LAPORAN PRAKTIKUM

- **Identitas Praktikum**

  Nama MK : Struktur Data
  Kode MK : CCK2AAB4
  Bobot SKS : 4 SKS
  Tempat : L-Program, Gedung DC, lantai 3
  Hari, tanggal : Selasa, 15 Oktober 2024
  Jam : 12:30-15:30 WIB
  Topik praktikum : Modul-4 SINGLE LINKED LIST (BAGIAN PERTAMA)

- **Identitas Mahasiswa**

  Nama lengkap : Afad Fath Musyarof Halim
  NIM : 2211104030
  Program Studi : S-1 Software Engineering

- **Hasil Praktikum**

  4.1 Linked List dengan Pointer

  Linked List adalah bentuk struktur data dimana ada banyak data yang saling berhubungan. Sedangkan pointer adalah bagaimana program mengakses nilai dari alokasi memori atau variable yang sudah dibuat

  4.2 Single Linked List

  SLL adalah Linked List dimana setiap data memiliki 2 bagian kepala dan ekor dimana kepala menyimpan data dan ekor menyimpan rujukan ke urutan data selanjutnya

  Contoh struktur data yang digunakan:

```
struct Mahasiswa
{
    string nama;
    string nim;
    string jurusan;
    Mahasiswa *next;
};
```

  - Nama, nim, jurusan berisi data
  - *next merujuk ke urutan data berikutnya

4.2.1 Single Linked List - Insert

```
void tambahMahasiswa(Mahasiswa *&head, string nama, string nim, string jurusan) {
    Mahasiswa *newMahasiswa = new Mahasiswa();
    newMahasiswa->nama = nama;
    newMahasiswa->nim = nim;
    newMahasiswa->jurusan = jurusan;
    newMahasiswa->next = nullptr;

    if (head == nullptr) {
        head = newMahasiswa;
    } else {
        Mahasiswa *temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
       }
        temp->next = newMahasiswa;
   }
}
```

- Membuat objek mahasiswa baru
- Mengisi data mahasiswa baru dari parameter yang didapat dan *next sebagai urutan kosong
- Cek apakah urutan data kosong
    - jika kosong masukan data dari mahasiswa baru
    - Jika terisi, iterasikan ke urutan selanjutnya sampai bertemu dengan urutan data kosong lalu isi dengan data dari mahasiswa baru

4.2.2 Single Linked List - View

```
void tampilkanMahasiswa(Mahasiswa *head) {
    if (head == nullptr) {
        cout << "Daftar mahasiswa kosong." << endl;
        return;
    }

    Mahasiswa *temp = head;
    while (temp != nullptr) {
        cout << "Nama: " << temp->nama << endl;
        cout << "NIM: " << temp->nim << endl;
        cout << "Jurusan: " << temp->jurusan << endl;
        cout << "------------------------" << endl;
        temp = temp->next;
    }
}
```

- Cek apakah urutan pertama kosong, jika kosong maka tidak ada sama sekali data dan selesaikan fungsi
- Jika urutan data terisi, tampilkan urutan data secara terurut berdasarkan iterasi urutan data

4.3 Single Linked List – Delete

4.3.1 Hapus Semua

```
1  void hapusMahasiswa(Mahasiswa *&head) {
2      while (head != nullptr) {
3          Mahasiswa *temp = head;
4          head = head->next;
5          delete temp;
6      }
7  }
```

- Selama urutan terisi
    - Dapatkan data urutan saat ini ke pointer
    - Pindah ke urutan data selanjutnya
    - Hapus urutan data saat ini dengan pointer

4.3.2 Hapus secara spesifik

```
void hapusMahasiswaNIM(Mahasiswa *&head, string nim) {
    Mahasiswa *temp = head;
    Mahasiswa *prev = nullptr;

    while (temp != nullptr && temp->nim != nim) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "NIM " << nim << " tidak ditemukan." << endl;
        return;
    }

    if (prev == nullptr) {
        head = temp->next;
    } else {
        prev->next = temp->next;
    }

    delete temp;
}
```

