LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AWAL PRAKTIKUM

Pertemuan ke-08

Sorting Lanjut-2

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Sorting adalah proses mengatur atau menyusun elemen-elemen dalam satu kumpulan data dengan aturan tertentu. Dalam program C++, terdapat beberapa algoritma sorting yang digunakan untuk mengurutkan array atau struktur data lainnya, seperti Bubble Sort, Selection Sort, dan Insertion Sort. Keuntungan menggunakan algoritma sorting yaitu menghemat waktu dan memudahkan dalam mencari nilai tertentu pada suatu array yang sudah terurut.

Shell Sort adalah algoritma sorting yang menggunakan konsep pembagian data menjadi beberapa bagian yang lebih kecil untuk dilakukan sorting secara terpisah. Algoritma ini menggunakan pendekatan incremental, di mana elemen-elemen yang berjarak jauh dibandingkan dan ditukar jika diperlukan. Proses ini terus berlanjut dengan mengurangi jarak antar elemen hingga mencapai jarak 1. Setelah itu, algoritma ini beralih ke penggunaan Insertion Sort untuk menyelesaikan proses sorting. Shell Sort memiliki kompleksitas waktu yang lebih baik daripada Insertion Sort, tetapi tidak secepat algoritma sorting yang lebih efisien seperti Quick Sort atau Merge Sort.

Insertion Sort adalah algoritma sorting sederhana yang bekerja dengan membagi data menjadi dua bagian: bagian yang sudah diurutkan dan bagian yang belum diurutkan. Algoritma ini memilih satu elemen dari bagian belum diurutkan dan memasukkannya ke posisi yang tepat di bagian yang sudah diurutkan. Proses ini diulang untuk setiap elemen dalam bagian belum diurutkan hingga seluruh data terurut. Insertion Sort memiliki kompleksitas waktu yang lebih baik daripada algoritma sorting seperti Bubble Sort atau Selection Sort untuk jumlah data yang kecil atau hampir terurut.

B. TUGAS PENDAHULUAN

1. Jelaskan kekurangan menggunakan metode Shell Sort dan Insertion Sort dengan metode-metode Sorting lainnya!

   Jawab :

   Kekurangan metode Shell Sort adalah membutuhkan method tambahan dan sulit untuk membagi masalah . Sedangkan kekurangan metode Insertion Sort adalah untuk larik yang jumlahnya besar tidak praktis.

2. Jelaskan perbedaan program Sorting dengan menggunakan antara metode Shell Sort dan Insertion Sort

   Jawab :

   Metode Shell Sort ini mengurutkan data dengan cara membandingkan suatu data dengan data lain yang meiliki jarak tertentu, kemudian dilakukan penukaran bila diperlukan. Perbedaannya dengan metode Insertion Sort ini memilih elemen dengan nilai paling rendah dan menukar dengan elemen terpilih dengan elemen ke-i. Nilai dari i dimulai dari 1 ke n, dimana n adalah jumlah total elemen yang dikurangi 1.

3. Jelaskan tahapan-tahapan Sorting menggunakan metode Shell Sort!

   Jawab :

   Jarak ditentukan dengan nDiv 2, dimana n adalah banyaknya elemen array. Lakukan pertukaran tempat jika setiap kali perbandingan dipenuhi (lebih besar untuk menaik dan lebih kecil untuk urut menurun). Setiap kali perbandingan terhadap keseluruhan elemen selesai dilakukan, maka perbandingan yang baru dilakukan kembali dimana jarak diperoleh dengan jarak div 2 (jarak diperoleh dari nilai jarak sebelumnya).

4. Jelaskan tahapan-tahapan Sorting menggunakan metode Insertion Sort!

   Jawab :

   Jarak ditentukan dengan nDiv 2, dimana n adalah banyaknya elemen array. Lakukan pertukaran tempat jika setiap kali perbandingan dipenuhi (lebih besar untuk menaik dan lebih kecil untuk urut menurun). Setiap kali perbandingan terhadap keseluruhan elemen selesai dilakukan, maka perbandingan yang baru dilakukan kembali dimana jarak diperoleh dengan jarak div 2 (jarak diperoleh dari nilai jarak sebelumnya).

C. TUGAS PRAKTIKUM

- Lat9_1

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
 cout << "Nama \t: Nova Ardiansyah\n";
 cout << "NIM \t: 211011401309\n";
 cout << "=========================\n\n";

 int Nilai[20];
 int i, k, N, l;
 int temp, jarak, s;

 cout << "Masukkan Banyak Bilangan : ";
 cin >> N;

 for (i = 0; i < N; i++)
 {
  cout << "Elemen Ke-" << i << " : ";
  cin >> Nilai[i];
 }

 cout << "\nData Sebelum diurutkan : ";
 for (i = 0; i < N; i++)
 {
  cout << Nilai[i] << " ";
 }

 cout << "\n\nMetode Shell Sort (Menaik) : ";

 jarak = N / 2;
 cout << "\nJarak = " << jarak;

 while (jarak >= 1) {
  do {
   s = 0;

   for (i = 0; i <= (N - jarak) - 1; i++) {
    k = i + jarak;

    if (Nilai[i] > Nilai[k]) {
     temp = Nilai[i];
     Nilai[i] = Nilai[k];
     Nilai[k] = temp;
     s = 1;

     for (l = 0; l < N; l++) {
      cout << Nilai[l] << " ";
     }
    }
   }
  }

  while (s != 0);
  jarak /= 2;
  cout << "\nJarak = " << jarak << "\n";
 }
```

```
  cout << "\n\nData Setelah diurutkan : ";
  for (i = 0; i < N; i++) {
    cout << Nilai[i] << " ";
  }

  return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Masukkan Banyak Bilangan : 3
Elemen Ke-0 : 23
Elemen Ke-1 : 12
Elemen Ke-2 : 32

Data Sebelum diurutkan : 23 12 32

Metode Shell Sort (Menaik) :
Jarak = 112 23 32
Jarak = 0


Data Setelah diurutkan : 12 23 32
--------------------------------
Process exited after 7.287 seconds with return value 0
Press any key to continue . . .
```

- Lat9_2

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  int Nilai[20];
  int i, k, N, l;
  int temp, jarak, s;

  cout << "Masukkan Banyak Bilangan : ";
  cin >> N;

  for (i = 0; i < N; i++)
  {
    cout << "Elemen Ke-" << i << " : ";
    cin >> Nilai[i];
  }

  cout << "\nData Sebelum diurutkan : ";
  for (i = 0; i < N; i++)
  {
    cout << Nilai[i] << " ";
  }

  cout << "\n\nMetode Shell Sort (Menurun) : ";

  jarak = N / 2;
  cout << "\nJarak = " << jarak;

  while (jarak >= 1) {
    do {
      s = 0;

      for (i = 0; i <= (N - jarak) - 1; i++) {
```

```
      k = i + jarak;

      if (Nilai[i] < Nilai[k]) {
        temp = Nilai[i];
        Nilai[i] = Nilai[k];
        Nilai[k] = temp;
        s = 1;

        for (l = 0; l < N; l++) {
          cout << Nilai[l] << " ";
        }
      }
    }
   }

   while (s != 0);
   jarak /= 2;
   cout << "\nJarak = " << jarak << "\n";
  }

  cout << "\n\nData Setelah diurutkan : ";
  for (i = 0; i < N; i++) {
   cout << Nilai[i] << " ";
  }

  return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Masukkan Banyak Bilangan : 3
Elemen Ke-0 : 43
Elemen Ke-1 : 23
Elemen Ke-2 : 45

Data Sebelum diurutkan : 43 23 45

Metode Shell Sort (Menaik) :
Jarak =  23 43 45
Jarak = 0


Data Setelah diurutkan : 23 43 45
--------------------------------
Process exited after 12.29 seconds with return value 0
Press any key to continue . . .
```

- Lat9_3

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  int Nilai[20];
  int i, j, k, N;
  int temp;

  cout << "Masukkan Banyak Bilangan : ";
  cin >> N;

  for (i = 0; i < N; i++)
  {
   cout << "Elemen Ke-" << i << " : ";
```

```
  cin >> Nilai[i];
 }

 cout << "\nData Sebelum diurutkan : ";
 for (i = 0; i < N; i++) {
  cout << Nilai[i] << " ";
 }

 cout << "\n\nMetode Insertion Sort (Menaik) : ";

 for (i = 0; i < N; i++) {
  temp = Nilai[i];
  j = i - 1;

  while ((temp <= Nilai[j]) && (j >= 1)) {
   Nilai[j + 1] = Nilai[j];
   j--;
  }

  if (temp >= Nilai[j]) {
   Nilai[j + 1] = temp;
  } else {
   Nilai[j + 1] = Nilai[j];
   Nilai[j] = temp;
  }
 }

 cout << "\n\nData Setelah diurutkan : ";
 for (i = 0; i < N; i++) {
  cout << Nilai[i] << " ";
 }

 return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Masukkan Banyak Bilangan : 3
Elemen Ke-0 : 43
Elemen Ke-1 : 12
Elemen Ke-2 : 5

Data Sebelum diurutkan : 43 12 5

Metode Insertion Sort (Menaik) :

Data Setelah diurutkan : 5 12 43
--------------------------------
Process exited after 6.35 seconds with return value 0
Press any key to continue . . .
```

- Lat9_4

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
 cout << "Nama \t: Nova Ardiansyah\n";
 cout << "NIM \t: 211011401309\n";
 cout << "========================\n\n";

 int Nilai[20];
 int i, j, k, N;
 int temp;

 cout << "Masukkan Banyak Bilangan : ";
```

```
  cin >> N;

  for (i = 0; i < N; i++)
  {
    cout << "Elemen Ke-" << i << " : ";
    cin >> Nilai[i];
  }

  cout << "\nData Sebelum diurutkan : ";
  for (i = 0; i < N; i++) {
    cout << Nilai[i] << " ";
  }

  cout << "\n\nMetode Insertion Sort (Menurun) : ";

  for (i = 0; i < N; i++) {
    temp = Nilai[i];
    j = i - 1;

    while ((temp > Nilai[j]) && (j >= 1)) {
      Nilai[j + 1] = Nilai[j];
      j--;
    }

    if (temp <= Nilai[j]) {
      Nilai[j + 1] = temp;
    } else {
      Nilai[j + 1] = Nilai[j];
      Nilai[j] = temp;
    }
  }

  cout << "\n\nData Setelah diurutkan : ";
  for (i = 0; i < N; i++) {
    cout << Nilai[i] << " ";
  }

  return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Masukkan Banyak Bilangan : 3
Elemen Ke-0 : 0
Elemen Ke-1 : 21
Elemen Ke-2 : 10

Data Sebelum diurutkan : 0 21 10

Metode Insertion Sort (Menurun) :

Data Setelah diurutkan : 21 10 0
--------------------------------
Process exited after 8.179 seconds with return value 0
Press any key to continue . . .
```

