LAPORAN PRAKTIKUM
PERTEMUAN 4
SINGLE LINKED LIST

Nama :

Martryatus Sofia 2311104003

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO 2024

A. Soal Praktek

Ikuti langkah-langkah berikut untuk mengerjakan TP modul 4 Materi Single Linked List :

1. **Membuat deklarasi tipe List**

   Buat file list.h dan ketik sintak sesuai gambar berikut.

```
#include <iostream>
#define first(L) L.first
#define next(P) P->next
#define info(P) P->info
using namespace std;
typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};
struct List{
    address first;
};
```

   Buat file list.cpp dan ketik sintak berikut

```
#include <iostream>
#include "list.h"
using namespace std;
```

2. **Membuat list kosong, yaitu procedure createList.**

   Tambahkan pada list.h primitif dari procedure createList

```
void createList(List &L);
```

   Tambahkan pada list.cpp implementasi dari procedure createList, sintak c++ sebagai berikut:

```
void createList(List &L){
    /** this procedure will initialize
        the list L*/
    first(L) = NULL;
}
```

3. **Setelah list sudah ada, selanjutnya buatlah elemen dengan menggunakan fungsi allocate.**

   Tambahkan pada list.h primitif dari fungsi allocate

```
address allocate(infotype x);
```

Tambahkan pada list.cpp implementasi dari fungsi allocate, sintak c++ sebagai berikut:

```
address allocate(infotype x){

    address p = new elmlist;
    info(p) = x;
    next(p) = NULL;

    return p;
}
```

4. **Setelah List dan elemen sudah ada, maka selanjutnya elemen tersebut harus diinsert ke List agar bisa menjadi elemen list.** Proses insert dapat menggunakan procedure Insert First, procedure Insert Last, atau procedure insert After. Pada Tugas Pendahuluan kali ini, akan dicontohkan menggunakan insert first.

   Tambahkan pada list.h primitif procedure insertFirst

```
void insertFirst(List &L, address P);
```

   Tambahkan pada list.cpp implementasi dari procedure insertFirst sesuai sintak berikut :

```
void insertFirst(List &L, address P){
    /** TODO: Insert the new element pointed
              by P to the first of list L*/
    // YOUR CODES HERE
    //------------------------
    next(P) = first(L);
    first(L) = P;
    //------------------------
}
```

5. **Setelah proses insert elemen, maka agar bisa mengetahui apakah elemen berhasil diinsertkan, maka kita perlu menampilkan isi list.**

   Tambahkan pada list.h primitif procedure printInfo

```
void printInfo (List L);
```

   Tambahkan pada list.cpp implementasi dari proc printInfo, sintak C++ sebagai berikut :

```
void printInfo(List L){
    /** this procedure will output
        the info of each element
        in list L*/

    address p = first(L);
    while (p != NULL){
        cout << info(p) << ", ";
        p = next(p);
    }
    cout << endl;
}
```

6. Sekarang, setelah ADT List sudah terisi dengan beberapa fungsi Procedur di atas, maka mari buat sebuah List berisi 3 elemen yang berisi 3 digit nim terakhir Anda di main.cpp
Adapun gambaran isi dari main.cpp nya adalah sbb :

```
#include <iostream>
#include "list.h"
using namespace std;


int main()
{
    // 1. Panggilah create list

    // 2. Buat sintak menanyakan angka pertama yang ingin diinputkan user ke list

    // 3. Panggil fungsi allocate agar data tersebut dialaskan elemen

    // 4. Panggil procedure insert first yang telah dibuat

    // 5. panggil procedure show info untuk mengecek apakah angak tersebut berhasil menjadi elemen di List.

    // 6. buat kembali sintak no 2 s/d no 5 untuk data angka kedua dari user

    // 7. buat kembali sintak no 2 s/d no 5 untuk data angka ketiga dari user


    return 0;

}
```

Tugas rekan-rekan adalah mengisi main.cpp di atas dengan sintak C++ sesuai dengan petunjuk. Buka kembali modul sebelumnya ya ^_^. Setelah selesai, compile dan run. Lalu **screen capture hasilnya.**

Jika dilayar muncul 3 digit nim Anda yang muncul secara terurut TERBALIK dari urutan input.