- Cek apakah urutan data saat ini kosong dan memiliki nim yg berbeda, jika iya pindah ke urutan data selanjutnya
- Jika sampai urutan data terakhir kosong, selesaikan fungsi dengan status nim tidak ditemukan
- Jika urutan data sebelumnya kosong, jadikan urutan data saat ini menjadi urutan data setelahnya
- Jika tidak, jadikan urutan data sebelumnya menjadi urutan data setelahnya
- Hapus urutan data saat ini

4.4 Single Linked List – Update/Before

```
void tambahSebelumNIM(Mahasiswa *&head, string nama, string nim, string jurusan, string nim_setelahnya) {
    Mahasiswa *newMahasiswa = new Mahasiswa();
    newMahasiswa->nama = nama;
    newMahasiswa->nim = nim;
    newMahasiswa->jurusan = jurusan;

    if (head == nullptr || head->nim == nim_setelahnya) {
        newMahasiswa->next = head;
        head = newMahasiswa;
        return;
    }

    Mahasiswa *temp = head;

    while (temp->next != nullptr && temp->next->nim != nim_setelahnya) {
        temp = temp->next;
    }

    if (temp->next == nullptr) {
        cout << "NIM " << nim_setelahnya << " tidak ditemukan." << endl;
        delete newMahasiswa;
    } else {
        newMahasiswa->next = temp->next;
        temp->next = newMahasiswa;
    }
}
```

- Buat objek mahasiswa baru
- Isi dengan parameter
- Jika urutan kosong atau nim urutan saat ini sama dengan parameter nim yg menentukkan posisi, jadikan urutan saat ini menjadi rujukan urutan selanjutnya dari posisi saat ini
- Iterasikan jika urutan setelahnya terisi dan nim tidak sama dengan parameter nim yg menentukkan posisi, pindah ke urutan selanjutnya
- Jika sampai urutan terakhir tidak ditemukan nim yg menentukan posisi, hapus objek mahasiswa baru
- Sisanya jadikan urutan saat ini menjadi rujukan urutan selanjutnya dari posisi saat ini

- **Soal**

  Buat sebuah single linked list yang memiliki 7 elemen bilangan prima (2, 3, 5, 7, 11, 13, 17).

1. Membuat single linked list

   - Bilangan.h

```h
#ifndef BILANGAN_H
#define BILANGAN_H

struct Bilangan {
    int Angka;
    Bilangan *next;
};

void insertBilangan(Bilangan *&head, int Angka);

#endif // BILANGAN_H
```

   - Bilangan.cpp

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

void insertBilangan(Bilangan *&head, int Angka) {
    Bilangan *newBilangan = new Bilangan();
    newBilangan->Angka = Angka;
    newBilangan->next = nullptr;

    if (head == nullptr) {
        head = newBilangan;
    }
    else {
        Bilangan *temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newBilangan;
    }
}
```

   - Main.cpp

```cpp
#include <iostream>
#include <string>
#include "bilangan.cpp"
using namespace std;

int main(){
    Bilangan *head = nullptr;

    cout << "1. Menambahkan 7 bilangan ke SLL ";
    int prima[] = {2, 3, 5, 7, 11, 13, 17};

    for (int i = 0; i < 7; i++) {
        insertBilangan(head, prima[i]);
    }
    cout << "\n================================\n" << endl;

    return 0;
}
```

- Output

```
Afadfath | output
# & .\'main.exe'
1. Menambahkan 7 bilangan ke SLL
================================

Afadfath | output
#
```

- Penjelasan insert
    - Membuat objek Bilangan baru
    - Isi objek dengan Angka parameter
    - Isi objek pointer next dengan nullptr / kosong
    - Cek apakah urutan saat ini kosong
        - Jika iya, masukan objek bilangan baru ke urutan tersebut
        - Jika tidak, iterasikan ke urutan selanjutnya sampai urutan kosong lalu masukan objek bilangan baru ke urutan tersebut

2. Menampilkan elemen single linked list
    - Tambahan di bilangan.h

```
void viewBilangan(Bilangan *head);
```

    - Tambahan di bilangan.cpp

```
void viewBilangan(Bilangan *head) {
    if (head == nullptr) {
        cout << "Daftar bilangan kosong" << endl;
        return;
    }