- Lat9_5

```
#include <iostream>
#include <string>
using namespace std;

void shellSortAsc()
{
  int Nilai[20];
  int i, k, N, l;
  int temp, jarak, s;
```

```
  cout << "Masukkan Banyak Bilangan : ";
  cin >> N;

  for (i = 0; i < N; i++)
  {
   cout << "Elemen Ke-" << i << " : ";
   cin >> Nilai[i];
  }

  cout << "\nData Sebelum diurutkan : ";
  for (i = 0; i < N; i++)
  {
   cout << Nilai[i] << " ";
  }

  cout << "\n\nMetode Shell Sort (Menaik) : ";

  jarak = N / 2;
  cout << "\nJarak = " << jarak;

  while (jarak >= 1) {
   do {
    s = 0;

    for (i = 0; i <= (N - jarak) - 1; i++) {
     k = i + jarak;

     if (Nilai[i] > Nilai[k]) {
      temp = Nilai[i];
      Nilai[i] = Nilai[k];
      Nilai[k] = temp;
      s = 1;

      for (l = 0; l < N; l++) {
       cout << Nilai[l] << " ";
      }
     }
    }
   }

   while (s != 0);
   jarak /= 2;
   cout << "\nJarak = " << jarak << "\n";
  }

  cout << "\n\nData Setelah diurutkan : ";
  for (i = 0; i < N; i++) {
   cout << Nilai[i] << " ";
  }
}

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  int pilihan;

  cout << "Pilih Metode Pengurutan : \n";
  cout << "1. Metode Shell Sort (Menaik) :\n";
  cout << "2. Metode Shell Sort (Menurun) :\n";
  cout << "3. Metode Insertion Sort (Menaik) :\n";
```

```
  cout << "4. Metode Insertion Sort (Menurun) :\n";

  cout << "\nPilihan : ";
  cin >> pilihan;

  switch (pilihan)
  {
    case 1:
      shellSortAsc();
      break;
    default:
      cout << "Pilihan tidak tersedia.";
      break;
  }

  return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Pilih Metode Pengurutan :
1. Metode Shell Sort (Menaik) :
2. Metode Shell Sort (Menurun) :
3. Metode Insertion Sort (Menaik) :
4. Metode Insertion Sort (Menurun) :

Pilihan : 1
Masukkan Banyak Bilangan :
3
Elemen Ke-0 : 10
Elemen Ke-1 : 6
Elemen Ke-2 : 2

Data Sebelum diurutkan : 10 6 2

Metode Shell Sort (Menaik) :
Jarak = 16 10 2 6 2 10 2 6 10
Jarak = 0


Data Setelah diurutkan : 2 6 10
--------------------------------
Process exited after 14.44 seconds with return value 0
Press any key to continue . . .
```

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-08

Sorting Lanjut-2

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

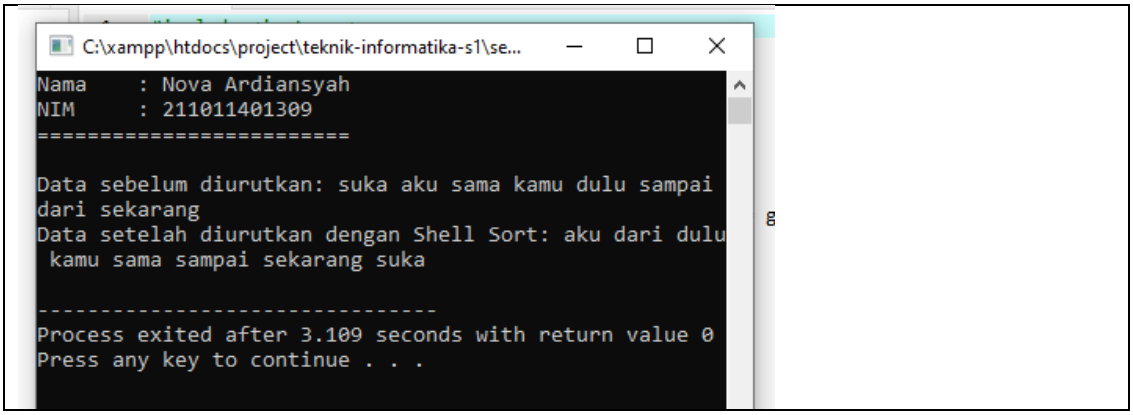
Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Buatlah program untuk mengurutkan sederetan data: suka, aku, sama, kamu, dulu, sampai, dari, sekarang. Dengan menggunakan salah satu metode Shell Sort dan Insertion Sort!

   Jawab :

```
#include <iostream>
using namespace std;

void shellSort(string arr[], int n) {
  for (int gap = n / 2; gap > 0; gap /= 2) {
    for (int i = gap; i < n; i++) {
      string temp = arr[i];
      int j;
      for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {
        arr[j] = arr[j - gap];
      }
      arr[j] = temp;
    }
  }
}

void printArray(string arr[], int n) {
  for (int i = 0; i < n; i++) {
    cout << arr[i] << " ";
  }
  cout << endl;
}

int main() {
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  const int n = 8;
  string data[n] = {"suka", "aku", "sama", "kamu", "dulu", "sampai", "dari", "sekarang"};

  cout << "Data sebelum diurutkan: ";
  printArray(data, n);

  shellSort(data, n);
  cout << "Data setelah diurutkan dengan Shell Sort: ";
  printArray(data, n);

  return 0;
}
```


C:\xampp\htdocs\project\teknik-informatika-s1\se...
Nama : Nova Ardiansyah
NIM : 211011401309
==========
Data sebelum diurutkan: suka aku sama kamu dulu sampai dari sekarang
Data setelah diurutkan dengan Shell Sort: aku dari dulu kamu sama sampai sekarang suka
---
Process exited after 3.109 seconds with return value 0
Press any key to continue . . .

2. KESIMPULAN

Secara singkatnya sorting adalah metode untuk pengurutan data. Secara garis besarnya, Sorting (Pengurutan) adalah suatu proses penyusunan kembali kumpulan objek menggunakan tata aturan tertentu. Sorting disebut juga sebagai suatu algoritma untuk meletakkan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen. Pengurutan atau sorting merupakan proses dasar yang ada dalam sebuah algoritma dan struktur data. Penggunaan algoritma sorting dapat pula diaplikasikan pada algoritma Python. Tujuan utama dari proses pengurutan atau sorting adalah untuk mengurutkan data berdasarkan keinginan baik itu dari yang terendah maupun yang tertinggi, sehingga data yang dihasilkan akan lebih terstruktur, teratur dan sesuai dengan kebutuhan.

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AWAL PRAKTIKUM

Pertemuan ke-09

Linked List

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Linked List adalah struktur berupa rangkaian elemen saling berkait dimana tiap elemen dihubungkan ke elemen yang lain melalui pointer. Pointer adalah alamat elemen. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logic walaupun tidak bersebelahan secara fisik di memori. Linked List merupakan kumpulan komponen yang saling berkaitan satu dengan yang lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau verteks.

Singly Linked List Merupakan Linked List yang paling sederhana. Setiap simpul dibagi menjadi dua bagian yaitu bagian isi dan bagian pointer. Bagian isi merupakan bagian yang berisi data yang disimpan oleh simpul, sedangkan bagian pointer merupakan bagian yang berisi alamat dari simpul berikutnya.

Doubly Linked List merupakan Linked List dimana setiap simpul dibagi menjadi tiga bagian, yaitu bagian isi, bagian pointer kiri, dan bagian pointer kanan. Bagian isi merupakan bagian yang berisi data yang disimpan oleh simpul, bagian pointer kiri merupakan bagian yang berisi alamat dari simpul sebelumnya dan bagian pointer kanan merupakan bagian yang berisi alamat dari simpul berikutnya. Deklarasi Doubly Linked List.

B. TUGAS PENDAHULUAN

1. Apa yang dimaksud dengan Linked List!
   Jawab :

   Linked List adalah struktur berupa rangkaian elemen saling berkait dimana tiap elemen dihubungkan ke elemen lain melalui pointer. Pointer juga merupakan alamat dari sebuah elemen.

2. Jelaskan perbedaan antara singly linked list, doubly linked list, dan circular linked list!
   Jawab :

Singly Linked List merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya, biasanya field pada tail menunjuk ke NULL.

Doubly Linked List merupakan suatu linked list yang memiliki dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya. Setiap head dan tailnya juga menunjuk ke NULL.

Circular Linked List merupakan suatu linked list dimana tail (node terakhir) menunjuk ke head (node pertama). Jadi tidak ada pointer yang menunjuk NULL.

3. Jelaskan operasi-operasi pada singly linked list!
   Jawab :

   - **Insert** Istilah Insert berarti menambahkan sebuah simpul baru ke dalam suatu linked list.
   - **Konstruktor** Fungsi ini membuat sebuah linked list yang baru dan masih kosong.
   - **IsEmpty** Fungsi ini menentukan apakah linked list kosong atau tidak.
   - **Find First** Fungsi ini mencari elemen pertama dari linked list.
   - **Find Next** Fungsi ini mencari elemen sesudah elemen yang ditunjuk now.
   - **Retrieve** Fungsi ini mengambil elemen yang ditunjuk oleh now. Elemen tersebut lalu dikembalikan oleh fungsi.
   - **Update** Fungsi ini mengubah elemen yang ditunjuk oleh now dengan isi dari sesuatu.
   - **Delete Now** Fungsi ini menghapus elemen yang ditunj uk oleh now. Jika yang dihapus adalah elemen pertama dari linked list (head), head akan berpindah ke elemen berikut.

4. Jelaskan operasi-operasi pada doubly linked list!
   Jawab :

- **Insert Tail** Fungsi insert tail berguna untuk menambah simpul di belakang (sebelah kanan) pada sebuah linked list.
- **Insert Head** Sesuai dengan namanya, fungsi Insert Head berguna untuk menambah simpul di depan (sebelah kiri). Fungsi ini tidak berada jauh dengan fungsi Insert Tail yang telah dijelaskan sebelumnya.
- **DeleteTail** Fungsi Delete Tail berguna untuk menghapus simpul dari belakang. Fungsi ini merupakan kebalikan dari fungsi Insert Tail yang menambah simpul dibelakang. Fungsi Delete Tail akan mengarahkan Now kepada Tail dan kemudian memanggil fungsi Delete Now.
- **Delete Head** Fungsi Delete Head merupakan kebalikan dari fungsi Delete Tail yang menghapus simpul dari belakang, sedangkan Delete Head akan menghapus simpul dari depan (sebelah kiri). Fungsi Delete Head akan mengarahkan Now kepada Head dan kemudian memanggil fungsi Delete Now.

