terdapat perulangan while yang akan berjalan terus hingga akhir listnya.
- Void bubblesort : untuk penukaran angka yang diinputkan, jadi jika list kosong atau hanya memiliki satu elemen maka gaudah diurutkan.
- Bool swapped : untuk mengecek apakah ada pertukaran atau engga
- Node* current : digunkan untuk poibter traversing list
- Lalu ada pengkondisian jika elemen yang saat ini lebih besar dari elemen berikutnya makan akan terjadi pertukaran.(if(current->data > current->next->data))
- Int main() : program utama
- Di dalam intmain akan menerima inputan dari user yang kemudian disimpan di num yang akan menjadi parameter dalam fungsi allocate agar data yang diinputkan user ini dimasukkan kedalam list.
- Lalu menampilkan data sebelum diurutkan dan menampilkan data setelah diurutkan.

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.

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

// Struktur untuk node dalam linked list
struct Node {
    int data;          // Menyimpan nilai
    Node* next;        // Pointer ke node
};

// Struktur untuk linked list
struct List {
    Node* head;        // Pointer ke node
};

// Fungsi untuk membuat list kosong
void createList_2311104016(List &L) {
    L.head = NULL;     // Set head ke NULL
}

// Fungsi untuk mengalokasi node baru
Node* allocate_2311104016(int x) {
    Node* newNode = new Node();    // Alok
    newNode->data = x;             // data
    newNode->next = NULL;
    return newNode;
}

// Fungsi untuk menampilkan isi list
void printList_2311104016(List L) {
    cout << "Isi List: ";
    Node* current = L.head;        // Mula
    while(current != NULL) {       // peru
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}
```

```cpp
// Fungsi untuk menambah elemen secara terurut
void insertSorted_2311104016(List &L, Node* P) {
    // Jika list kosong atau data baru lebih kecil dari head
    if (L.head == NULL || P->data < L.head->data) {
        P->next = L.head;
        L.head = P;                // Update head ke P
        return;
    }

    // Mencari posisi yang tepat untuk menyisipkan node baru
    Node* current = L.head;
    while (current->next != NULL && current->next->data < P->data) {
        current = current->next;
    }

    // Sisipkan node P setelah posisi current
    P->next = current->next;
    current->next = P;
}

int main() {
    List L;
    createList_2311104016(L);      // Inisialisasi list kosong

    // Input 4 elemen pertama
    cout << "Masukkan 4 bilangan integer untuk list awal:\n";
    for(int i = 0; i < 4; i++) {
        int num;
        cout << "Bilangan ke-" << (i+1) << ": ";
        cin >> num;
        // mbuat node baru trus masukin secara terurut
        Node* newNode = allocate_2311104016(num);
        insertSorted_2311104016(L, newNode);

        cout << "List setelah penambahan: ";
        printList_2311104016(L);
    }

    // Input elemen tambahan
    cout << "\nMasukkan bilangan tambahan: ";
    int tambahan;
    cin >> tambahan;
    Node* newNode = allocate_2311104016(tambahan);
    insertSorted_2311104016(L, newNode);

    // Tampilkan hasil akhir
    cout << "\nList setelah penambahan elemen baru:\n";
    printList_2311104016(L);

    return 0;
}
```

```
Masukkan 4 bilangan integer untuk list awal:
Bilangan ke-1: 4
List setelah penambahan: Isi List: 4
Bilangan ke-2: 3
List setelah penambahan: Isi List: 3 4
Bilangan ke-3: 1
List setelah penambahan: Isi List: 1 3 4
Bilangan ke-4: 5
List setelah penambahan: Isi List: 1 3 4 5

Masukkan bilangan tambahan: 2

List setelah penambahan elemen baru:
Isi List: 1 2 3 4 5
PS C:\Users\ZBooK\OneDrive\Dokumen\Struktur data>
```

Penjelasan :

- Struct node : struktur untuk node dalam linked list
- Struct list : struktur untuk linked list
- Void createlist : fungsi untuk membuat list kosong
- Node* allocate : untuk mengalokaso node baru, jadi ada nweNode=>data = x yang atyinya data buat newnode itu diisi dnegan x(inputan)
- Void printlist : untuk menampilkan isi list, terdapat while dimana akan terus berjalan selama dia belum sampai akhir
- Void insertsorted : untuk menambahkan elemen secara terurut, didalamnya ada perbandingan, membandingkan inputan dengan data” yg ada di listnya
- Int main () : program utama
- Inputan user didalam tambahan yg kemudia menjadi parameter di fungsi allocate untuk dimasukkan datanya dan akan dirutkan di fungsi insertsorted.
- Lalu terakhir menampilkan hasil akhir inputan ketika sudah terurut.