    Bilangan *temp = head;
    while (temp != nullptr) {
        cout << temp->Angka << " ";
        temp = temp->next;
    }
}
```

    - Tambahan di main.cpp

```
cout << "2. Menampilkan SLL: " << endl;
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output

```
Afadfath | output
# & .\'main.exe'
1. Menambahkan 7 bilangan ke SLL
================================

2. Menampilkan SLL:
2 3 5 7 11 13 17
================================

Afadfath | output
#
```

- Penjelasan Fungsi 'viewBilangan'
    - Cek apakah daftar kosong, jika iya maka fungsi berhenti dan menampilkan urutan kosong
    - Iterasikan dari urutan pertama sampai terakhir dan menampilkan data angka pada masing masing urutan

3. Menyisipkan elemen bernilai 1 di awai SLL
    - Tambahan bilangan.h

```
void TambahDiAwal(Bilangan *&head, int AngkaBaru);
```

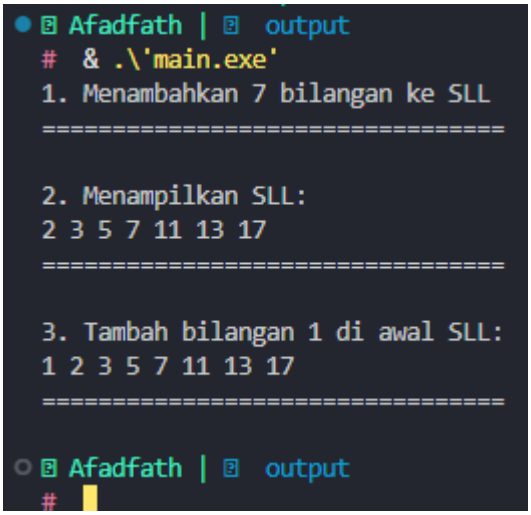
    - Tambahan bilangan.cpp

```
void TambahDiAwal(Bilangan *&head, int AngkaBaru) {
    Bilangan *newBilangan = new Bilangan;
    newBilangan->Angka = AngkaBaru;
    newBilangan->next = head;
    head = newBilangan;
}
```

    - Tambahan main.cpp

```
cout << "3. Tambah bilangan 1 di awal SLL: " << endl;
TambahDiAwal(head, 1);
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output



- Penjelasan Fungsi 'TambahDiAwal':
    - Buat objek baru
    - Masukan isi dengan Angka baru
    - Jadikan pointer next menjadi data dari urutan pertama
    - Jadikan urutan pertama menjadi objek baru

4. Menyisipkan elemen bernilal 19 di akhir SLL
    - Tambahan bilangan.h

```
void TambahDiAkhir(Bilangan *&head, int AngkaBaru);
```

    - Tambahan bilangan.cpp

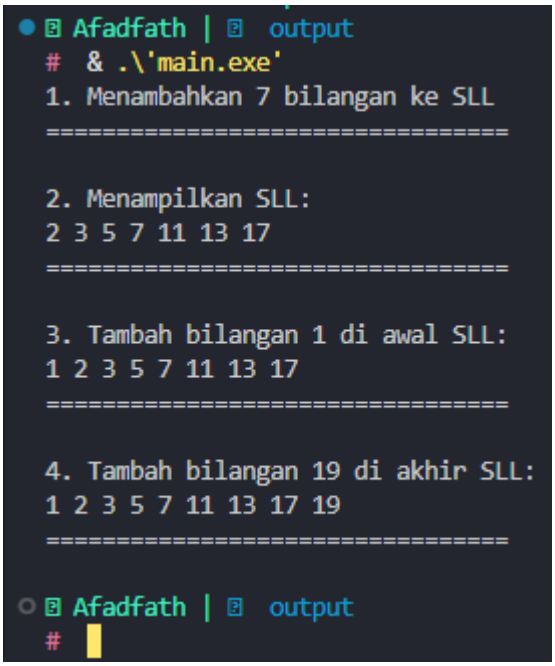
```
void TambahDiAkhir(Bilangan *&head, int AngkaBaru) {
    Bilangan *newBilangan = new Bilangan;
    newBilangan->Angka = AngkaBaru;
    newBilangan->next = nullptr;

    if (head == nullptr) {
        head = newBilangan;
    }
    else {
        Bilangan *temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newBilangan;
    }
}
```

- Tambahan main.cpp