C. TUGAS PRAKTIKUM

- Lat10_1

```
#include <iostream>
using namespace std;

struct node {
  char Isi;
  node* Next;
};

typedef node* simpul;

void Sisip_Depan(simpul& L, char elemen);
void Sisip_Belakang(simpul& L, char elemen);
void Sisip_Tengah1(simpul& L, char elemen1, char elemen2);
void Sisip_Tengah2(simpul& L, char elemen1, char elemen2);
void Hapus_Depan(simpul& L);
void Hapus_Belakang(simpul& L);
void Hapus_Tengah(simpul& L, char elemen);
void Cetak(simpul L);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  char huruf, huruf2;
  simpul L = NULL;

  cout << "=============== Operasi SLL ===============\n\n";

  cout << "1. Sisip Depan\n";
  cout << "==============\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Depan(L, huruf);

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Depan(L, huruf);
  Cetak(L);

  cout << "\n\n2. Sisip Belakang\n";
  cout << "=================\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);
  Cetak(L);

  cout << "\n\n3. Sisip setelah simpul tertentu\n";
  cout << "================================\n\n";

  cout << "Masukkan huruf : ";
```

```
  cin >> huruf;

  cout << "Masukkan huruf setelah huruf " << huruf << " : ";
  cin >> huruf2;
  Sisip_Tengah1(L, huruf, huruf2);
  Cetak(L);

  cout << "\n\n4. Sisip sebelum simpul tertentu\n";
  cout << "================================\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Tengah2(L, huruf, huruf2);
  Cetak(L);

  cout << "\n\n5. Hapus simpul depan\n";
  cout << "=====================\n\n";

  Hapus_Depan(L);
  Cetak(L);

  cout << "6. Hapus simpul belakang\n";
  cout << "========================\n\n";

  Hapus_Belakang(L);
  Cetak(L);

  cout << "\n\n7. Hapus simpul tertentu\n";
  cout << "========================\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;

  Hapus_Tengah(L, huruf);
  Cetak(L);

  return 0;
}

void Sisip_Depan(simpul& L, char elemen)
{
  simpul baru = new node;
  baru->Isi = elemen;
  baru->Next = L;

  L = baru;
}

void Sisip_Tengah1(simpul& L, char elemen1, char elemen2)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul bantu = L;
  while (bantu != NULL && bantu->Isi != elemen2)
  {
    bantu = bantu->Next;
  }

  if (bantu == NULL)
```

```
  {
   cout << "Simpul dengan elemen " << elemen2 << " tidak ditemukan\n";
   return;
  }

  simpul baru = new node;
  baru->Isi = elemen1;
  baru->Next = bantu->Next;

  bantu->Next = baru;
}

void Sisip_Tengah2(simpul& L, char elemen1, char elemen2)
{
  if (L == NULL)
  {
   cout << "List kosong\n";
   return;
  }

  if (L->Isi == elemen2)
  {
   Sisip_Depan(L, elemen1);
   return;
  }

  simpul bantu = L;
  while (bantu->Next != NULL && bantu->Next->Isi != elemen2)
  {
   bantu = bantu->Next;
  }

  if (bantu->Next == NULL)
  {
   cout << "Simpul dengan elemen " << elemen2 << " tidak ditemukan\n";
   return;
  }

  simpul baru = new node;
  baru->Isi = elemen1;
  baru->Next = bantu->Next;

  bantu->Next = baru;
}

void Sisip_Belakang(simpul& L, char elemen)
{
  simpul baru = new node;
  baru->Isi = elemen;
  baru->Next = NULL;

  if (L == NULL)
  {
   L = baru;
  }
  else
  {
   simpul bantu = L;
   while (bantu->Next != NULL)
   {
    bantu = bantu->Next;
   }
   bantu->Next = baru;
```

```
  }
}

void Cetak(simpul L)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
  }
  else
  {
    simpul bantu = L;
    while (bantu != NULL)
    {
      cout << bantu->Isi << "-->";
      bantu = bantu->Next;
    }
    cout << endl;
  }
}

void Hapus_Depan(simpul& L)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul hapus = L;
  L = L->Next;
  delete hapus;
}

void Hapus_Belakang(simpul& L)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  if (L->Next == NULL)
  {
    delete L;
    L = NULL;
    return;
  }

  simpul bantu = L;
  while (bantu->Next->Next != NULL)
  {
    bantu = bantu->Next;
  }

  delete bantu->Next;
  bantu->Next = NULL;
}

void Hapus_Tengah(simpul& L, char elemen)
{
  if (L == NULL)
  {
```

```
  cout << "List kosong\n";
  return;
 }

 if (L->Isi == elemen)
 {
  Hapus_Depan(L);
  return;
 }

 simpul bantu = L;
 while (bantu->Next != NULL && bantu->Next->Isi != elemen)
 {
  bantu = bantu->Next;
 }

 if (bantu->Next == NULL)
 {
  cout << "Simpul dengan elemen " << elemen << " tidak ditemukan\n";
  return;
 }

 simpul hapus = bantu->Next;
 bantu->Next = hapus->Next;
 delete hapus;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
=========================

=============== Operasi SLL ===============

1. Sisip Depan
==============

Masukkan huruf : a
Masukkan huruf : b
b-->a-->


2. Sisip Belakang
=================

Masukkan huruf : c
Masukkan huruf : d
b-->a-->c-->d-->

3. Sisip setelah simpul tertentu
================================

Masukkan huruf : e
Masukkan huruf setelah huruf e : a
b-->a-->e-->c-->d-->


4. Sisip sebelum simpul tertentu
================================

Masukkan huruf : f
b-->f-->a-->e-->c-->d-->


5. Hapus simpul depan
=====================
```

```
6. Hapus simpul belakang
========================

f-->a-->e-->c-->


7. Hapus simpul tertentu
========================

Masukkan huruf : e
f-->a-->c-->

--------------------------------
Process exited after 25.16 seconds with return value 0
Press any key to continue . . .
```

