Laporan Praktikum

Algoritma dan Struktur Data

Ganjil 2025/2026
Program Studi Teknik Informatika
Institut Teknologi Sumatera

Modul : Single Linked List

Nama : Adhitya Warman

NIM : 124140007

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.

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 merupakan salah satu bentuk struktur data dinamis yang terdiri dari sekumpulan elemen yang disebut *node* dan saling terhubung satu arah melalui pointer. Setiap node pada Single Linked List terdiri dari dua bagian utama, yaitu bagian data (*info field*) yang berfungsi untuk menyimpan informasi, serta bagian alamat (*link field*) yang digunakan untuk menunjuk ke node berikutnya. Node pertama dalam list disebut *head*, sedangkan node terakhir menunjuk ke *NULL* yang

menandakan akhir dari list. Tidak seperti array yang memiliki ukuran tetap, Single Linked List bersifat dinamis karena jumlah elemen di dalamnya dapat bertambah atau berkurang sesuai kebutuhan selama program berjalan.

Struktur data ini memiliki karakteristik utama berupa hubungan satu arah antar-node, di mana setiap node hanya mengetahui keberadaan node setelahnya. Hal tersebut membuat proses akses data dilakukan secara berurutan mulai dari node pertama hingga node terakhir. Meskipun demikian, Single Linked List menawarkan fleksibilitas yang tinggi karena proses penambahan maupun penghapusan elemen dapat dilakukan dengan efisien tanpa perlu menggeser elemen lain di memori seperti pada array.

Beberapa operasi dasar yang dapat dilakukan pada Single Linked List meliputi pembuatan list kosong (*CreateEmpty*), pengecekan kondisi list (*IsEmpty*), penambahan node baru (*Insert*), penghapusan node (*Delete*), dan penelusuran data (*Traversal*) untuk menampilkan atau mencari elemen tertentu. Dalam penerapannya, Single Linked List banyak digunakan untuk memanipulasi data yang tidak memiliki jumlah tetap serta menjadi dasar bagi pengembangan struktur data lain seperti Double Linked List dan Circular Linked List.

Kelebihan utama dari Single Linked List adalah kemampuannya dalam mengelola data secara dinamis serta efisiensi dalam proses penyisipan dan penghapusan data. Namun, kelemahannya terletak pada akses data yang hanya dapat dilakukan secara sekuensial dan penggunaan memori tambahan untuk menyimpan pointer. Secara keseluruhan, Single Linked List merupakan struktur data yang penting dan fundamental dalam pemrograman karena konsepnya banyak digunakan dalam pembuatan struktur data yang lebih kompleks seperti *stack*, *queue*, serta berbagai sistem penyimpanan data yang memerlukan efisiensi dan fleksibilitas tinggi.