```
cout << "4. Tambah bilangan 19 di akhir SLL: " << endl;
TambahDiAkhir(head, 19);
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output



- Penjelasan Fungsi 'TambahDiAkhir':
    - Buat objek baru
    - Masukan isi dengan Angka baru
    - Jadikan pointer next menjadi nullptr(kosong)
    - Jika urutan data pertama kosong maka jadikan objek menjadi urutan data pertama
    - Jika urutan data terisi
        - iterasikan ke urutan data selanjutnya sampai urutan data selanjutnya kosong
        - Jadikan urutan data selanjutnya menjadi objek yang telah dibuat

5. Menyisipkan elemen bernilai 10 di sebelum elemen 11
    - Tambahan bilangan.h

```
void SisipkanAngkaSebelum(Bilangan *&head, int AngkaBaru, int AngkaSebelum);
```

- Tambahan bilangan.cpp

```cpp
void SisipkanAngkaSebelum(Bilangan *&head, int AngkaBaru, int AngkaSebelum){
    Bilangan *newBilangan = new Bilangan;
    newBilangan->Angka = AngkaBaru;
    newBilangan->next = nullptr;

    if (head == nullptr) {
        head = newBilangan;
    }
    else {
        Bilangan *p = head;
        while (p->next != nullptr && p->next->Angka != AngkaSebelum) {
            p = p->next;
        }
        if (p->next == nullptr) {
            cout << "Posisi angka " << AngkaSebelum << " tidak ditemukan" << endl;
            return;
        }
        newBilangan->next = p->next;
        p->next = newBilangan;
    }
};
```

- Tambahan main.cpp

```cpp
cout << "5. Tambah bilangan 10 sebelum angka 11 di SLL: " << endl;
SisipkanAngkaSebelum(head, 10, 11);
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output

```
Afadfath | output
# & .\'main.exe'
1. Menambahkan 7 bilangan ke SLL
================================

2. Menampilkan SLL:
2 3 5 7 11 13 17
================================

3. Tambah bilangan 1 di awal SLL:
1 2 3 5 7 11 13 17
================================

4. Tambah bilangan 19 di akhir SLL:
1 2 3 5 7 11 13 17 19
================================

5. Tambah bilangan 10 sebelum angka 11 di SLL:
1 2 3 5 7 10 11 13 17 19
================================

Afadfath | output
#
```

- Penjelasan Fungsi 'SisipkanAngkaSebelum'
    - Buat objek baru
    - Jadikan isi menjadi Angka Baru
    - Jadikan pointer next menjadi nullptr(kosong)
    - Jika urutan data pertama kosong, jadikan objek menjadi urutan data pertama
    - Jika urutan data tidak kosong maka:
        - Dapatkan urutan data pertama (p)
        - Cek dan Iterasikan ke urutan data selanjutnya sampai urutan data selanjutnya tidak kosong dan berisi angka dari posisi yang di inginkan
            - Jika posisi tidak ditemukan maka output Angka tidak ditemukan
        - Jadikan pointer next dari objek yang dibuat menjadi urutan data selanjutnya
        - Jadikan urutan data selanjutnya menjadi objek yang telah dibuat

6. Menghapus elemen bernilai 1 di awal SSL
    - Tambahan bilangan.h

```
void hapusAwal(Bilangan *&head);
```

    - Tambahan bilangan.cpp

```
void hapusAwal(Bilangan *&head){
    Bilangan *hapus = head;
    head = head->next;
    delete hapus;
};
```

    - Tambahan main.cpp