- Lat10_2

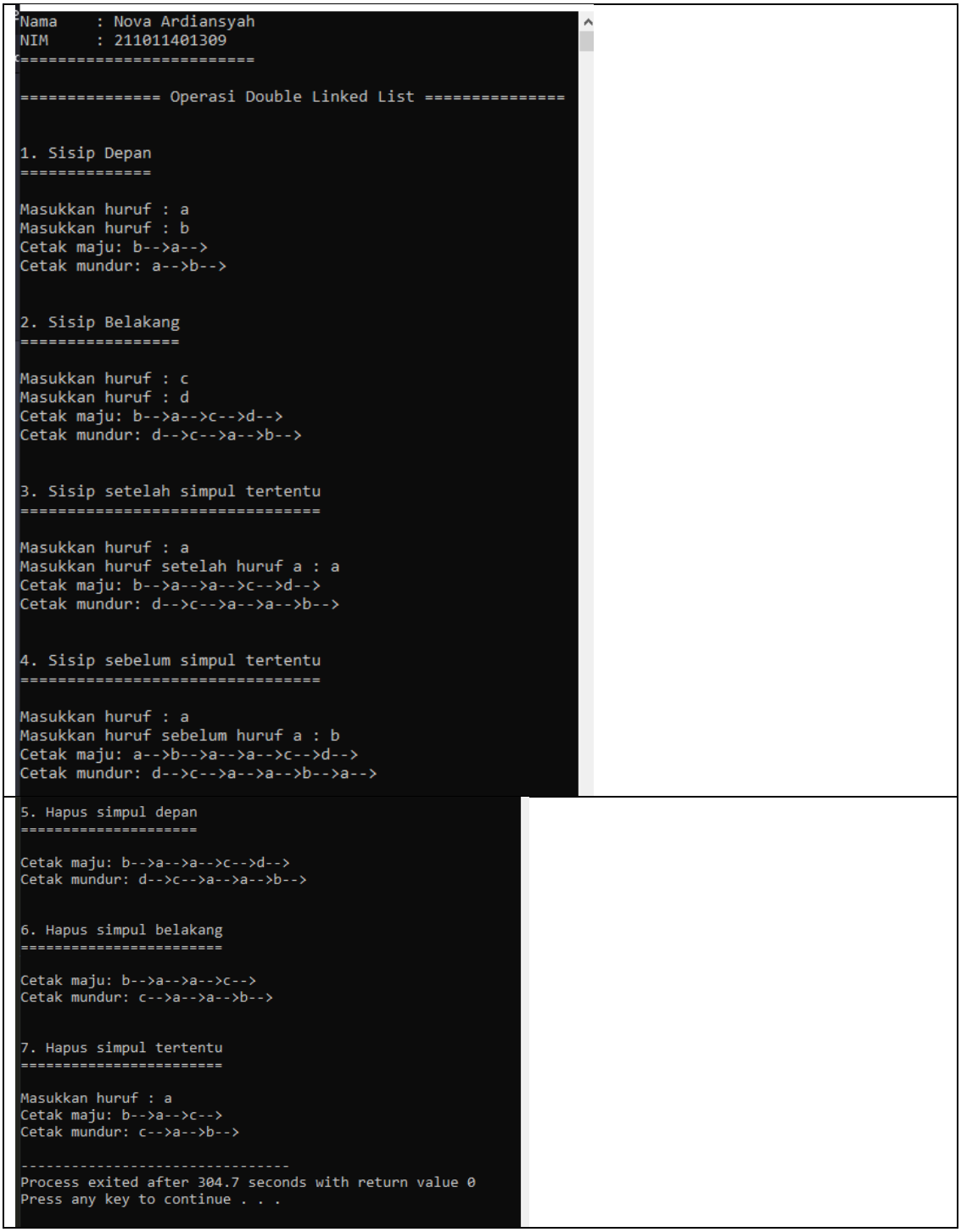
```
#include <iostream>
using namespace std;

struct node {
  char Isi;
  node* Prev;
  node* Next;
};

typedef node* simpul;

void Sisip_Depan(simpul& L, char elemen);
void Sisip_Belakang(simpul& L, char elemen);
void Sisip_Tengah1(simpul& L, char elemen1, char elemen2);
void Sisip_Tengah2(simpul& L, char elemen1, char elemen2);
void Hapus_Depan(simpul& L);
void Hapus_Belakang(simpul& L);
void Hapus_Tengah(simpul& L, char elemen);
void Cetak_Maju(simpul L);
void Cetak_Mundur(simpul L);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  char huruf, huruf2;
  simpul L = NULL;

  cout << "=============== Operasi Double Linked List ===============\n";

  cout << "\n\n1. Sisip Depan\n";
  cout << "==============\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Depan(L, huruf);

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Depan(L, huruf);
  Cetak_Maju(L);
  Cetak_Mundur(L);

  cout << "\n\n2. Sisip Belakang\n";
  cout << "=================\n\n";
```

```
  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);

  cout << "Masukkan huruf : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);

  Cetak_Maju(L);
  Cetak_Mundur(L);

  cout << "\n\n3. Sisip setelah simpul tertentu\n";
  cout << "================================\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;

  cout << "Masukkan huruf setelah huruf " << huruf << " : ";
  cin >> huruf2;
  Sisip_Tengah1(L, huruf, huruf2);

  Cetak_Maju(L);
  Cetak_Mundur(L);

  cout << "\n\n4. Sisip sebelum simpul tertentu\n";
  cout << "================================\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;
  cout << "Masukkan huruf sebelum huruf " << huruf << " : ";
  cin >> huruf2;
  Sisip_Tengah2(L, huruf, huruf2);

  Cetak_Maju(L);
  Cetak_Mundur(L);

  cout << "\n\n5. Hapus simpul depan\n";
  cout << "=====================\n\n";

  Hapus_Depan(L);
  Cetak_Maju(L);
  Cetak_Mundur(L);

  cout << "\n\n6. Hapus simpul belakang\n";
  cout << "========================\n\n";

  Hapus_Belakang(L);
  Cetak_Maju(L);
  Cetak_Mundur(L);

  cout << "\n\n7. Hapus simpul tertentu\n";
  cout << "========================\n\n";

  cout << "Masukkan huruf : ";
  cin >> huruf;

  Hapus_Tengah(L, huruf);
  Cetak_Maju(L);
  Cetak_Mundur(L);

  return 0;
}
```

```
// FUNCTION

void Sisip_Depan(simpul& L, char elemen)
{
  simpul baru = new node;
  baru->Isi = elemen;
  baru->Prev = NULL;
  baru->Next = L;

  if (L != NULL)
    L->Prev = baru;

  L = baru;
}

void Sisip_Tengah1(simpul& L, char elemen1, char elemen2)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul bantu = L;
  while (bantu != NULL && bantu->Isi != elemen2)
    bantu = bantu->Next;

  if (bantu == NULL)
  {
    cout << "Simpul dengan elemen " << elemen2 << " tidak ditemukan\n";
    return;
  }

  simpul baru = new node;
  baru->Isi = elemen1;
  baru->Prev = bantu;
  baru->Next = bantu->Next;

  if (bantu->Next != NULL)
    bantu->Next->Prev = baru;

  bantu->Next = baru;
}

void Sisip_Tengah2(simpul& L, char elemen1, char elemen2)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul bantu = L;
  while (bantu != NULL && bantu->Isi != elemen2)
    bantu = bantu->Next;

  if (bantu == NULL)
  {
    cout << "Simpul dengan elemen " << elemen2 << " tidak ditemukan\n";
    return;
  }

  simpul baru = new node;
```

```
  baru->Isi = elemen1;
  baru->Prev = bantu->Prev;
  baru->Next = bantu;

  if (bantu->Prev != NULL)
    bantu->Prev->Next = baru;
  else
    L = baru;

  bantu->Prev = baru;
}

void Sisip_Belakang(simpul& L, char elemen)
{
  simpul baru = new node;
  baru->Isi = elemen;
  baru->Prev = NULL;
  baru->Next = NULL;

  if (L == NULL)
  {
    L = baru;
    return;
  }

  simpul bantu = L;
  while (bantu->Next != NULL)
    bantu = bantu->Next;

  bantu->Next = baru;
  baru->Prev = bantu;
}

void Hapus_Depan(simpul& L)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul hapus = L;
  L = L->Next;

  if (L != NULL)
    L->Prev = NULL;

  delete hapus;
}

void Hapus_Belakang(simpul& L)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul bantu = L;
  while (bantu->Next != NULL)
    bantu = bantu->Next;

  if (bantu->Prev != NULL)
```

```
    bantu->Prev->Next = NULL;
  else
    L = NULL;

  delete bantu;
}

void Hapus_Tengah(simpul& L, char elemen)
{
  if (L == NULL)
  {
    cout << "List kosong\n";
    return;
  }

  simpul bantu = L;
  while (bantu != NULL && bantu->Isi != elemen)
    bantu = bantu->Next;

  if (bantu == NULL)
  {
    cout << "Simpul dengan elemen " << elemen << " tidak ditemukan\n";
    return;
  }

  if (bantu->Prev != NULL)
    bantu->Prev->Next = bantu->Next;
  else
    L = bantu->Next;

  if (bantu->Next != NULL)
    bantu->Next->Prev = bantu->Prev;

  delete bantu;
}

void Cetak_Maju(simpul L)
{
  cout << "Cetak maju: ";
  simpul bantu = L;
  while (bantu != NULL)
  {
    cout << bantu->Isi << "-->";
    bantu = bantu->Next;
  }
  cout << endl;
}

void Cetak_Mundur(simpul L)
{
  cout << "Cetak mundur: ";
  simpul bantu = L;
  while (bantu != NULL && bantu->Next != NULL)
    bantu = bantu->Next;

  while (bantu != NULL)
  {
    cout << bantu->Isi << "-->";
    bantu = bantu->Prev;
  }
  cout << endl;
}
```



```
Nama    : Nova Ardiansyah
NIM     : 211011401309
=========================

=============== Operasi Double Linked List ===============


1. Sisip Depan
==============

Masukkan huruf : a
Masukkan huruf : b
Cetak maju: b-->a-->
Cetak mundur: a-->b-->


2. Sisip Belakang
=================

Masukkan huruf : c
Masukkan huruf : d
Cetak maju: b-->a-->c-->d-->
Cetak mundur: d-->c-->a-->b-->


3. Sisip setelah simpul tertentu
================================

Masukkan huruf : a
Masukkan huruf setelah huruf a : a
Cetak maju: b-->a-->a-->c-->d-->
Cetak mundur: d-->c-->a-->a-->b-->


4. Sisip sebelum simpul tertentu
================================

Masukkan huruf : a
Masukkan huruf sebelum huruf a : b
Cetak maju: a-->b-->a-->a-->c-->d-->
Cetak mundur: d-->c-->a-->a-->b-->a-->


5. Hapus simpul depan
=====================

Cetak maju: b-->a-->a-->c-->d-->
Cetak mundur: d-->c-->a-->a-->b-->


6. Hapus simpul belakang
========================

Cetak maju: b-->a-->a-->c-->
Cetak mundur: c-->a-->a-->b-->


7. Hapus simpul tertentu
========================

Masukkan huruf : a
Cetak maju: b-->a-->c-->
Cetak mundur: c-->a-->b-->

--------------------------------
Process exited after 304.7 seconds with return value 0
Press any key to continue . . .
```

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-09

Linked List

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Buatlah Program Menu untuk menampilkan program diatas!

   Jawab :

```
#include<iostream>
#include<cstdlib>

using namespace std;
typedef struct node *simpul;

struct node
{
  char isi;
  simpul next;
};

void sisipDepan (simpul &l, char elemen);
void sisipBelakang(simpul &l, char elemen);
void sisipTengah1 (simpul &l, char elemen1, char elemen2);
void sisipTengah2 (simpul &l, char elemen1, char elemen2);
void hapusDepan (simpul &l);
void hapusBelakang(simpul &l);
void hapusTengah (simpul &l, char elemen);
void cetak (simpul l);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  char huruf, huruf2;
  simpul l = NULL;
  int menu;
  cout << "OPERASI PADA SINGLE LINKED LIST" << endl << endl;

    do {
      cout << "Menu : " << endl;
      cout << "1. sisip depan" << endl;
      cout << "2. sisip belakang" << endl;
      cout << "3. sisip tengah1" << endl;
      cout << "4. sisip tengah2" << endl;
      cout << "5. hapus depan" << endl;
      cout << "6. hapus belakang" << endl;
      cout << "7. hapus tengah" << endl;
      cout << "8. cetak" << endl;
      cout << "9. keluar" << endl;

      cout << "Pilih menu : ";
      cin     >> menu;
      cout << endl;

      switch(menu) {
        case 1 :
          cout << "## Masukan huruf : ";
          cin >> huruf;
          sisipDepan(l, huruf);
          cout << endl;
          break;
```

```
      case 2 :
       cout << "## Masukan huruf : ";
       cin >> huruf;
       sisipBelakang(l, huruf);
       cout << endl;
       break;
      case 3 :
       cout << "## Masukan huruf : ";
       cin >> huruf; cout << endl;
       cout << "## Disisip setelah huruf : ";
       cin  >> huruf2; cout << endl;
       sisipTengah1(l, huruf, huruf2);
       break;
      case 4 :
       cout << "## Masukan huruf : ";
       cin  >> huruf; cout << endl;
       cout << "## Disisip sebelum huruf : ";
       cin  >> huruf2; cout << endl;
       sisipTengah2(l, huruf, huruf2);
       break;
      case 5 :
       hapusDepan(l);
       cout << "## Simpul depan dihapus" << endl << endl;
       break;
      case 6 :
       hapusBelakang(l);
       cout << "## Simpul belakang dihapus" << endl << endl;
       break;
      case 7 :
       cout << "## Masukan huruf tengah yang akan dihapus : ";
       cin >> huruf;
       hapusTengah(l, huruf);
       cout << endl;
       break;
      case 8 :
       cetak(l);
       cout << endl << endl;
       break;
      case 9 :
       cout << "## Keluar program..." << endl;
       break;
     default :
      cout << "## kode salah, coba lagi" << endl << endl;
      break;
     }
   }

  while(menu != 9);
  return 0;
}

void sisipDepan (simpul &l, char elemen)
{
   simpul baru;
   baru = (simpul) malloc(sizeof(simpul));
   baru->isi = elemen;
   baru-> next = NULL;
   if(l == NULL)
      l = baru;
   else
   {
      baru->next = l;
      l = baru;
```

```
   }
}

//fungsi sisip setelah simpul tertentu
void sisipTengah1 (simpul &l, char elemen1, char elemen2)
{
   simpul bantu, baru;
   baru = (simpul) malloc(sizeof(simpul));
   baru->isi = elemen1;
   baru->next = NULL;
   if(l == NULL)
      cout << "List kosong....." << endl;
   else
   {
      bantu = l;
      while(bantu -> isi != elemen2) bantu = bantu -> next;
      baru -> next = bantu -> next;
      bantu -> next = baru;
   }

}

//fungsi sisip simpul sebelum simpul tertentu
void sisipTengah2 (simpul &l, char elemen1, char elemen2)
{
   simpul bantu, baru;
   baru = (simpul) malloc(sizeof(simpul));
   baru -> isi = elemen1;
   baru -> next = NULL;
   if(l == NULL)
      cout << "list kosong....." << endl;
   else
   {
      bantu = l;
      while(bantu -> isi != elemen2) bantu = bantu -> next;
      baru -> next = bantu -> next;
      bantu -> next = baru;
   }
}

//fungsi simpul di belakang
void sisipBelakang(simpul &l, char elemen)
{
   simpul bantu, baru;
   baru = (simpul) malloc(sizeof(simpul));
   baru -> isi = elemen;
   baru -> next = NULL;
   if(l == NULL)
      l = baru;
   else
   {
      bantu = l;
      while(bantu -> next != NULL) bantu = bantu -> next;
      bantu -> next = baru;
   }
}

//fungis mencetak isi liked list
void cetak(simpul l)
{
   simpul bantu;
   if(l == NULL)
      cout << "Linked list kosong....." << endl;
```

```
    else
    {
       bantu = l;
       cout << "isi linked list : ";
       while(bantu -> next != NULL)
       {
          cout << bantu -> isi << "-->";
          bantu = bantu -> next;
       }
       cout << bantu -> isi;
    }
}

//funsi hapus simpul depan
void hapusDepan(simpul &l)
{
    simpul hapus;
    if(l == NULL)
       cout << "linked list kosong....." << endl;
    else
    {
       hapus = l;
       l = l -> next;
       hapus -> next == NULL;
       free(hapus);
    }
}

//fungsi hapus simpul belakang
void hapusBelakang(simpul &l)
{
  simpul bantu, hapus;
  if (l == NULL) {
    cout << "linked list kosong..... " << endl;
  } else {
    bantu = l;
    while(bantu -> next -> next != NULL) bantu = bantu -> next;
    hapus = bantu -> next;
    bantu -> next = NULL;
    free(hapus);
  }
}

void hapusTengah(simpul &l, char elemen)
{
    simpul bantu, hapus;
    if (l == NULL) {
      cout << "Linked list kosong.....";
    } else {
      bantu = l;
      while(bantu -> next -> isi != elemen) bantu = bantu -> next;
      hapus = bantu -> next;
      bantu -> next = bantu -> next -> next;
      hapus -> next = NULL;
      free(hapus);
    }
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

OPERASI PADA SINGLE LINKED LIST

Menu :
1. sisip depan
2. sisip belakang
3. sisip tengah1
4. sisip tengah2
5. hapus depan
6. hapus belakang
7. hapus tengah
8. cetak
9. keluar
Pilih menu : 1

## Masukan huruf : a

Menu :
1. sisip depan
2. sisip belakang
3. sisip tengah1
4. sisip tengah2
5. hapus depan
6. hapus belakang
7. hapus tengah
8. cetak
9. keluar
Pilih menu : 8

isi linked list : a
```

