Laporan Praktikum

Algoritma dan Struktur Data

Ganjil 2025/2026
Program Studi Teknik Informatika
Institut Teknologi Sumatera

Modul : Single Linked List

Nama : Kenzie Sahasika Tariana

NIM : 124140103

Kelas (Kelas Asal) : RD

Instruksi sederhana :

- Disarankan untuk edit menggunakan Google Docs agar tidak berantakan,
- Silahkan mengganti nama modul baik yang ada pada **cover** maupun **header** sesuai dengan materi praktikum,
- Gunakan text styling seperti Heading 1, Normal Text yang telah terformat, atau text style lainnya untuk menjaga estetika laporan,
- Gunakan Syntax Highlighter untuk merapikan kode yang sudah anda buat ke dalam laporan.

Dirancang oleh Muhammad Fida Raditya
Terinspirasi dari Format Tata Tulis Laporan Embedded

Soal/Pertanyaan:

1. Latihan 1: Sistem Nilai Siswa Dinamis

Objektif: Membuat sistem penyimpanan nilai siswa menggunakan Single Linked List.

Spesifikasi:

Input: Nama siswa dan nilai (dapat ditambah kapan saja)
Output: Tampilkan semua data siswa

Latihan 2: Insert dengan Urutan Tersorting

Objektif: Implementasi insertion yang menjaga list tetap terurut.

Spesifikasi:

Setiap data baru diinsert pada posisi yang tepat. List selalu dalam kondisi terurut ascending.

```
void InsertSorted(List *L, infotype x) {
    if (IsEmpty(*L) || (*L).first->info > x) {
        InsertFirst(L, x);
    } else {
        address temp = (*L).first;

        while (temp->next != NULL && temp->next->info < x) {
            temp = temp->next;
        }

        InsertAfter(temp, x);
    }
}
```

Dasar Teori

Single Linked List

Single Linked List (SLL) adalah salah satu struktur data linier yang terdiri dari sekumpulan node, di mana setiap node menyimpan dua informasi utama:

1. Data :
   nilai atau informasi yang disimpan.

2. Pointer (next):
   alamat atau referensi menuju node berikutnya dalam daftar.

Kegunaan

Linked List digunakan ketika:

- Ukuran data sering berubah-ubah (tidak tetap seperti array),
- Sering melakukan penyisipan (insert) atau penghapusan (delete) data di tengah/awal/akhir dengan efisien.

Operasi Dasar

Ada beberapa operasi dasar pada saat menggunakan single linked list, di antaranya adalah :

1. **Menambah data**
   - Di awal → buat node baru, sambungkan ke head.
   - Di akhir → cari node terakhir, sambungkan ke node baru.

2. **Menghapus data**
   - Temukan node yang berisi nilai tertentu, ubah pointer `next` agar melewati node tersebut.

3. **Menampilkan data**
   - Telusuri dari `head` sampai `NULL`, tampilkan setiap `data`

Berbeda dengan array, Linked List tidak disimpan secara berurutan di memori, karena setiap node "menunjuk" ke node berikutnya dengan menggunakan pointer.
Berikut struktur sederhananya bisa digambarkan seperti ini:
`[Data | Next] → [Data | Next] → [Data | Next] → NULL`

Struktur Node (C++)

Kita dapat membuat struktur node pada kode program dengan cara seperti ini:

```
struct Node {

    int data;
    Node* next;
};
```

Setiap `Node` dapat menyimpan sebuah integer (`data`) dan pointer (`next`) ke node berikutnya.

Operasi Dasar pada Single Linked List

Berikut ini merupakan beberapa operasi dasar yang pada umum digunakan untuk pengopersian single linked list :

- Menambahkan node di awal program:

```
void insertAtBeginning(Node*& head, int value) {
    Node* newNode = new Node;   // Buat node baru
    newNode->data = value;
    newNode->next = head;       // Next menunjuk ke node lama
    head = newNode;             // Head berpindah ke node baru
}
```

- **Menambahkan node di akhir**

```
void insertAtEnd(Node*& head, int value) {
    Node* newNode = new Node;
```

```
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL) {  // Jika list masih kosong
        head = newNode;
        return;
    }

    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;  // Tambahkan di akhir
}
```

