

LAPORAN PRAKTIKUM

Modul 4

Single Linked List



Disusun Oleh:

Eduardo Bagus Prima Julian

2311104025 S1SE-07-01

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

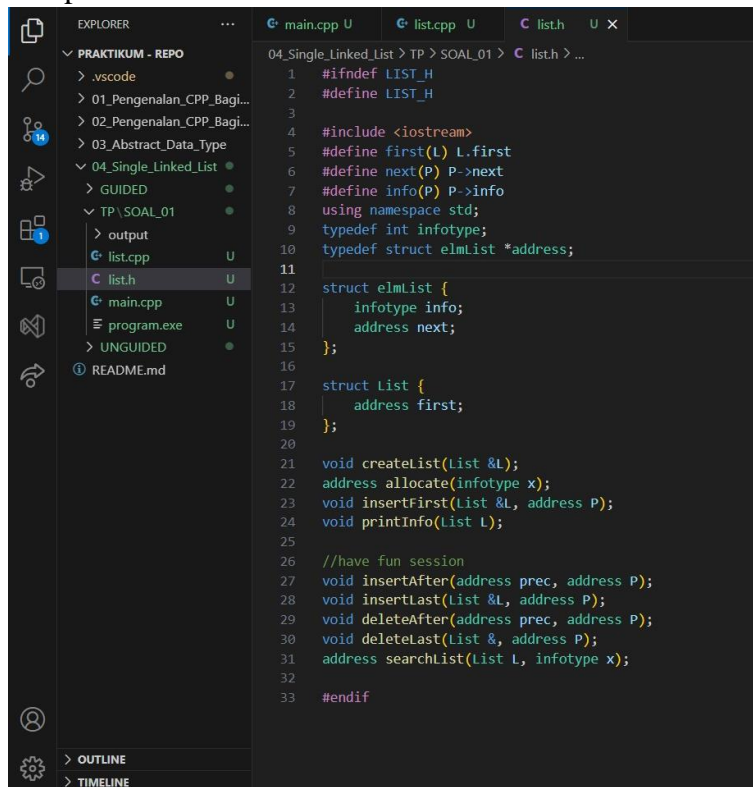
TELKOM UNIVERSITY PURWOKERTO

2024

I. Code

1. list.h

File header pada gambar mendefinisikan struktur dan fungsi untuk mengelola linked list sederhana dalam C++. Ia mencakup definisi untuk node (struktur `elmList`) yang menyimpan data bertipe integer dan pointer ke node berikutnya, serta struktur `List` yang menunjuk ke node pertama. Terdapat berbagai fungsi untuk membuat list, mengalokasikan memori untuk node baru, memasukkan node di awal atau akhir list, mencetak isi list, serta menghapus node tertentu dan mencari node berdasarkan nilai. Penggunaan header guard mencegah file di-include lebih dari sekali, menjaga kode tetap terorganisir dan bebas dari kesalahan kompilasi.



```
1  #ifndef LIST_H
2  #define LIST_H
3
4  #include <iostream>
5  #define first(L) L.first
6  #define next(P) P->next
7  #define info(P) P->info
8  using namespace std;
9  typedef int infotype;
10 typedef struct elmList *address;
11
12 struct elmList {
13     infotype info;
14     address next;
15 };
16
17 struct List {
18     address first;
19 };
20
21 void createList(List &L);
22 address allocate(infotype x);
23 void insertFirst(List &L, address P);
24 void printInfo(List L);
25
26 //have fun session
27 void insertAfter(address prec, address P);
28 void insertLast(List &L, address P);
29 void deleteAfter(address prec, address P);
30 void deleteLast(List &L, address P);
31 address searchList(List L, infotype x);
32
33 #endif
```