2. KESIMPULAN

Linked list adalah strukur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AWAL PRAKTIKUM

Pertemuan ke-10

Linked List (Lanjut)

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Linked List adalah struktur berupa rangkaian elemen saling berkait dimana tiap elemen dihubungkan ke elemen yang lain melalui pointer. Pointer adalah alamat elemen. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logic walaupun tidak bersebelahan secara fisik di memori. Linked List merupakan kumpulan komponen yang saling berkaitan satu dengan yang lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau verteks.

Singly Linked List Merupakan Linked List yang paling sederhana. Setiap simpul dibagi menjadi dua bagian yaitu bagian isi dan bagian pointer. Bagian isi merupakan bagian yang berisi data yang disimpan oleh simpul, sedangkan bagian pointer merupakan bagian yang berisi alamat dari simpul berikutnya.

Doubly Linked List merupakan Linked List dimana setiap simpul dibagi menjadi tiga bagian, yaitu bagian isi, bagian pointer kiri, dan bagian pointer kanan. Bagian isi merupakan bagian yang berisi data yang disimpan oleh simpul, bagian pointer kiri merupakan bagian yang berisi alamat dari simpul sebelumnya dan bagian pointer kanan merupakan bagian yang berisi alamat dari simpul berikutnya. Deklarasi Doubly Linked List.

B. TUGAS PENDAHULUAN

1. Jelaskan Fungsi antara Singly Linked List, Doubly dan Linked List!
   Jawab :

   Sinkly Linked List menggunakan setiap simpul dibagi menjadi 2 bagian yaitu bagian isi dan bagian pointer. Doubly Liked List menggunakan setiap simpul dibagi menjadi 3 bagian, yaitu bagian isi, bagian pointer kiri ddan bagian pointer kanan. Sedangkan Circular Linked List menggunakan linked list yang tidak memiliki nili nil/NULL untuk medan sambungannya.

2. Jelaskan Operasi-Operasi pada Circular Linked List!
   Jawab :

Simpul depan Simpul yang disisipkan selalu berada di posisi depan dari Linked List (CL). Misalkan kita memiliki suatu Linked List CL dengan empat simpul di mana masing-masing berisi informasi A, B, C, dan D, dan kita juga memiliki suatu simpul baru yang berisi informasi E yang akan disisipkan di posisi depan dari Linked List CL. Langkah-langkah penyisipan simpul baru dapat dilakukan Simpul belakang Penyisipan belakang sebenarnya hampir sama dengan penyisipan depan. Yang membedakan hanyalah pemindahan CL ke simpul yang ditunjuk oleh baru untuk penyisipan belakang tidak ada.

Walaupun dalam penyisipan depan juga tidak harus dilakukan pemindahan CL, karena dalam Circular semua simpul berhak menjadi depan. Langkah-langkah penyisipan simpul baru dapat dilakukan sebagai berikut: Simpul tengah Dalam melakukan penyisipan simpul tengah, maka Linked List harus dijamin tidak boleh kosong. Penyisipan tengah dapat dilakukan pada sebelum atau setelah simpul tertentu. Yang akan dilakukan di sini adalah hanya penyisipan simpul setelah simpul tertentu. Penghapusan Simpul penghapusan simpul adalah operasi penghapusan simpul dari Linked List.

Jika kita akan melakukan penghapusan simpul maka pastikan bahwa Linked List tidak boleh kosong. Penghapusan pada Circular Singly Linked List juga dapat dilakukan pada simpul yang berada di posisi depan, posisi tengah, maupun yang berada pada posisi belakang..

3. Jelaskan Operasi-Operasi pada Doubly Linked List!
   Jawab :

   - **Insert First** Penyisipan di awal list, sehingga pointer head juga akan berpindah ke elemen baru.
   - **Insert Last** Penyisipan di akhir list, sehingga pointer tail juga akan berpindah ke elemen baru
   - **Insert After / Before** Penyisipan after/before kurang lebih sama satu sama lain. Pada kasus diatas berlaku juga insert before 3

- **Delete First** Penghapusan di awal list, pointer head akan berpindah ke node selanjutnya, sementara node awal akan di dealokasi.
- **Delete Last** Penghapusan di akhir list, pointer tail akan berpindah ke node sebelumnya,sementara node akhir akan di dealokasi.
- **Delete Node** Penghapusan node dengan data tertentu, pada kasus diatas yaitu delete node 2.

4. Jelaskan Jenis Program Yang menggunakan Linked List!

   Jawab :

   Linked list dapat dibagi ke dalam 4 jenis, yakni: Singly linked list, Doubly linked list, Circular linked list, dan Circular doubly linked list.

C. TUGAS PRAKTIKUM

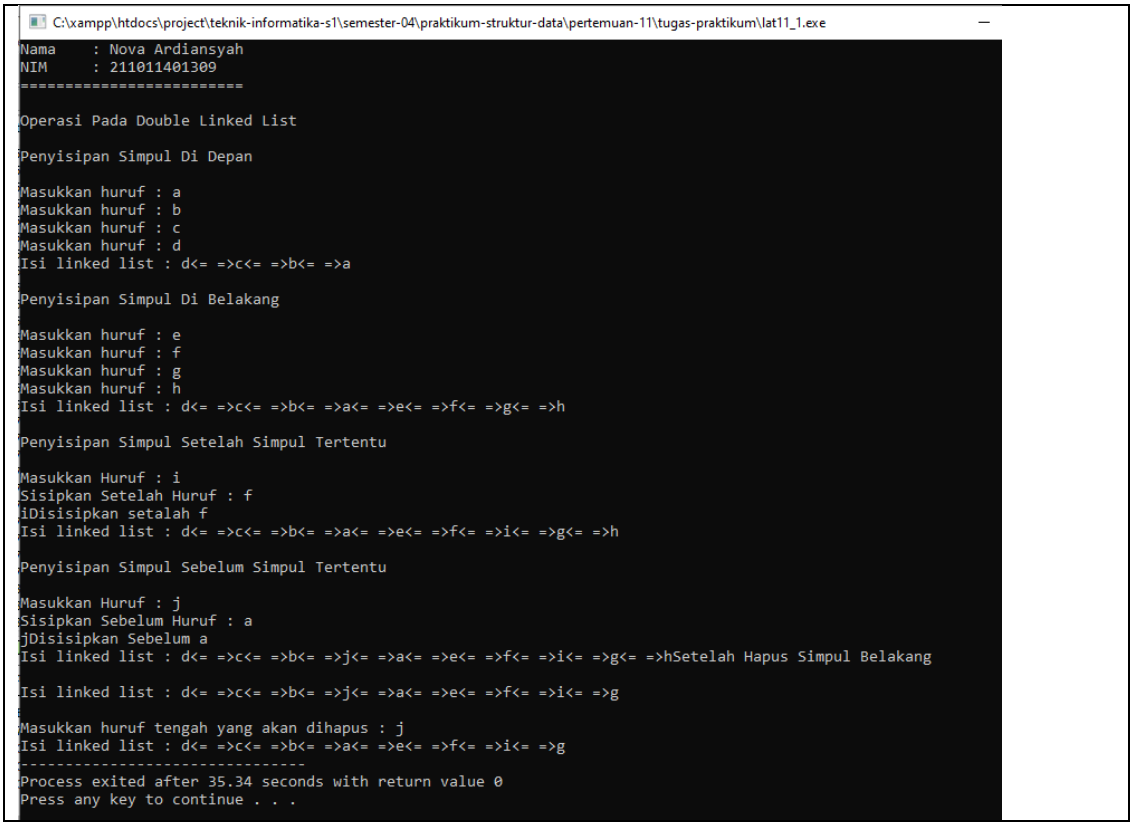
- Lat11_1

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

#define true 1
#define false 0

using namespace std;

typedef struct node *simpul;

struct node
{
  char Isi;
  simpul kanan;
  simpul kiri;
};

void Sisip_Depan (simpul &DL, char elemen);
void Sisip_Belakang (simpul &DL, char elemen);
void Sisip_Tengah1 (simpul &DL, char elemen1, char elemen2);
void Sisip_Tengah2 (simpul &DL, char elemen1, char elemen2);
void Hapus_Depan (simpul &DL);
void Hapus_Belakang (simpul &DL);
void Hapus_Tengah (simpul &DL, char elemen);
void Cetak (simpul DL);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  char huruf, huruf2;
  simpul DL = NULL;
  int i;

  cout << "Operasi Pada Double Linked List\n\n";

  cout << "Penyisipan Simpul Di Depan\n\n";
  for (i = 1; i <= 4; i++) {
    cout << "Masukkan huruf : ";
    cin >> huruf;
    Sisip_Depan(DL, huruf);
  }
  Cetak(DL);

  cout << "\n\nPenyisipan Simpul Di Belakang\n\n";
  for (i = 1; i <= 4; i++) {
    cout << "Masukkan huruf : ";
    cin >> huruf;
    Sisip_Belakang(DL, huruf);
  }
  Cetak(DL);