```
cout << "6. Menghapus bilangan 1 di awal SSL: " << endl;
hapusAwal(head);
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output

```
Afadfath | output
# & .\'main.exe'
1. Menambahkan 7 bilangan ke SLL
==================================

2. Menampilkan SLL:
2 3 5 7 11 13 17
==================================

3. Tambah bilangan 1 di awal SLL:
1 2 3 5 7 11 13 17
==================================

4. Tambah bilangan 19 di akhir SLL:
1 2 3 5 7 11 13 17 19
==================================

5. Tambah bilangan 10 sebelum angka 11 di SLL:
1 2 3 5 7 10 11 13 17 19
==================================

6. Menghapus bilangan 1 di awal SSL:
2 3 5 7 10 11 13 17 19
==================================

Afadfath | output
#
```

- Penjelasan fungsi 'HapusAwal'
    - Buat pointer objek baru ke urutan data pertama
    - Majukan urutan data pertama ke selanjutnya
    - Hapus urutan data pertama dengan pointer objek baru

7. Menghapus elemen bernilai 19 di akhir SSL
    - Tambahan bilangan.h

```
void hapusAkhir(Bilangan *&head);
```

- Tambahan bilangan.cpp

```
void hapusAkhir(Bilangan *&head){
    if (head->next == nullptr) {
        delete head;
        head = nullptr;
        return;
    }

    Bilangan *p = head;
    while (p->next->next != nullptr) {
        p = p->next;
    }
    delete p->next;
    p->next = nullptr;
};
```

- Tambahan main.cpp

```
cout << "7. Menghapus bilangan 19 di akhir SSL: " << endl;
hapusAkhir(head);
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output

```
Afadfath | output
# & .\'main.exe'
1. Menambahkan 7 bilangan ke SLL
==================================

2. Menampilkan SLL:
2 3 5 7 11 13 17
==================================

3. Tambah bilangan 1 di awal SLL:
1 2 3 5 7 11 13 17
==================================

4. Tambah bilangan 19 di akhir SLL:
1 2 3 5 7 11 13 17 19
==================================

5. Tambah bilangan 10 sebelum angka 11 di SLL:
1 2 3 5 7 10 11 13 17 19
==================================

6. Menghapus bilangan 1 di awal SSL:
2 3 5 7 10 11 13 17 19
==================================

7. Menghapus bilangan 19 di akhir SSL:
2 3 5 7 10 11 13 17
==================================

Afadfath | output
#
```

- Penjelasan fungsi ‘hapusAkhir’
    - Cek apakah urutan data selanjutnya kosong, jika kosong hapus urutan data pertama
    - Buat pointer objek untuk merujuk posisi
    - Selama urutan ke 2 dari objek terisi maka iterasikan p ke urutan data selanjutnya
    - Hapus urutan data selanjutnya
    - Jadikan pointer next menjadi nullptr (kosong)

8. Menghapus elemen sebelum elemen bernilai 11
    - Tambahan bilangan.h

```
void hapusSebelumPosisi(Bilangan *&head, int Posisi);
```

- Tambahan bilangan.cpp

```
void hapusSebelumPosisi(Bilangan *&head, int Posisi){
    if (head->Angka == Posisi) {
        cout << "Tidak ada bilangan sebelum " << Posisi << endl;
        return;
    }

    if (head->next->Angka == Posisi) {
        hapusAwal(head);
        return;
    }

    Bilangan *p = head;
    while (p->next->next != nullptr && p->next->next->Angka != Posisi) {
        p = p->next;
    }

    Bilangan *hapus = p->next;
    p->next = p->next->next;
    delete hapus;
};
```

- Tambahan main.cpp

```
cout << "8. Menghapus bilangan sebelum bilangan 11: " << endl;
hapusSebelumPosisi(head, 11);
viewBilangan(head);
cout << "\n================================\n" << endl;
```

- Output

- Penjelasan fungsi ‘hapusSebelumPosisi’
    - Jika angka posisi yang ditentukan berada di urutan pertama maka tidak ada bilangan sebelum posisi tersebut
    - Jika angka posisi yang ditentukan berada di urutan kedua maka hapus urutan pertama
    - Buat pointer objek (p) untuk menentukkan posisi
    - Cek apakah urutan ke 2 dari p terisi dan tidak di isi dengan angka posisi maka majukan urutan p lalu ulangi
    - Buat pointer objek dari angka yang akan dihapus
    - Jadikan urutan setelah p menjadi urutan setelahnya lagi
    - Hapus angka dari pointer objek