- **Menghapus node dengan nilai tertentu**

```
void deleteNode(Node*& head, int value) {
    if (head == NULL) return;

    if (head->data == value) { // Jika data di head yang dihapus
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

    Node* current = head;
    while (current->next != NULL && current->next->data != value) {
        current = current->next;
    }
```

```
    if (current->next == NULL) return; // Tidak ditemukan

    Node* temp = current->next;
    current->next = temp->next;
    delete temp;
}
```

- **Menampilkan isi linked list**

```
void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        std::cout << temp->data << " -> ";
        temp = temp->next;
    }
    std::cout << "NULL\n";
}
```

Source Code

Nomor 1

```
1. #include <iostream>
2. using namespace std;
3.
4. // Membuat struktur node untuk menyimpan data siswa
5. struct Node {
6.     char nama[50];
7.     int nilai;
8.     Node* next;
9. };
10.
11.// Membuat fungsi untuk menyalin nama
```

```
12. void salinNama(char* tujuan, const char* sumber) {
13.     int i = 0;
14.     while (sumber[i] != '\0' && i < 49) {
15.         tujuan[i] = sumber[i];
16.         i++;
17.     }
18.     tujuan[i] = '\0';  // menutup string dengan null-terminator
19. }
20.
21. // Memberikan kelas LinkedList untuk menyimpan data siswa secara
    dinamis
22. class LinkedList {
23. private:
24.     Node* head;
25.
26. public:
27.     LinkedList() : head(NULL) {}
28.
29.     // Membuat fungsi untuk menambahkan data di akhir list
30.     void tambahData(const char* nama, int nilai) {
31.         Node* baru = new Node;
32.         salinNama(baru->nama, nama);
33.         baru->nilai = nilai;
34.         baru->next = NULL;
35.
36.         // jika list masih kosong, maka node baru menjadi head
37.         if (head == NULL) {
38.             head = baru;
39.         }
40.         // jika tidak kosong, maka akan mencari node terakhir dan di
    sambungkan ke sana
41.         else {
42.             Node* temp = head;
43.             while (temp->next != NULL)
44.                 temp = temp->next;
45.             temp->next = baru;
46.         }
47.     }
48.
49.     // Membuat fungsi untuk menampilkan semua data siswa
50.     void tampilkan() {
51.
```

```
52.         if (head == NULL) {// jika list kosong
53.             cout << "Data masih kosong.\n";
54.             return;
55.         }
56.         // Untuk mengeluarkan daftar nilai yang sudah ada
57.         cout << "\n=== Daftar Nilai Siswa ===\n";
58.         Node* temp = head;
59.         while (temp != NULL) {
60.             cout << "Nama : " << temp->nama << "\n";
61.             cout << "Nilai: " << temp->nilai << "\n";
62.             cout << "--------------------\n";
63.             temp = temp->next;          // pindah ke node berikutnya
64.         }
65.     }
66. };
67. 
68. 
69. int main() {
70.     // Melakukn deklarasi untuk beberapa variabel yang akan di gunakan
    dalam program
71.     LinkedList list;
72.     int pilih;
73.     char nama[50];
74.     int nilai;
75. 
76.      // Memberikan output menu utama menggunakan do while, sehingga
    pengguna dapat memilih operasi apa yang ingin dilakukan lebih dari
    sekali
77.     do {
78.         cout << "\n=== Sistem Nilai Siswa Dinamis ===\n";
79.         cout << "1. Tambah Data\n";
80.         cout << "2. Tampilkan Data\n";
81.         cout << "0. Keluar\n";
82.         cout << "Pilih: ";
83.         cin >> pilih;
84.         cin.ignore();
85. 
86.         // Menggunakan switch case untuk melakukan operasi yang sesuai
    dengan pilihan pengguna
87.         switch (pilih) {
88.             case 1:
89.                 cout << "\nMasukkan nama: ";
```

```
90.                 cin.getline(nama, 50);  // baca satu baris nama
   (termasuk spasi)
91.                 cout << "Masukkan nilai: ";
92.                 cin >> nilai;           // baca nilai
93.                 list.tambahData(nama, nilai);  // tambahkan ke list
94.                 break;
95.             case 2:
96.                 list.tampilkan();
97.                 break;
98.             case 0:
99.                 cout << "Keluar dari program.\n";
100.                   break;
101.               default:
102.                   cout << "Pilihan tidak valid.\n";
103.           }
104.       } while (pilih != 0); // ulang sampai user pilih keluar
105.
106.       return 0;
107.   }
```