  // * Sisip simpul setelah simpul
  cout << "\n\nPenyisipan Simpul Setelah Simpul Tertentu\n\n";
  cout << "Masukkan Huruf : ";
```

```
  cin >> huruf;

  cout << "Sisipkan Setelah Huruf : ";
  cin >> huruf2;

  cout << huruf << "Disisipkan setalah " << huruf2 << endl;
  Sisip_Tengah1(DL, huruf, huruf2);

  Cetak(DL);

  //  * Sisip simpul sebelum simpul
  cout << "\n\nPenyisipan Simpul Sebelum Simpul Tertentu\n\n";
  cout << "Masukkan Huruf : ";
  cin >> huruf;

  cout << "Sisipkan Sebelum Huruf : ";
  cin >> huruf2;

  cout << huruf << "Disisipkan Sebelum " << huruf2 << endl;
  Sisip_Tengah2(DL, huruf, huruf2);

  Cetak(DL);

  // * Hapus simpul belakang
  cout << "Setelah Hapus Simpul Belakang\n\n";
  Hapus_Belakang(DL);
  Cetak(DL);

  // * Hapus simpul tengah
  cout << "\n\nMasukkan huruf tengah yang akan dihapus : ";
  cin>>huruf;
  Hapus_Tengah(DL, huruf);
  Cetak(DL);

  return 0;
}

void Sisip_Depan (simpul &DL, char elemen)
{
  simpul baru;
  baru = (simpul) malloc (sizeof(simpul));
  baru->Isi = elemen;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
    DL = baru;
  } else {
    baru->kanan = DL;
    DL->kiri = baru;
    DL = baru;
  }
}

void Sisip_Tengah1(simpul &DL, char elemen1, char elemen2)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen1;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
```

```
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->Isi != elemen2) {
      bantu = bantu->kanan;
    }
    baru->kanan = bantu->kanan;
    baru->kiri = bantu;
    bantu->kanan->kiri = baru;
    bantu->kanan = baru;
  }
}

void Sisip_Tengah2(simpul &DL, char elemen1, char elemen2)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen1;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->kanan->Isi != elemen2) {
      bantu = bantu->kanan;
    }
    baru->kanan = bantu->kanan;
    baru->kiri = bantu;
    bantu->kanan->kiri = baru;
    bantu->kanan = baru;
  }
}

void Sisip_Belakang(simpul &DL, char elemen)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(simpul));
  baru->Isi = elemen;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
    DL = baru;
  } else {
    bantu = DL;
    while (bantu->kanan != NULL) {
      bantu = bantu->kanan;
    }
    bantu->kanan = baru;
    baru->kiri = bantu;
  }
}


void Cetak(simpul DL)
{
  simpul bantu;
  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
```

```
   cout << "Isi linked list : ";
   while (bantu->kanan != NULL) {
     cout << bantu->Isi << "<= =>";
     bantu = bantu->kanan;
   }
   cout << bantu->Isi;
  }
}

void Hapus_Depan(simpul &DL)
{
  simpul hapus;
  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    hapus = DL;
    DL = DL->kanan;
    DL->kiri = NULL;
    hapus->kanan = NULL;
    free(hapus);
  }
}

void Hapus_Belakang(simpul &DL)
{
  simpul bantu, hapus;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->kanan->kanan != NULL) bantu = bantu->kanan;
    hapus = bantu->kanan;
    bantu->kanan = NULL;
    hapus->kiri = NULL;
    free(hapus);
  }
}

void Hapus_Tengah(simpul &DL, char elemen)
{
  simpul bantu, hapus;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->kanan->Isi != elemen)
    bantu = bantu->kanan;
    hapus = bantu->kanan;
    bantu->kanan->kanan->kiri = bantu;
    bantu->kanan = bantu->kanan->kanan;
    hapus->kanan = NULL;
    hapus->kiri = NULL;
    free(hapus);
  }
}
```


C:\xampp\htdocs\project\teknik-informatika-s1\semester-04\praktikum-struktur-data\pertemuan-11\tugas-praktikum\lat11_1.exe
Nama : Nova Ardiansyah
NIM : 211011401309
Operasi Pada Double Linked List
Penyisipan Simpul Di Depan
Masukkan huruf : a
Masukkan huruf : b
Masukkan huruf : c
Masukkan huruf : d
Isi linked list : d<= =>c<= =>b<= =>a
Penyisipan Simpul Di Belakang
Masukkan huruf : e
Masukkan huruf : f
Masukkan huruf : g
Masukkan huruf : h
Isi linked list : d<= =>c<= =>b<= =>a<= =>e<= =>f<= =>g<= =>h
Penyisipan Simpul Setelah Simpul Tertentu
Masukkan Huruf : i
Sisipkan Setelah Huruf : f
iDisisipkan setalah f
Isi linked list : d<= =>c<= =>b<= =>a<= =>e<= =>f<= =>i<= =>g<= =>h
Penyisipan Simpul Sebelum Simpul Tertentu
Masukkan Huruf : j
Sisipkan Sebelum Huruf : a
jDisisipkan Sebelum a
Isi linked list : d<= =>c<= =>b<= =>j<= =>a<= =>e<= =>f<= =>i<= =>g<= =>hSetelah Hapus Simpul Belakang
Isi linked list : d<= =>c<= =>b<= =>j<= =>a<= =>e<= =>f<= =>i<= =>g
Masukkan huruf tengah yang akan dihapus : j
Isi linked list : d<= =>c<= =>b<= =>a<= =>e<= =>f<= =>i<= =>g
Process exited after 35.34 seconds with return value 0
Press any key to continue . . .

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-10

Linked List (Lanjut)

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Modifikasi program diatas dengan ditambahkan linked List!

   Jawab :

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

#define true 1
#define false 0

using namespace std;

typedef struct node *simpul;

struct node
{
  char Isi;
  simpul kanan;
  simpul kiri;
};

void Sisip_Depan (simpul &DL, char elemen);
void Sisip_Belakang (simpul &DL, char elemen);
void Sisip_Tengah1 (simpul &DL, char elemen1, char elemen2);
void Sisip_Tengah2 (simpul &DL, char elemen1, char elemen2);
void Hapus_Depan (simpul &DL);
void Hapus_Belakang (simpul &DL);
void Hapus_Tengah (simpul &DL, char elemen);
void Cetak (simpul DL);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  char huruf, huruf2;
  simpul DL = NULL;
  int i;

  cout << "Operasi Pada Double Linked List\n\n";

  cout << "Penyisipan Simpul Di Depan\n\n";
  for (i = 1; i <= 4; i++) {
    cout << "Masukkan huruf : ";
    cin >> huruf;
    Sisip_Depan(DL, huruf);
  }
  Cetak(DL);

  cout << "\n\nPenyisipan Simpul Di Belakang\n\n";
  for (i = 1; i <= 4; i++) {
    cout << "Masukkan huruf : ";
    cin >> huruf;
    Sisip_Belakang(DL, huruf);
  }
  Cetak(DL);
```

```
  // * Sisip simpul setelah simpul
  cout << "\n\nPenyisipan Simpul Setelah Simpul Tertentu\n\n";
  cout << "Masukkan Huruf : ";
  cin >> huruf;

  cout << "Sisipkan Setelah Huruf : ";
  cin >> huruf2;

  cout << huruf << " Disisipkan setelah " << huruf2 << endl;
  Sisip_Tengah1(DL, huruf, huruf2);

  Cetak(DL);

  //  * Sisip simpul sebelum simpul
  cout << "\n\nPenyisipan Simpul Sebelum Simpul Tertentu\n\n";
  cout << "Masukkan Huruf : ";
  cin >> huruf;

  cout << "Sisipkan Sebelum Huruf : ";
  cin >> huruf2;

  cout << huruf << " Disisipkan Sebelum " << huruf2 << endl;
  Sisip_Tengah2(DL, huruf, huruf2);

  Cetak(DL);

  // * Hapus simpul belakang
  cout << "Setelah Hapus Simpul Belakang\n\n";
  Hapus_Belakang(DL);
  Cetak(DL);

  // * Hapus simpul tengah
  cout << "\n\nMasukkan huruf tengah yang akan dihapus : ";
  cin >> huruf;
  Hapus_Tengah(DL, huruf);
  Cetak(DL);

  return 0;
}

void Sisip_Depan (simpul &DL, char elemen)
{
  simpul baru;
  baru = (simpul) malloc (sizeof(node));
  baru->Isi = elemen;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
    DL = baru;
  } else {
    baru->kanan = DL;
    DL->kiri = baru;
    DL = baru;
  }
}

void Sisip_Tengah1(simpul &DL, char elemen1, char elemen2)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen1;
  baru->kanan = NULL;
```

```
  baru->kiri = NULL;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->Isi != elemen2) {
      bantu = bantu->kanan;
    }
    baru->kanan = bantu->kanan;
    if (bantu->kanan != NULL) {
      bantu->kanan->kiri = baru;
    }
    baru->kiri = bantu;
    bantu->kanan = baru;
  }
}

void Sisip_Tengah2(simpul &DL, char elemen1, char elemen2)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen1;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->kanan != NULL && bantu->kanan->Isi != elemen2) {
      bantu = bantu->kanan;
    }
    if (bantu->kanan != NULL) {
      baru->kanan = bantu->kanan;
      baru->kiri = bantu;
      bantu->kanan->kiri = baru;
      bantu->kanan = baru;
    } else {
      cout << "Simpul " << elemen2 << " tidak ditemukan\n";
    }
  }
}

void Sisip_Belakang(simpul &DL, char elemen)
{
  simpul bantu, baru;
  baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen;
  baru->kanan = NULL;
  baru->kiri = NULL;

  if (DL == NULL) {
    DL = baru;
  } else {
    bantu = DL;
    while (bantu->kanan != NULL) {
      bantu = bantu->kanan;
    }
    bantu->kanan = baru;
    baru->kiri = bantu;
  }
}
```

```
void Cetak(simpul DL)
{
  simpul bantu;
  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    cout << "Isi linked list : ";
    while (bantu->kanan != NULL) {
      cout << bantu->Isi << "<= =>";
      bantu = bantu->kanan;
    }
    cout << bantu->Isi;
  }
}

void Hapus_Depan(simpul &DL)
{
  simpul hapus;
  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    hapus = DL;
    DL = DL->kanan;
    if (DL != NULL) {
      DL->kiri = NULL;
    }
    hapus->kanan = NULL;
    free(hapus);
  }
}

void Hapus_Belakang(simpul &DL)
{
  simpul bantu, hapus;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->kanan->kanan != NULL) {
      bantu = bantu->kanan;
    }
    hapus = bantu->kanan;
    bantu->kanan = NULL;
    hapus->kiri = NULL;
    free(hapus);
  }
}

void Hapus_Tengah(simpul &DL, char elemen)
{
  simpul bantu, hapus;

  if (DL == NULL) {
    cout << "List Kosong.....\n";
  } else {
    bantu = DL;
    while (bantu->kanan != NULL && bantu->kanan->Isi != elemen) {
      bantu = bantu->kanan;
    }
```

```
    if (bantu->kanan != NULL) {
      hapus = bantu->kanan;
      bantu->kanan->kanan->kiri = bantu;
      bantu->kanan = bantu->kanan->kanan;
      hapus->kanan = NULL;
      hapus->kiri = NULL;
      free(hapus);
    } else {
      cout << "Simpul " << elemen << " tidak ditemukan\n";
    }
  }
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Operasi Pada Double Linked List

Penyisipan Simpul Di Depan

Masukkan huruf : a
Masukkan huruf : b
Masukkan huruf : c
Masukkan huruf : d
Isi linked list : d<= =>c<= =>b<= =>a

Penyisipan Simpul Di Belakang

Masukkan huruf : e
Masukkan huruf : f
Masukkan huruf : g
Masukkan huruf : h
Isi linked list : d<= =>c<= =>b<= =>a<= =>e<= =>f<= =>g<= =>h

Penyisipan Simpul Setelah Simpul Tertentu

Masukkan Huruf : i
Sisipkan Setelah Huruf : b
i Disisipkan setelah b
Isi linked list : d<= =>c<= =>b<= =>i<= =>a<= =>e<= =>f<= =>g<= =>h

Penyisipan Simpul Sebelum Simpul Tertentu

Masukkan Huruf : j
Sisipkan Sebelum Huruf : i
j Disisipkan Sebelum i
Isi linked list : d<= =>c<= =>b<= =>j<= =>i<= =>a<= =>e<= =>f<= =>g<= =>h
Setelah Hapus Simpul Belakang

Isi linked list : d<= =>c<= =>b<= =>j<= =>i<= =>a<= =>e<= =>f<= =>g

Masukkan huruf tengah yang akan dihapus : i
Isi linked list : d<= =>c<= =>b<= =>j<= =>a<= =>e<= =>f<= =>g
--------------------------------
Process exited after 20.07 seconds with return value 0
Press any key to continue . . .
```