2. list.cpp

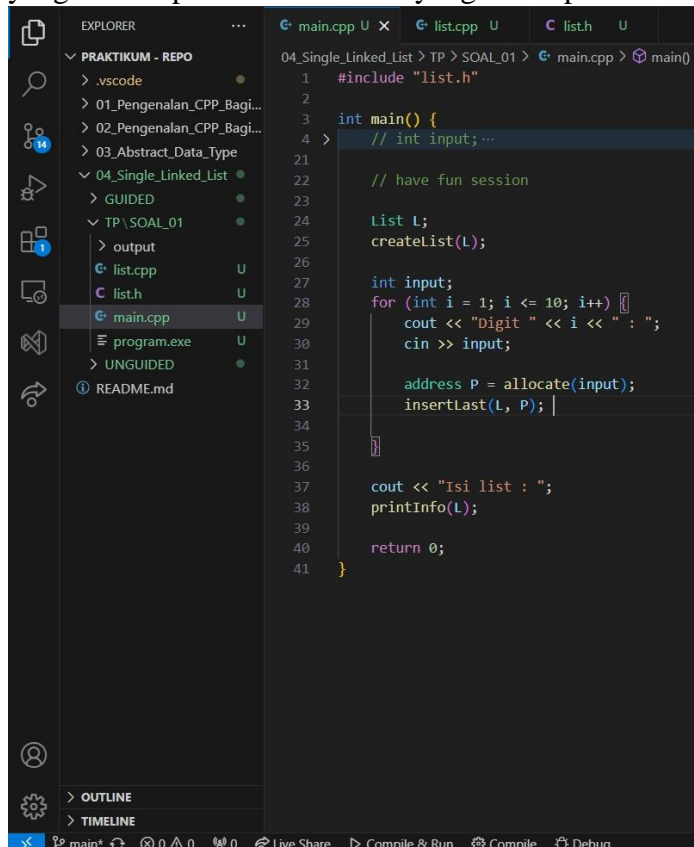
Kode digambar merupakan implementasi fungsi-fungsi untuk mengelola linked list sederhana dalam C++. Fungsi `createList` menginisialisasi list dengan menjadikannya kosong, sedangkan `allocate` mengalokasikan memori untuk node baru dengan menyimpan nilai integer. Fungsi `insertFirst` dan `insertLast` menambahkan node di awal dan akhir list, masing-masing, dengan mengatur pointer dengan benar. `printInfo` mencetak nilai setiap node dalam list. Fungsi `insertAfter` menyisipkan node baru setelah node yang diberikan, sementara `deleteAfter` dan `deleteLast` menghapus node setelah node yang ditunjuk dan node terakhir dalam list. Terakhir, `searchList` mencari dan mengembalikan node yang memiliki nilai tertentu, atau mengembalikan `NULL` jika tidak ditemukan.

```
1 #include "list.h"
2 using namespace std;
3
4 void createList(List &L) {
5     first(L) = NULL;
6 }
7
8 address allocate(infotype x) {
9     address p = new elmList();
10    info(p) = x;
11    next(p) = NULL;
12    return p;
13 }
14
15
16 void insertFirst(List &L, address P) {
17     next(P) = first(L);
18     first(L) = P;
19 }
20
21 void printInfo(List L) {
22     address p = first(L);
23     while (p != NULL) {
24         std::cout << info(p) << " ";
25         p = next(p);
26     }
27     std::cout << std::endl;
28 }
29
30 //have fun session
31 void insertAfter(address prec, address P) {
32     next(P) = next(prec);
33     next(prec) = P;
34 }
35
36 void insertLast(List &L, address P) {
37     if (first(L) == NULL) {
38         first(L) = P;
39     }
40 }
```

```
31 void insertAfter(address prec, address P) {
32     next(P) = next(prec);
33     next(prec) = P;
34 }
35
36 void insertLast(List &L, address P) {
37     if (first(L) == NULL) {
38         first(L) = P;
39     } else {
40         address temp = first(L);
41         while (next(temp) != NULL) {
42             temp = next(temp);
43         }
44         next(temp) = P;
45     }
46     next(P) = NULL;
47     //P = first(L);
48     //next(P) = P;
49 }
50
51 void deleteAfter(address prec, address P) {
52     P = next(prec);
53     next(prec) = next(P);
54 }
55
56 void deleteLast(List &L, address P) {
57     P = first(L);
58     P = next(P);
59     next(P) = NULL;
60 }
61
62 address searchList(List L, infotype x) {
63     address P = first(L);
64     while (P != NULL) {
65         if (info(P) == x) {
66             return P;
67         }
68         P = next(P);
69     }
70     return NULL;
71 }
```