Nomor 2

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

// Membuat struktur node untuk menyimpan data siswa
struct Node {
    char nama[50];
    int nilai;
    Node* next;
};

// Membuat fungsi untuk menyalin nama
void salinNama(char* tujuan, const char* sumber) {
    int i = 0;
    while (sumber[i] != '\0' && i < 49) {
        tujuan[i] = sumber[i];
        i++;
    }
    tujuan[i] = '\0';  // menutup string dengan null-terminator
}

// Memberikan kelas LinkedList untuk menyimpan data siswa secara dinamis
class LinkedList {
private:
    Node* head;

public:
    LinkedList() : head(NULL) {}

    // Fungsi untuk menyisipkan data baru secara terurut (ascending)
    void tambahData(const char* nama, int nilai) {
        Node* baru = new Node;
        salinNama(baru->nama, nama);
        baru->nilai = nilai;
        baru->next = NULL;

        // Kasus 1: list kosong atau nilai baru lebih kecil dari head
        if (head == NULL || nilai < head->nilai) {
            baru->next = head;
            head = baru;
```

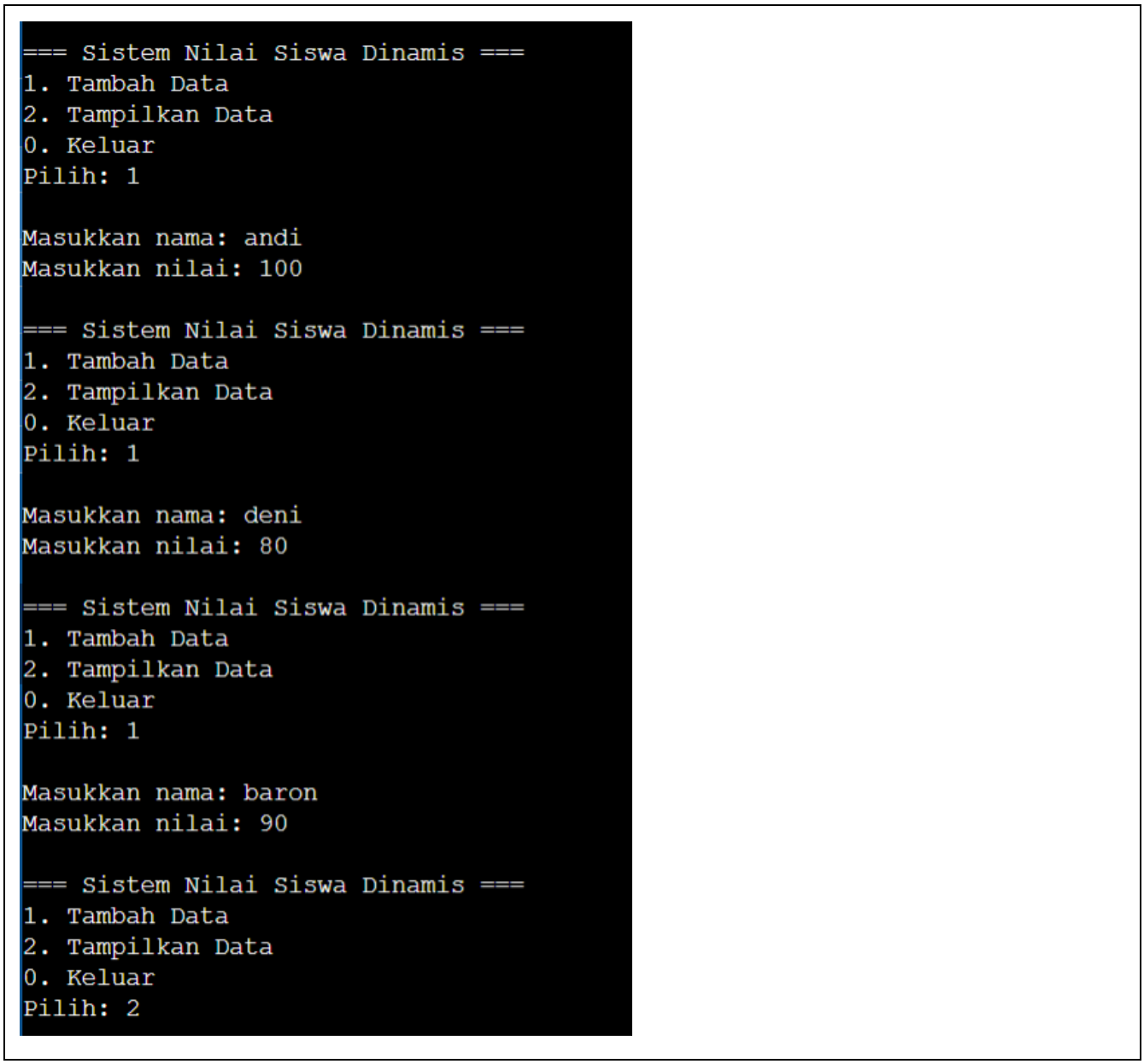
```
        }
        // Kasus 2: cari posisi yang tepat (nilai sebelumnya < nilai baru)
        else {
            Node* temp = head;