Source Code

```
1. #include <iostream>     // Header standar untuk input-output
2. using namespace std;    // Agar tidak perlu menulis std:: di setiap penggunaan
3.
4. // ======================
5. // Struktur Node (elemen Linked List)
6. // ======================
7. struct Node {
8.     char nama[50];  // Menyimpan nama siswa (maks. 50 karakter)
9.     int nilai;      // Menyimpan nilai siswa
10.    Node* next;     // Pointer yang menunjuk ke node berikutnya
11.};
12.
13.// Pointer awal dari Linked List (list dimulai dari sini)
14.Node* head = NULL;
```

```
15.
16.// ======================
17.// Fungsi untuk menambahkan data di akhir list
18.// ======================
19.void tambahData(char nama[], int nilai) {
20.    // Buat node baru
21.    Node* baru = new Node;
22.
23.    // Salin data nama ke dalam node baru (manual karena menggunakan char[])
24.    int i = 0;
25.    while (nama[i] != '\0') {
26.        baru->nama[i] = nama[i];
27.        i++;
28.    }
29.    baru->nama[i] = '\0';    // Tambahkan karakter akhir string
30.    baru->nilai = nilai;     // Isi nilai
31.    baru->next = NULL;       // Node baru belum punya penerus (next = NULL)
32.
33.    // Jika list masih kosong, jadikan node ini sebagai head
34.    if (head == NULL)
35.        head = baru;
36.    else {
37.        // Jika sudah ada data, telusuri hingga node terakhir
38.        Node* temp = head;
39.        while (temp->next != NULL)
40.            temp = temp->next;
41.        // Sambungkan node baru di akhir list
42.        temp->next = baru;
43.    }
44.
45.    cout << "Data berhasil ditambahkan!\n";
46.}
47.
48.// ======================
49.// Fungsi untuk menampilkan semua data siswa
50.// ======================
51.void tampilkanData() {
52.    // Jika list kosong
53.    if (head == NULL) {
54.        cout << "\nData masih kosong.\n";
55.        return;
56.    }
57.
58.    // Jika ada data, tampilkan semuanya
59.    Node* temp = head;
60.    cout << "\n=== Daftar Nilai Siswa ===\n";
61.    while (temp != NULL) {
62.        // Menampilkan nama dan nilai setiap node
63.        cout << "Nama : " << temp->nama << "\t| Nilai : " << temp->nilai << endl;
```

```
64.         temp = temp->next; // Pindah ke node berikutnya
65.     }
66. }
67.
68. // ======================
69. // Fungsi utama (main program)
70. // ======================
71. int main() {
72.     int pilihan;   // Untuk memilih menu
73.     char nama[50]; // Input nama siswa
74.     int nilai;     // Input nilai siswa
75.
76.     do {
77.         // Menu utama
78.         cout << "\n=== SISTEM NILAI SISWA DINAMIS ===\n";
79.         cout << "1. Tambah Data Siswa\n";
80.         cout << "2. Tampilkan Semua Data\n";
81.         cout << "3. Keluar\n";
82.         cout << "Pilih menu: ";
83.         cin >> pilihan;
84.         cin.ignore(); // Membersihkan karakter newline (\n) agar input berikutnya
    tidak terganggu
85.
86.         switch (pilihan) {
87.             case 1:
88.                 // Input data siswa baru
89.                 cout << "\nMasukkan Nama Siswa : ";
90.                 cin.getline(nama, 50); // Membaca nama lengkap (bisa mengandung
    spasi)
91.                 cout << "Masukkan Nilai      : ";
92.                 cin >> nilai;
93.                 cin.ignore(); // Bersihkan newline sebelum kembali ke menu
94.                 tambahData(nama, nilai); // Panggil fungsi untuk menambah data
95.                 break;
96.
97.             case 2:
98.                 // Menampilkan semua data siswa
99.                 tampilkanData();
100.                     break;
101.
102.                case 3:
103.                    // Keluar dari program
104.                    cout << "\nTerima kasih!\n";
105.                    break;
106.
107.                default:
108.                    // Jika input menu tidak sesuai
109.                    cout << "Pilihan tidak valid!\n";
110.            }
```