3. main.cpp

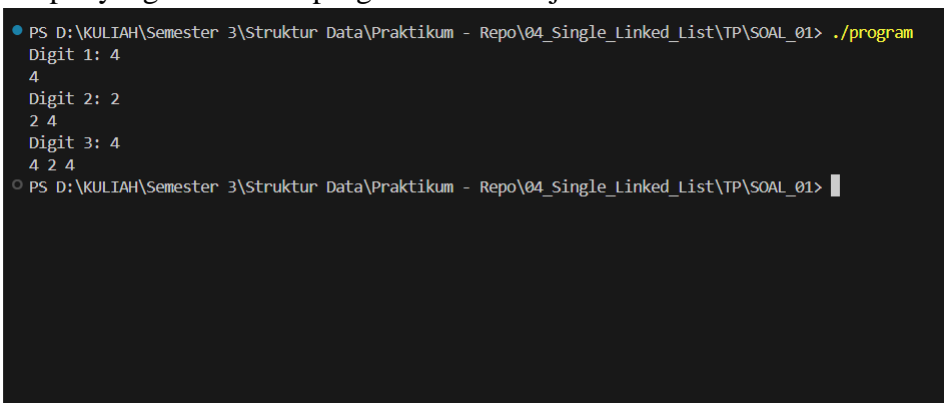
Kode di atas adalah implementasi fungsi utama (main) untuk program yang mengelola linked list sederhana. Pertama, ia mendeklarasikan sebuah variabel list L dan menginisialisasinya dengan fungsi createList, yang menjadikannya kosong. Selanjutnya, dalam loop yang meminta input dari pengguna sebanyak 10 kali, setiap digit yang dimasukkan dialokasikan memori sebagai node baru dengan fungsi allocate, lalu disisipkan di akhir list menggunakan insertLast. Setelah semua digit dimasukkan, program mencetak isi list menggunakan fungsi printInfo, yang menampilkan semua nilai yang tersimpan dalam list.



```
1 #include "list.h"
2
3 int main() {
4     // int input; ...
5
6     // have fun session
7
8     List L;
9     createList(L);
10
11     int input;
12     for (int i = 1; i <= 10; i++) {
13         cout << "Digit " << i << " : ";
14         cin >> input;
15
16         address P = allocate(input);
17         insertLast(L, P);
18     }
19
20     cout << "Isi list : ";
21     printInfo(L);
22
23     return 0;
24 }
```

4. output

output yang keluar dari program setelah dijalankan



```
PS D:\KULIAH\Semester 3\Struktur Data\Praktikum - Repo\04_Single_Linked_List\TP\SOAL_01> ./program
Digit 1: 4
4
Digit 2: 2
2 4
Digit 3: 4
4 2 4
PS D:\KULIAH\Semester 3\Struktur Data\Praktikum - Repo\04_Single_Linked_List\TP\SOAL_01>
```

Output pada sesi have fun

```
Digit 8 : 8
Digit 9 : 9
Digit 10 : 4
Isi list : 1 2 3 4 5 6 7 8 9 4
PS D:\KULIAH\Semester 3\Struktur Data\Praktikum - Repo\04_Single_Linked_List\TP\SOAL_01> g++ list.cpp main.cpp -o program
PS D:\KULIAH\Semester 3\Struktur Data\Praktikum - Repo\04_Single_Linked_List\TP\SOAL_01> ./program
Digit 1 : 2
Digit 2 : 3
Digit 3 : 1
Digit 4 : 1
Digit 5 : 1
Digit 6 : 0
Digit 7 : 4
Digit 8 : 0
Digit 9 : 1
Digit 10 : 3
Isi list : 2 3 1 1 1 0 4 0 1 3
PS D:\KULIAH\Semester 3\Struktur Data\Praktikum - Repo\04_Single_Linked_List\TP\SOAL_01> |
```