            // maju sampai menemukan posisi sebelum tempat penyisipan
            while (temp->next != NULL && temp->next->nilai < nilai)
                temp = temp->next;

            // sisipkan node baru di tengah / akhir
            baru->next = temp->next;
            temp->next = baru;
        }
    }
    // Membuat fungsi untuk menampilkan semua data siswa
    void tampilkan() {

        if (head == NULL) {// jika list kosong
            cout << "Data masih kosong.\n";
            return;
        }
        // Untuk mengeluarkan daftar nilai yang sudah ada dan data yang telah tersusun secara ascending
        cout << "\n=== Daftar Nilai Siswa (Ascending) ===\n";
        Node* temp = head;
        while (temp != NULL) {          // loop semua node
            cout << "Nama : " << temp->nama << "\n";
            cout << "Nilai: " << temp->nilai << "\n";
            cout << "--------------------\n";
            temp = temp->next;
        }
    }
};


int main() {
    // Melakukn deklarasi untuk beberapa variabel yang akan di gunakan dalam program
    LinkedList list;
    int pilih;
    char nama[50];
```

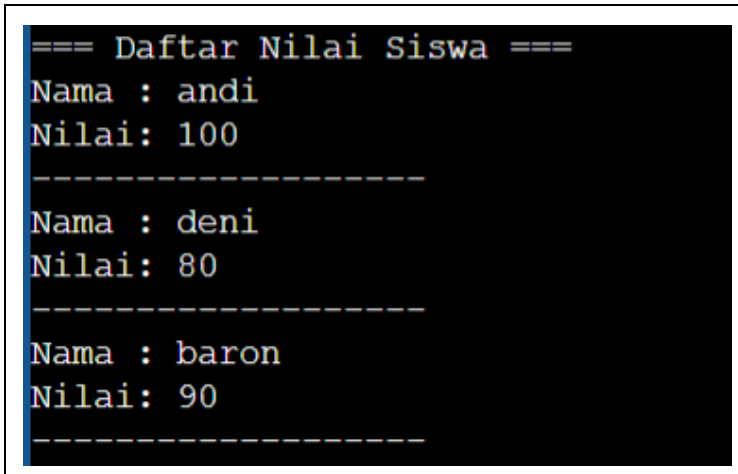
```
    int nilai;

     // Memberikan output menu utama menggunakan do while, sehingga pengguna dapat memilih operasi apa yang ingin dilakukan lebih dari sekali
    do {
        cout << "\n=== Sistem Nilai Siswa Dinamis ===\n";
        cout << "1. Tambah Data\n";
        cout << "2. Tampilkan Data\n";
        cout << "0. Keluar\n";
        cout << "Pilih: ";
        cin >> pilih;
        cin.ignore();