7. SESI HAVE FUN. Rekan-rekan dapat mencoba hal di bawah ini agar memudahkan saat praktikum:
   i. Tambahkan procedure insertLast, insertAfter, deleteLast, deleteAfter pada list.h dan list.cpp
   ii. Tambahkan Function searchInfo pada list.h dan list.cpp
   iii. Ubah main.cpp agar proses insert N data tidak satu persatu, tapi sesuai dengan jumlah digit NIM yaitu 10 data (clue : gunakan looping). Dan NIM yang diinput, saat di show tidak boleh terurut terbalik (clue : gunakan insert Last) Tampilan (underscore adalah inputan user):

```
Masukkan NIM perdigit
Digit 1 : 1
Digit 2 : 1
Digit 3 : 3
Digit 4 : 1
Digit 5 : 9
Digit 6 : 6
Digit 7: 4
Digit 8 : 7
Digit 9 : 4
Digit 10 : 2
Isi list : 1131964742
```

Jawab :

List.h

```h
#ifndef LIST_H
#define LIST_H

#include <iostream>

#define first(L) L.first
#define next(P) P->next
#define info(P) P->info

using namespace std;

typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};

struct List {
    address first;
};

void createList(List &L);
address allocate(infotype x);
void insertFirst(List &L, address P);
void insertLast(List &L, address P);
address searchInfo(List L, infotype x);
void printInfo(List L);
void deleteLast(List &L, address &P);
void deleteAfter(List &L, address prec,
address &P);

#endif
```

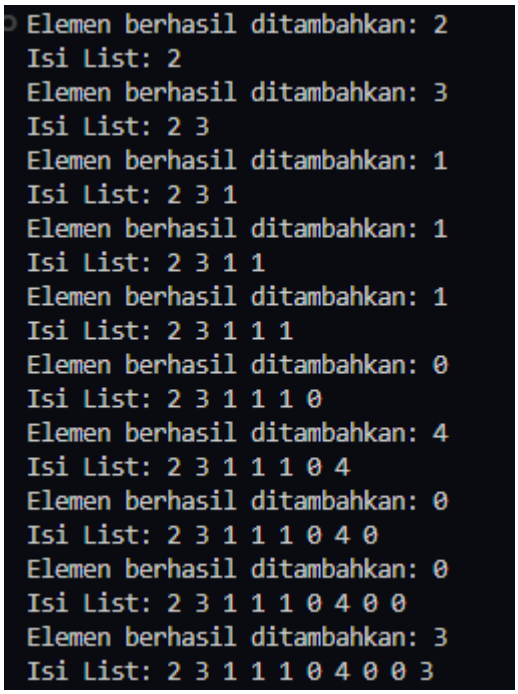
List.cpp

```cpp
#include <iostream>
#include "list.h"

using namespace std;

void createList(List &L) {
    first(L) = NULL;
}

address allocate(infotype x) {
    address p = new elmlist;
    if (p != NULL) {
        info(p) = x;
        next(p) = NULL;
    }
    return p;
}

void insertFirst(List &L, address P) {
    next(P) = first(L);
    first(L) = P;
}

void insertLast(List &L, address P) {
    if (first(L) == NULL) {
        first(L) = P;
    } else {
        address last = first(L);
        while (next(last) != NULL) {
            last = next(last);
        }
        next(last) = P;
    }
}

void printInfo(List L) {
    address p = first(L);
    while (p != NULL) {
        cout << info(p) << " ";
        p = next(p);
    }
    cout << endl;
}

address searchInfo(List L, infotype x) {
    address p = first(L);
    while (p != NULL && info(p) != x) {
        p = next(p);
    }
    return p;
}

void deleteLast(List &L, address &P) {
    if (first(L) != NULL) {
        if (next(first(L)) == NULL) {
            P = first(L);
            first(L) = NULL;
        } else {
            address prec = first(L);
            while (next(next(prec)) != NULL) {
                prec = next(prec);
            }
            P = next(prec);
            next(prec) = NULL;
        }
    }
}

void deleteAfter(List &L, address prec, address
&P) {
    if (next(prec) != NULL) {
        P = next(prec);
        next(prec) = next(P);
    }
}
```

Main.cpp

```cpp
#ifndef LIST_H
#define LIST_H

#include <iostream>

#define first(L) L.first
#define next(P) P->next
#define info(P) P->info

using namespace std;

typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};

struct List {
    address first;
};

void createList(List &L);
address allocate(infotype x);
void insertFirst(List &L, address P);
void insertLast(List &L, address P);
address searchInfo(List L, infotype x);
void printInfo(List L);
void deleteLast(List &L, address &P);
void deleteAfter(List &L, address prec, address &P);

#endif
```

HASIL RUNNING



SELAMAT ANDA BERHASIL MEMBUAT SINGLE LINGKED LIST.