# PRAKTIKUM 8

# LINKED LIST

## A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Stuktur Program menggunakan bahasa C++

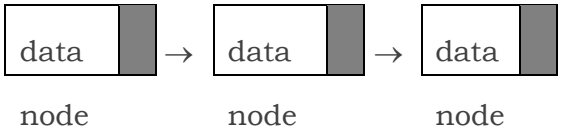
## B. Peralatan

1. PC Desktop
2. Windows 7
3. Notepad++ dan MinGW atau Dev C++

## C. Teori

Linked list sering disebut juga senarai berantai adalah struktur data yang terdiri dari sekelompok node (simpul) yang membentuk rangkaian secara runtut, sekuensial, saling sambung menyambung dan dinamis. Linked list saling terhubung dengan bantuan variabel pointer. Masing-masing data dalam Linked List disebut dengan node (simpul) yang menempati alokasi memori secara dinamis dan biasanya berupa struct yang terdiri dari beberapa field.

Linked list dalam bentuk yang sederhana setiap node terdiri dari 2 bagian, yaitu data dan penyambung ke node berikutnya. Analoginya mirip dengan rangkaian gerbong kereta api. Masing-masing gerbong itulah yang disebut struct/tipe data bentukan. Agar gerbong kereta itu saling bertautan maka dibutuhkan sebuah kait yang disebut sebagai pointer. Berikut ini Gambar 9.1 adalah ilustrasi dari linked list sederhana.




*Gambar 9. 1 Contoh linked list*

Gambar di atas adalah ilustrasi linked list sederhana di mana setiap node memiliki segmen untuk data dan satu segmen sebagai penghubung ke node di belakangnya.

Penggunaan linked list memiliki beberapa keuntungan dibandingkan dengan array konvensional, di antaranya:

1. Bersifat dinamis, pengalokasian memori hanya sesuai dengan yang dibutuhkan.
2. Operasi penambahan dan penghapusan dapat diimplementasikan dengan mudah.

3. Dapat diimplementasikan dengan mudah untuk struktur data linear seperti stack dan queue.

Selain kelebihan linked list juga memiliki beberapa kekurangan, diantaranya:

1. Penggunaan memori tambahan untuk pointer di setiap node.
2. Dalam linked list sederhana pembacaan hanya dapat dilakukan secara sekuensial dari depan ke belakang (forward list).
3. Pengaksesan node tidak bisa dilakukan secara acak.

Sebuah node pada linked list yang paling sederhana dapat didefinisikan menggunakan struct beranggotakan satu elemen dan satu pointer yang menunjuk ke node berikutnya. Berikut ini adalah contoh definisi struct Node untuk linked list dengan elemen bertipe int.

```
Node
{
        int elm;
        Node* next;
};
```

Struct Node berisi anggota elemen bertipe int sebagai representasi data yang tersimpan dalam linked list dan pointer next pertipe Node sebagai penghubung dengan node yang lain. Linked list dapat dibentuk dari beberapa pointer bertipe Node sebagai handle. Ada 2 handle yang dapat dibentuk, yaitu sebagai pemegang awal (head) rangkaian dan pemegang akhir (tail) rangkaian. Berikut ini adalah deklarasi untuk 2 handle tersebut:

```
Node* hd;     // head, handle awal
Node* tl;     // tail, handle akhir
size_t sz;    // counter jumlah node
```

Rangkaian linked list dapat dialokasikan secara dinamis dalam heap menggunakan 2 handle yang sudah dibentuk. Selain itu terdapat sebuah variabel sz bertipe size_t yang digunakan sebagai pencatat jumlah node dalam linked list. sz akan selalu berubah nilainya setiap kali terjadi operasi penambahan ataupun pengurangan node.
Sebelum digunakan linked list harus diinisialisasi terlebih dahulu. Inisialisasi sebaiknya dilakukan saat pendeklarasian, sehingga kode sebelumnya dapat dimodifikasi seperti berikut ini.

```
Node* hd{nullptr};   // head, handle awal
Node* tl{nullptr};   // tail, handle akhir
size_t sz{0};        // counter jumlah node
                     // inisialisasi dengan 0
```

Nilai diinisialisasi dengan 0 yang menyatakan bahwa dalam linked list belum terdapat rangkaian node sama sekali. Semua handle menunjuk ke nilai nullptr yang menyatakan bahwa ketiganya tidak menunjuk ke manapun.