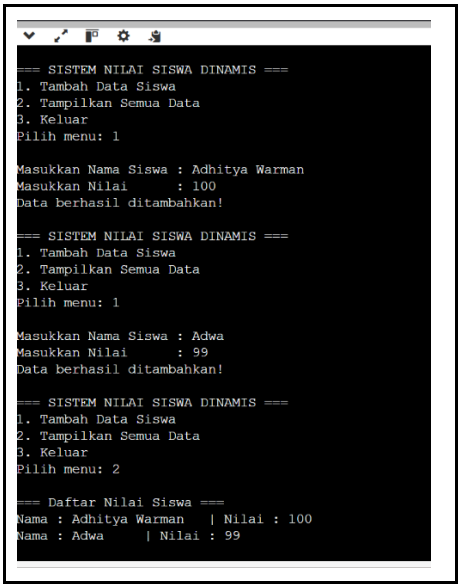
```
111.
112.          } while (pilihan != 3); // Ulangi selama pilihan bukan "Keluar"
113.
114.          return 0; // Program selesai
115.      }
```

```
1. #include <iostream>     // Library standar untuk input-output
2. using namespace std;    // Agar bisa menggunakan cout, cin tanpa menulis std::
3.
4. // ================================
5. // Struktur Node untuk menyimpan data siswa
6. // ================================
7. struct Node {
8.     char nama[50];  // Menyimpan nama siswa (maksimum 50 karakter)
9.     int nilai;      // Menyimpan nilai siswa
10.    Node* next;     // Pointer yang menunjuk ke node berikutnya
11.};
12.
13.// Pointer global sebagai penanda elemen pertama (head) dari linked list
14.Node* head = NULL;
15.
16.// ================================
17.// Fungsi: tambahDataTerurut
18.// Tujuan: Menambahkan data siswa secara terurut (ascending berdasarkan nilai)
19.// ================================
20.void tambahDataTerurut(char nama[], int nilai) {
21.    // Buat node baru
22.    Node* baru = new Node;
23.
24.    // Salin nama dari input ke field nama node baru (manual karena pakai array
   char)
25.    int i = 0;
26.    while (nama[i] != '\0') {  // Ulangi sampai akhir string
27.        baru->nama[i] = nama[i];
28.        i++;
29.    }
30.    baru->nama[i] = '\0';     // Tambahkan karakter akhir string
31.    baru->nilai = nilai;      // Simpan nilai ke node baru
32.    baru->next = NULL;        // Set pointer next menjadi NULL (belum terhubung)
33.
34.    // Jika list kosong atau nilai lebih kecil dari nilai node pertama (head)
35.    if (head == NULL || nilai < head->nilai) {
36.        // Sisipkan di awal list
37.        baru->next = head;  // Node baru menunjuk ke node lama (head)
38.        head = baru;        // Jadikan node baru sebagai head
39.    }
40.    else {
41.        // Jika list tidak kosong, cari posisi yang tepat untuk menyisipkan
42.        Node* temp = head;
```

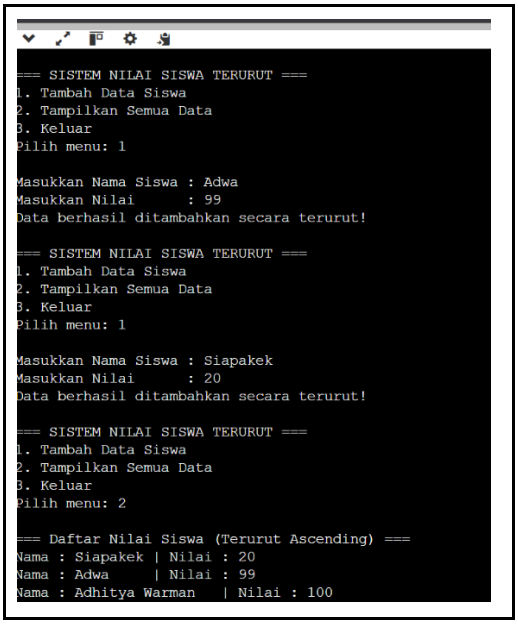
```
43.
44.         // Telusuri hingga menemukan node dengan nilai lebih besar
45.         // atau hingga mencapai akhir list
46.         while (temp->next != NULL && temp->next->nilai < nilai) {
47.             temp = temp->next;
48.         }
49.
50.         // Setelah posisi ditemukan, hubungkan node baru ke dalam list
51.         baru->next = temp->next; // Node baru menunjuk ke node setelah temp
52.         temp->next = baru;       // Node sebelumnya menunjuk ke node baru
53.     }
54.
55.     cout << "Data berhasil ditambahkan secara terurut!\n";
56. }
57.
58. // ================================
59. // Fungsi: tampilkanData
60. // Tujuan: Menampilkan seluruh data siswa dari awal hingga akhir list
61. // ================================
62. void tampilkanData() {
63.     if (head == NULL) { // Jika list kosong
64.         cout << "\nData masih kosong.\n";
65.         return;
66.     }
67.
68.     Node* temp = head;  // Mulai dari node pertama (head)
69.     cout << "\n=== Daftar Nilai Siswa (Terurut Ascending) ===\n";
70.
71.     // Telusuri list dan tampilkan semua data
72.     while (temp != NULL) {
73.         cout << "Nama : " << temp->nama << "\t| Nilai : " << temp->nilai << endl;
74.         temp = temp->next;  // Pindah ke node berikutnya
75.     }
76. }
77.
78. // ================================
79. // Fungsi utama (main program)
80. // Menyediakan menu interaktif bagi pengguna
81. // ================================
82. int main() {
83.     int pilihan;   // Variabel untuk menyimpan pilihan menu
84.     char nama[50]; // Variabel untuk input nama siswa
85.     int nilai;     // Variabel untuk input nilai siswa
86.
87.     do {
88.         // Tampilkan menu utama
89.         cout << "\n=== SISTEM NILAI SISWA TERURUT ===\n";
90.         cout << "1. Tambah Data Siswa\n";
91.         cout << "2. Tampilkan Semua Data\n";
```

```
92.         cout << "3. Keluar\n";
93.         cout << "Pilih menu: ";
94.         cin >> pilihan;
95.         cin.ignore(); // Bersihkan karakter newline (\n) agar getline tidak
    terganggu
96.
97.         switch (pilihan) {
98.             case 1:
99.                 // Input data siswa
100.                    cout << "\nMasukkan Nama Siswa : ";
101.                    cin.getline(nama, 50); // Membaca nama lengkap (bisa ada
    spasi)
102.                    cout << "Masukkan Nilai      : ";
103.                    cin >> nilai;
104.                    cin.ignore(); // Bersihkan buffer sebelum kembali ke menu
105.                    tambahDataTerurut(nama, nilai); // Panggil fungsi insert
    terurut
106.                    break;
107.
108.                case 2:
109.                    // Tampilkan seluruh data siswa
110.                    tampilkanData();
111.                    break;
112.
113.                case 3:
114.                    // Keluar dari program
115.                    cout << "\nTerima kasih!\n";
116.                    break;
117.
118.                default:
119.                    // Jika pengguna memasukkan angka selain 1-3
120.                    cout << "Pilihan tidak valid!\n";
121.            }
122.        } while (pilihan != 3); // Ulangi selama pengguna belum memilih keluar
123.
124.        return 0; // Program selesai
125.    }
```