        // Menggunakan switch case untuk melakukan operasi yang sesuai dengan pilihan pengguna
        switch (pilih) {
            case 1:
                cout << "\nMasukkan nama: ";
                cin.getline(nama, 50);  // baca satu baris nama (termasuk spasi)
                cout << "Masukkan nilai: ";
                cin >> nilai;           // baca nilai
                list.tambahData(nama, nilai);  // tambahkan ke list
                break;
            case 2:
                list.tampilkan();
                break;
            case 0:
                cout << "Keluar dari program.\n";
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    } while (pilih != 0); // ulang sampai user pilih keluar

    return 0;
}
```

Dokumentasi Hasil Running

Nomor 1

Pada hasil running untuk kode program dari soal nomor 1 ini penggunaan single linked list dalam membuat sistem penyimpanan list nilai dari beberapa siswa yang diinputkan oleh pengguna secara manual. Sesuai dengan ketentuan soal, pengguna dapat menginputkan data nama siswa dan nilai dari masing-masing siswa secara manual. Lalu data yang telah diinputkan disimpan, yang akan dapat dipanggil kembali untuk ditampilkan oleh penguna.



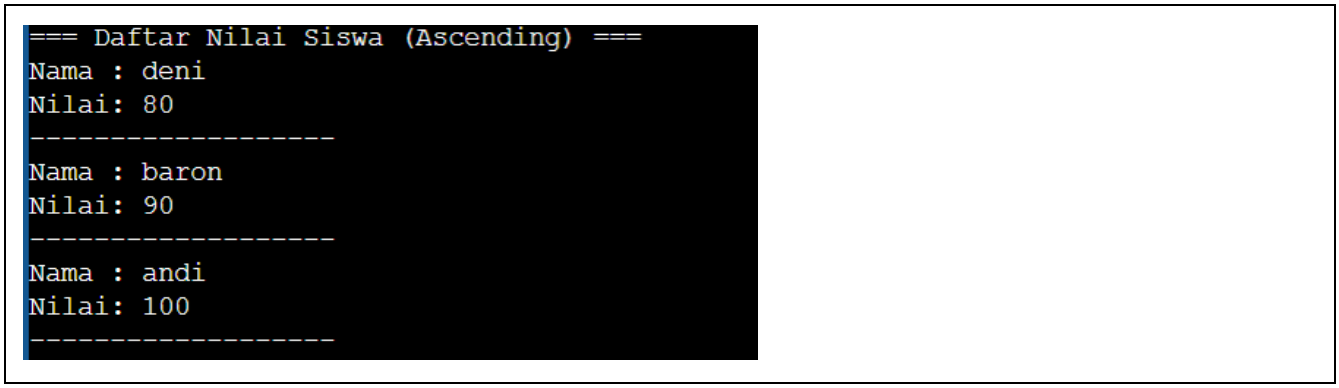


Gambar 01. Sistem penyimpanan Single Linked List

Nomor 2

Kode program pada nomor 2 ini hampir sama dengan nomor sebelumnya, namun ditambahkan beberapa penyesuaian. Pada kode program kali ini ditambahkan fungsi yang berfungsi untuk melakukan pengecekan terhadap nilai dari siswa yang ada, dan selalu melakukan pengurutan nilai siswa secara ascending. Sehingga pada saat pengguna ingin menampilkan data. Program akan menampilkan data nilai siswa secara terurut ascending, walaupun pada saat penginputan data dilakukan secara acak tidak terurut karena akan disusun oleh program yang ada.

```
=== Sistem Nilai Siswa Dinamis ===
1. Tambah Data
2. Tampilkan Data
0. Keluar
Pilih: 1

Masukkan nama: andi
Masukkan nilai: 100

=== Sistem Nilai Siswa Dinamis ===
1. Tambah Data
2. Tampilkan Data
0. Keluar
Pilih: 1

Masukkan nama: deni
Masukkan nilai: 80

=== Sistem Nilai Siswa Dinamis ===
1. Tambah Data
2. Tampilkan Data
0. Keluar
Pilih: 1

Masukkan nama: baron
Masukkan nilai: 90

=== Sistem Nilai Siswa Dinamis ===
1. Tambah Data
2. Tampilkan Data
0. Keluar
Pilih: 2
```



Gambar 02. Sistem penyimpanan Single Linked List selalu dalam kondisi terurut (ascending)

Link GitHub/GDB Online:

1. https://github.com/kenzie781/PRAKTIKUM-ASD-RD-124140103/tree/1e3f1891adf934f5439e4ef863d0ce8cdd211973/Tugas%202

Referensi

KALAU KALIAN PAKE GEN AI (CHAT GPT, GEMINI, CLAUDE, DLL. KALIAN BISA SHARE LINK PERCAKAPAN LINK GEN AI KALIAN DI SINI SEBAGAI BAHAN REFERENSI CODE KALIAN)