## D. PRAKTIKUM

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

using namespace std;

typedef struct Data
{
    int nilai;
    Data *next;
};

Data *head;
Data *tail;

void awal ()
{
    head=NULL;
}

bool isEmpty()
{
    if (head==NULL)
        return true;
    return false;
}

void tambahDataDepan (int DataBaru)
{
    Data *baru;
    baru = new Data;
    baru -> nilai = DataBaru;
    baru -> next = NULL;
    if (isEmpty())
    {
        head = baru;
        head->next=NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
    cout << "Data Depan" << DataBaru << "Masuk"<< endl;
}
```

```
void tambahDataBelakang (int DataBaru)
{
    Data *baru, *bantu;
    baru = new Data;
    baru->nilai=DataBaru;
    baru->next = NULL;
    if (isEmpty())
    {
        head=baru;
        head->next=NULL;
    }
    else
    {
        bantu=head;
        while(bantu->next!=NULL)
        {
            bantu=bantu->next;
        }
        bantu->next=baru;
    }
    cout << "Data Belakang " << DataBaru << " Masuk"  << endl;
}

void hapusDepan ()
{
    Data *hapus;
    int d;
    if (!isEmpty())
    {
        if(head->next !=NULL)
        {
            hapus = head;
            d = hapus->nilai;
            head = hapus->next;
            delete hapus;
        }
        else
        {
            d = head->nilai;
            head = NULL;
        }
        cout << d << " Terhapus" << endl;
    }
    else cout << "Masih Kosong" << endl;
}

void hapusBelakang ()
{
    Data *hapus, *bantu;
    int tmp;
    if (!isEmpty())
    {
```

```
        if (head->next!=NULL)
        {
            bantu = head;
            while (bantu->next->next!=NULL)
            {
                bantu = bantu->next;
            }
            hapus = bantu->next;
            tmp = hapus->nilai;
            bantu->next=NULL;
            delete hapus;
        }
        else
        {
            tmp=head->nilai;
            head=NULL;
        }
        cout <<  tmp << " Terhapus" << endl;
    }
    else cout << "Masih Kosong" << endl;
}

void Cetak ()
{
    if (!isEmpty())
    {
        Data *bantu;
        bantu=head;
        do
        {
            cout << bantu->nilai<< " ";
            bantu=bantu->next;
        }
        while (bantu!=NULL);
        cout << endl;
    }
}

int panjang ()
{
    int count=0;
    if (!isEmpty())
    {
        count=1;
        Data *bantu;
        bantu=head;
        if (bantu->next==NULL)
        {
            count=1;
        }
        else
        {
            do
```

```
            {
                count++;
                bantu=bantu->next;
            }
            while (bantu->next!=NULL);
        }
    }
    else
    {
        count=0;
    }
    return count;
}

int main()
{
    awal();
    tambahDataBelakang(5);
    tambahDataDepan(7);
    tambahDataBelakang(17);
    tambahDataBelakang(1);
    tambahDataBelakang(27);
    tambahDataBelakang(10);
    cout << "Data pada single linked list non circular:" << endl;
    Cetak();
    cout << "Data paling depan dihapus:" << endl;
    hapusDepan();
    cout << "Data pada single linked list non circular:" << endl;
    Cetak();
    cout << "Data paling belakang dihapus:" << endl;
    hapusBelakang();
    cout << "Data pada single linked list non circlar:" << endl;
    Cetak();
    cout << "Panjang linked list :" << endl;
    cout << panjang();
    getch();
    return 0;
}
```

## E. STUDI KASUS

```cpp
#include <iostream>

using namespace std;

// deklarasi single linked list
struct Buku{

  // komponen / member
  string judul, pengarang;
  int tahunTerbit;

  Buku *next;

};

Buku *head, *tail, *cur, *newNode, *del, *before;

// create single linked list
void createSingleLinkedList(string judul, string pengarang, int tB){
  head = new Buku();
  head->judul = judul;
  head->pengarang = pengarang;
  head->tahunTerbit = tB;
  head->next = NULL;
  tail = head;
}


// print single linked list
int countSingleLinkedList(){
  cur = head;
  int jumlah = 0;
  while( cur != NULL ){
    jumlah++;
    cur = cur->next;
  }
  return jumlah;
}
```