2. KESIMPULAN

Struktur berupa rangkaian elemen saling berkait dimana tiap elemen dihubungkan ke elemen yang lain melalui pointer. Pointer adalah alamat elemen. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logic walaupun tidak bersebelahan secara fisik di memori. Linked List merupakan kumpulan komponen yang saling berkaitan satu dengan yang lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau verteks.

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AWAL PRAKTIKUM

Pertemuan ke-11

Stack

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Stack atau tumpukan adalah kumpulan elemen yang hanya dapat di tambah atau dihapus dari satu ujung (gerbang) yang sama. Hal ini menunjukan bahwa seolah-olah suatu elemen diletakan di atas elemen yang lain. Yang memberi gambaran bahwa Stack mempunyai sifat LIFO (Last In First Out) yang berarti bahwa elemen yang terakhir masuk akan pertama keluar. Secara sederhana stack dimisalkan kita mempunyai 4 buah kotak (A,B,C, dan D) yang ditumpukkan. Kotak A diletakkan paling bawah, lalu diikuti kotak B, C, dan yang teratas atau terakhir adalah D. Maka untuk mengambil tiap kotak harus dilakukan berurutan dari kotak D, C. B kemudian A. Karena jika kita mengambil kotak B tanpa terlebih dahulu mengambil kotak di atasnya maka tumpukan akan roboh.

B. TUGAS PENDAHULUAN

1. Apa yang dimaksud dengan Stack!

   Jawab :

   Stack atau Tumpukan adalah suatu struktur data yang terbentuk dari barisan hingga yang terurut dari satuan data. Pada Stack, penambahan dan penghapusan elemennya hanya dapat dilakukan pada satu posisi, yaitu posisi akhir stack.

2. Bagaimana tahapan-tahapan proses operasi PUSH!

   Jawab :

   - Periksa apakah stack penuh (isfull), jika bernilai false/0 (tidak penuh) maka proses push dilaksanakan dan jika pemeriksaan ini bernilai true/1, maka proses push digagalkan.
   - Proses push-nya sendiri adalah dengan menambahkan field top dengan 1, kemudian elemen pada posisi top di isi dengan elemen data baru

3. Bagaimana tahapan-tahapan proses operasi POP!

   Jawab :

Operasi ini biasanya dibuat dalam bentuk function yang me-return-kan nilai sesuai data yang ada di top. Operasi pop pada stack yang menggunakan array adalah terlebih dahulu memeriksa apakah stack sedang keadaan kosong, jika tidak kosong maka data diambil pada posisi yang ditunjuk oleh posisi top, kemudian disimpan dalam variabel baru dengan nama “data”.

4. Jelaskan karakteristik-karakteristik dari Stack!
   Jawab :

   - Elemen stack yaitu item-item data di elemen stack.
   - Top (elemen puncak dari stack)
   - Jumlah elemen pada stack
   - Status/kondisi stack Kondisi stack yang menjadi perhatian adalah [penuh/kosong]

C. TUGAS PRAKTIKUM

- Lat11_1

```
#include <iostream>
using namespace std;

#define MaxS 10

struct Stack {
  char Isi[MaxS];
  int Top;
};

void INITS(Stack& S);
void PUSH(Stack& S, char Data);
void CETAK(Stack S);
char POP(Stack& S, char& Hsl);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  char huruf;
  Stack S;
  INITS(S);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  CETAK(S);
  POP(S, huruf);

  cout << "\nYang dihapus ... : " << huruf << endl;
  CETAK(S);

  cout << "\nMasukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  CETAK(S);
  POP(S, huruf);
```

```
  cout << "\nYang dihapus ... : " << huruf << endl;
  CETAK(S);

  return 0;
}

void INITS(Stack& S)
{
  S.Top = -1;
}

void PUSH(Stack& S, char Data)
{
  if (S.Top < MaxS - 1) {
    S.Top++;
    S.Isi[S.Top] = Data;
  }
  else {
    cout << "Stack Penuh";
  }
}

void CETAK(Stack S)
{
  int i;
  cout << "\nIsi Stack : ";

  if (S.Top != -1) {
    for (i = 0; i <= S.Top; i++) {
      cout << S.Isi[i] << " ";
    }
  }
  else {
    cout << "Stack Kosong";
  }
}

char POP(Stack& S, char& Hsl)
{
  if (S.Top != -1) {
    Hsl = S.Isi[S.Top];
    S.Top--;
  }
  else {
    cout << "Stack Kosong";
  }

  return Hsl;
}
```

```
Nama     : Nova Ardiansyah
NIM      : 211011401309
=========================

Masukkan Karakter : a
Masukkan Karakter : b
Masukkan Karakter : c

Isi Stack : a b c
Yang dihapus ... : c

Isi Stack : a b
Masukkan Karakter : d
Masukkan Karakter : e
Masukkan Karakter : f

Isi Stack : a b d e f
Yang dihapus ... : f

Isi Stack : a b d e
```

- Lat11_2

```
#include <iostream>
#include <stack>
using namespace std;

#define MaxS 10

struct Stack {
  char Isi[MaxS];
  int Top;
};

void INITS(Stack& S);
void PUSH(Stack& S, char Data);
void CETAK(Stack S);
char POP(Stack& S, char& Hsl);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  char huruf;
  Stack S;
  INITS(S);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  CETAK(S);
  POP(S, huruf);

  cout << "\nYang dihapus ... : " << huruf << endl;
  CETAK(S);
```

```
  cout << "\nMasukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  cout << "Masukkan Karakter : ";
  cin >> huruf;
  PUSH(S, huruf);

  CETAK(S);
  POP(S, huruf);

  cout << "\nYang dihapus ... : " << huruf << endl;
  CETAK(S);

  // Membalik karakter-karakter dalam stack
  stack<char> charStack;
  for (int i = 1; i <= S.Top; i++) {
    charStack.push(S.Isi[i]);
  }

  cout << "\nKarakter yang terbalik: ";
  while (!charStack.empty()) {
    cout << charStack.top();
    charStack.pop();
  }

  return 0;
}

void INITS(Stack& S)
{
  S.Top = 0;
}

void PUSH(Stack& S, char Data)
{
  if (S.Top < MaxS) {
    S.Top++;
    S.Isi[S.Top] = Data;
  } else {
    cout << "Stack Penuh";
  }
}

void CETAK(Stack S)
{
  int i;
  cout << "\nIsi Stack : ";

  if (S.Top != 0) {
    for (i = 1; i <= S.Top; i++) {
      cout << S.Isi[i] << " ";
    }
  } else {
    cout << "Stack Kosong";
  }
}
```

```
char POP(Stack& S, char& Hsl)
{
  if (S.Top != 0) {
    Hsl = S.Isi[S.Top];
    S.Top--;
  } else {
    cout << "Stack Kosong";
  }

  return Hsl;
}
```

```
Nama     : Nova Ardiansyah
NIM      : 211011401309
=======================

Masukkan Karakter : a
Masukkan Karakter : b
Masukkan Karakter : c

Isi Stack : a b c
Yang dihapus ... : c

Isi Stack : a b
Masukkan Karakter : d
Masukkan Karakter : e
Masukkan Karakter : f

Isi Stack : a b d e f
Yang dihapus ... : f

Isi Stack : a b d e
Karakter yang terbalik: edba
```

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-11

Stack

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Buatlah program untuk mengkonversi bilangan desimal menjadi bilangan biner dengan menggunakan Stack!