Dokumentasi Hasil Running



Gambar 1. Hasil Running dari add and show data



Gambar 2. Hasil Running dari add and show data, namun akan diurutkan dari yang nilai terkecil ke terbesar secara urutan

Contoh Penggunaan:

Tujuan penggunaan program ini adalah untuk mengimplementasikan konsep *Single Linked List* yang mampu melakukan penyimpanan data secara dinamis sekaligus menjaga agar data selalu berada dalam urutan yang teratur (ascending). Melalui penerapan metode *Insert Sorted*, setiap data baru yang dimasukkan tidak hanya ditambahkan ke dalam list, tetapi juga langsung ditempatkan pada posisi yang sesuai berdasarkan nilai yang dimilikinya. Dengan demikian, data akan selalu tersusun dari nilai terkecil hingga terbesar tanpa perlu dilakukan proses pengurutan ulang. Pendekatan ini bertujuan untuk meningkatkan efisiensi dalam pengelolaan data, terutama pada sistem yang membutuhkan penyimpanan dan penelusuran data secara berurutan. Selain itu, penggunaan struktur *Single Linked List* memberikan fleksibilitas dalam penambahan data karena tidak memerlukan alokasi memori dengan ukuran tetap seperti pada array, sehingga program ini mampu mengatur data siswa secara efisien, terstruktur, dan mudah diakses kapan saja.

Link GitHub/GDB Online:

1. https://github.com/AW2606/LAPRAK-ASD-RD-007.git

Referensi

Drozdek, A. (2013). *Data Structures and Algorithms in C++* (4th ed.). Boston: Cengage Learning.

Malik, D. S. (2010). *Data Structures Using C++* (2nd ed.). Boston: Course Technology.

Wicaksono, A. (2018). *Algoritma dan Struktur Data Menggunakan C++*. Yogyakarta: Deepublish.

Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++* (4th ed.). Pearson Education.

Siregar, R. H., & Lubis, A. (2020). *Algoritma dan Struktur Data*. Medan: Yayasan Kita Menulis.

GeeksforGeeks. (2024). *Linked List Data Structure*. Retrieved from https://www.geeksforgeeks.org/data-structures/linked-list/

Chat GPT