```
40   // tambahAwal Single linked list
41  void addFirst(string judul, string pengarang, int tB){
42    newNode = new Buku();
43    newNode->judul = judul;
44    newNode->pengarang = pengarang;
45    newNode->tahunTerbit = tB;
46    newNode->next = head;
47    head = newNode;
48  }
49
50   // tambahAkhir Single linked list
51  void addLast(string judul, string pengarang, int tB){
52    newNode = new Buku();
53    newNode->judul = judul;
54    newNode->pengarang = pengarang;
55    newNode->tahunTerbit = tB;
56    newNode->next = NULL;
57    tail->next = newNode;
58    tail = newNode;
59  }
60
61   // tambah tengah single linked list
62  void addMiddle(string judul, string pengarang, int tB, int posisi){
63    if( posisi < 1 || posisi > countSingleLinkedList() ){
64      cout << "Posisi diluar jangkauan" << endl;
65    }else if( posisi == 1){
66      cout << "Posisi bukan posisi tengah" << endl;
67    }else{
68      newNode = new Buku();
69      newNode->judul = judul;
70      newNode->pengarang = pengarang;
71      newNode->tahunTerbit = tB;
72
73       // tranversing
74       cur = head;
75       int nomor = 1;
76       while( nomor < posisi - 1 ){
77         cur = cur->next;
78         nomor++;
79       }
80       newNode->next = cur->next;
81       cur->next = newNode;
82    }
83  }
84
85
86   // Remove First
87  void removeFirst(){
88    del = head;
89    head = head->next;
90    delete del;
91  }
92
93   // Remove Last
94  void removeLast(){
95    del = tail;
96    cur = head;
97    while( cur->next != tail ){
98      cur = cur->next;
99    }
100   tail = cur;
101   tail->next = NULL;
102   delete del;
103 }
```

```
// remove middle
void removeMiddle(int posisi){
  if( posisi < 1 || posisi > countSingleLinkedList() ){
    cout << "Posisi diluar jangkauan" << endl;
  }else if( posisi == 1){
    cout << "Posisi bukan posisi tengah" << endl;
  }else{
    int nomor = 1;
    cur = head;
    while( nomor <= posisi ){
      if( nomor == posisi-1 ){
        before = cur;
      }
      if( nomor == posisi ){
        del = cur;
      }
      cur = cur->next;
      nomor++;
    }
    before->next = cur;
    delete del;
  }
}

// ubahAwal Single linked list
void changeFirst(string judul, string pengarang, int tB){
  head->judul = judul;
  head->pengarang = pengarang;
  head->tahunTerbit = tB;
}

// ubahAkhir Single linked list
void changeLast(string judul, string pengarang, int tB){
  tail->judul = judul;
  tail->pengarang = pengarang;
  tail->tahunTerbit = tB;
}
```

```
// ubah Tengah Single linked list
void changeMiddle(string judul, string pengarang, int tB, int posisi){
  if( posisi < 1 || posisi > countSingleLinkedList() ){
    cout << "Posisi diluar jangkauan" << endl;
  }else if( posisi == 1 || posisi == countSingleLinkedList() ){
    cout << "Posisi bukan posisi tengah" << endl;
  }else{
    cur = head;
    int nomor = 1;
    while( nomor < posisi ){
      cur = cur->next;
      nomor++;
    }
    cur->judul = judul;
    cur->pengarang = pengarang;
    cur->tahunTerbit = tB;
  }
}

// print single linked list
void printSingleLinkedList(){
  cout << "Jumlah data ada : " << countSingleLinkedList() << endl;
  cur = head;
  while( cur != NULL ){
    cout << "Judul Buku : " << cur->judul << endl;
    cout << "Pengarang Buku : " << cur->pengarang << endl;
    cout << "Tahun Terbit Buku : " << cur->tahunTerbit << endl;

    cur = cur->next;
  }
}
```

```cpp
int main(){

    createSingleLinkedList("Kata", "Geez & Aan", 2018);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    addFirst("Dia adalah Kakakku", "Tere Liye", 2009);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    addLast("Aroma Karsa", "Dee Lestari", 2018);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    removeFirst();

    printSingleLinkedList();

    cout << "\n\n" << endl;

    addLast("11.11", "Fiersa Besari", 2018);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    removeLast();

    printSingleLinkedList();

    cout << "\n\n" << endl;

    changeFirst("Berhenti di Kamu", "Gia Pratama", 2018);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    addMiddle("Bumi Manusia", "Pramoedya Anata Toer", 2005, 2);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    addMiddle("Negeri 5 Menara", "Ahmad Fuadi", 2009, 2);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    removeMiddle(5);

    printSingleLinkedList();

    cout << "\n\n" << endl;

    changeMiddle("Sang Pemimpi", "Andrea Hirata", 2006, 2);

    printSingleLinkedList();

    cout << "\n\n" << endl;

}
```

**Latihan**

Note: kerjakan semua percobaan diatas baru anda mengerjakan tugas dibawah ini:

1. Lakukan revisi pada source code singly linked list non circular supaya menjadi linked list yang circular.
2. Tambahkan fungsi untuk pencarian buku berdasarkan judul dan pencarian buku berdasarkan nama pengarang.
3. Tambahkan fungsi pengurutan buku berdasarkan tahun terbit.