   Jawab :

```
#include <iostream>
using namespace std;

const int MAX_SIZE = 100;

void DecimalToBinary(int decimal) {
  int binary[MAX_SIZE];
  int index = 0;

  while (decimal > 0) {
    int remainder = decimal % 2;
    binary[index++] = remainder;
    decimal /= 2;
  }

  cout << "Biner: ";
  for (int i = index - 1; i >= 0; i--) {
    cout << binary[i];
  }
  cout << endl;
}

int main() {
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  int decimal;

  cout << "Masukkan bilangan desimal: ";
  cin >> decimal;

  DecimalToBinary(decimal);

  return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Masukkan bilangan desimal: 15
Biner: 1111

--------------------------------
Process exited after 4.582 seconds with return value 0
Press any key to continue . . .
```

2. KESIMPULAN

Stack adalah suatu tumpukan. Konsep utama dari stack adalah LIFO (Last In First Out), yaitu benda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari tumpukan. Dalam C++ ada dua cara penerapan stack. Sesuai dengan sifat stack, maka pengambilan/penghapusan elemen dalam stack harus dimulai dari elemen teratas. Deklarasi konstanta, tipe, dan variable yang akan di pakai dalam penjelasan operasi-operasi stack dengan array.

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AWAL PRAKTIKUM

Pertemuan ke-12

Stack Lanjut

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Stack atau tumpukan adalah kumpulan elemen yang hanya dapat di tambah atau dihapus dari satu ujung (gerbang) yang sama. Hal ini menunjukan bahwa seolah-olah suatu elemen diletakan di atas elemen yang lain. Yang memberi gambaran bahwa Stack mempunyai sifat LIFO (Last In First Out) yang berarti bahwa elemen yang terakhir masuk akan pertama keluar. Representasi Stack dapat dilakukan menggunakan Array atau Linked List. Kedua representasi mempunyai keunggulan dan kelemahan. Dengan Array, stack juga dapat disajikan dengan Single Stack dan Double Stack.

B. TUGAS PENDAHULUAN

1. Jelaskan perbedaan program Stack antara menggunakan Array dan Linked List!
   Jawab :

   Program stack menggunakan array.

   Proses inisialisasi dimana proses ini untuk stack yang menggunakan array adalah dengan mengisi nilai field top dengan 0 (nol), jika elemen pertama diawali dengan nomor 1. Kalau elemen pertama array dimulai dengan 0 (contoh bahasa c), maka top di isi dengan nilai -1.

   - Top yang menunjuk posisi data terakhir (top).
   - Elemen yang berisi data yang ada dalam stack. Bagian ini lah yang berbentuk array.
   - Maks elemen yaitu variabel yang menunjuk maksimal banyaknya elemen dalam stack

   Program stack menggunakan linked list

   Adapun stack yang menggunakan linked list, hanya memerlukan suatu pointer yang menunjuk ke data terakhir (perhatikan proses dihalaman sebelumnya), setiap elemen linked list mempunyai 2 field yaitu elemen datanya dan pointer bawah yang menunjuk posisi terakhir sebelum proses push

2. Jelaskan Aplikasi-Aplikasi Stack dalam dunia nyata!
   Jawab :

Dalam dunia nyata bisa kita bayangkan seperti tumpukan buku, tumpukan kartu, atau tumpukan kursi yang tersusun secara menumpuk ke atas. Konsep stack yang utuh memiliki beberapa aturan atau batasan tersendiri yang membedakannya dengan struktur data lain, misalnya kita tidak bisa menambah data langsung ditengah-tengah tumpukan dengan cara diselipkan. Beberapa contoh aplikasi yang menerapkan stack, diantaranya adalah:

- Expression evaluation, baik ekspresi aritmatika, lojik maupun boolean.
- Notasi infix, prefix, dan postfix, proses perhitungannya maupun konversi antar notasi tersebut
- Backtracking, contohnya history call pada browser (tombol back).
- Membantu penelusuran simpul pohon dengan algoritma DFS (Depth-First-Search).
- Manajemen memori dan alokasi memori, komputer modern saat ini menerapkan stack untuk memodelkan manajemen memori dari program yang sedang berjalan (running program).
- Permainan Tower of Hanoi.
- Konversi bilangan desimal ke binner. Sampai yang paling sederhana yaitu membalikkan urutan string

3. Tuliskan contoh program pada operasi Full!

   Jawab :

```
int IsFull ()
{
  if (tumpuk.top == MAX_STACK-1)
    return 1;
  else
    return 0;
}
```

4. Tuliskan contoh program pada operasi Empty!

   Jawab :

```
int IsEmpty ()
{
  if (tumpuk.top == -1)
    return 1;
  else
    return 0;
}
```

C. TUGAS PRAKTIKUM

- Lat12_1

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

#define true 1
#define false 0

using namespace std;

typedef struct node *simpul;

struct node {
  char Isi;
  simpul Next;
};

void Sisip_Belakang(simpul &L, char elemen);
void Hapus_Belakang(simpul &L);
void Cetak(simpul L);

int main() {
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  char huruf;
  simpul L = NULL;
  cout << "Operasi Single Linked List Pada Stack\n\n";

  cout << "Penyisipan Stack \n\n";

  cout << "Masukkan Elemen : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);
  cout << "Masukkan Elemen : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);
  cout << "Masukkan Elemen : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);
  cout << "Masukkan Elemen : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);
  cout << "Masukkan Elemen : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);
  cout << "Masukkan Elemen : ";
  cin >> huruf;
  Sisip_Belakang(L, huruf);

  Cetak(L);

  cout << "\n\nHapus Elemen \n";
  Hapus_Belakang(L);
  Cetak(L);

  cout << "\n\nHapus Elemen \n";
  Hapus_Belakang(L);
  Cetak(L);
```

```
  cout << "\n\nHapus Elemen \n";
  Hapus_Belakang(L);
  Cetak(L);

  cout << "\n\nHapus Elemen \n";
  Hapus_Belakang(L);
  Cetak(L);

  return 0;
}

void Sisip_Belakang(simpul &L, char elemen) {
  simpul baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen;
  baru->Next = NULL;

  if (L == NULL) {
    L = baru;
  } else {
    simpul bantu = L;
    while (bantu->Next != NULL) {
      bantu = bantu->Next;
    }
    bantu->Next = baru;
  }
}

void Hapus_Belakang(simpul &L) {
  if (L == NULL) {
    cout << "List Kosong, Tidak ada yang dihapus\n";
  } else if (L->Next == NULL) {
    free(L);
    L = NULL;
  } else {
    simpul bantu = L;
    while (bantu->Next->Next != NULL) {
      bantu = bantu->Next;
    }
    simpul hapus = bantu->Next;
    bantu->Next = NULL;
    free(hapus);
  }
}

void Cetak(simpul L) {
  if (L == NULL) {
    cout << "List Kosong\n";
  } else {
    simpul bantu = L;
    cout << "\nIsi List : ";
    while (bantu->Next != NULL) {
      cout << bantu->Isi << "->";
      bantu = bantu->Next;
    }
    cout << bantu->Isi;
  }
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Operasi Single Linked List Pada Stack

Penyisipan Stack

Masukkan Elemen : a
Masukkan Elemen : b
Masukkan Elemen : c
Masukkan Elemen : d
Masukkan Elemen : e
Masukkan Elemen : f

Isi List : a->b->c->d->e->f

Hapus Elemen

Isi List : a->b->c->d->e

Hapus Elemen

Isi List : a->b->c->d

Hapus Elemen

Isi List : a->b->c

Hapus Elemen

Isi List : a->b
--------------------------------
Process exited after 8.578 seconds with return value 0
Press any key to continue . . .
```

- Lat12_2

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

using namespace std;

typedef struct node *simpul;

struct node {
  char Isi;
  simpul Next;
};

void Sisip_Belakang(simpul &L, char elemen);
void Hapus_Belakang(simpul &L);
void Cetak(simpul L);
void TampilkanMenu(simpul &L);

int main() {
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "========================\n\n";

  char huruf;
  simpul L = NULL;
  cout << "Operasi Single Linked List Pada Stack\n\n";

  int pilihan;
  do {
    TampilkanMenu(L);
    cout << "Pilih operasi (1-3): ";
    cin >> pilihan;

    switch (pilihan) {
```

```
      case 1:
        cout << "Masukkan Elemen : ";
        cin >> huruf;
        Sisip_Belakang(L, huruf);
        break;
      case 2:
        Hapus_Belakang(L);
        break;
      case 3:
        Cetak(L);
        break;
      case 0:
        cout << "Terima kasih. Program selesai.\n";
        break;
      default:
        cout << "Pilihan tidak valid. Silakan coba lagi.\n";
        break;
    }
  } while (pilihan != 0);

  return 0;
}

void Sisip_Belakang(simpul &L, char elemen) {
  simpul baru = (simpul) malloc(sizeof(node));
  baru->Isi = elemen;
  baru->Next = NULL;

  if (L == NULL) {
    L = baru;
  } else {
    simpul bantu = L;
    while (bantu->Next != NULL) {
      bantu = bantu->Next;
    }
    bantu->Next = baru;
  }
}

void Hapus_Belakang(simpul &L) {
  if (L == NULL) {
    cout << "List Kosong, Tidak ada yang dihapus\n";
  } else if (L->Next == NULL) {
    free(L);
    L = NULL;
  } else {
    simpul bantu = L;
    while (bantu->Next->Next != NULL) {
      bantu = bantu->Next;
    }
    simpul hapus = bantu->Next;
    bantu->Next = NULL;
    free(hapus);
  }
}

void Cetak(simpul L) {
  if (L == NULL) {
    cout << "List Kosong\n";
  } else {
    simpul bantu = L;
    cout << "\nIsi List : ";
    while (bantu->Next != NULL) {
```

```
    cout << bantu->Isi << "->";
    bantu = bantu->Next;
   }
   cout << bantu->Isi;
  }
  cout << endl;
}


void TampilkanMenu(simpul &L) {
  cout << "\nMenu:\n";
  cout << "1. Sisipkan Elemen\n";
  cout << "2. Hapus Elemen\n";
  cout << "3. Cetak List\n";
  cout << "4. Keluar\n";
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Operasi Single Linked List Pada Stack


Menu:
1. Sisipkan Elemen
2. Hapus Elemen
3. Cetak List
4. Keluar
Pilih operasi (1-3): 1
Masukkan Elemen : a

Menu:
1. Sisipkan Elemen
2. Hapus Elemen
3. Cetak List
4. Keluar
Pilih operasi (1-3): 1
Masukkan Elemen : b

Menu:
1. Sisipkan Elemen
2. Hapus Elemen
3. Cetak List
4. Keluar
Pilih operasi (1-3): 3

Isi List : a->b
```

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-12

Stack Lanjut

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Buatlah program untuk mengetahui suatu kalimat adalah polindrom atau tidak! Polindrom adalah suatu kata atau kalimat yang jika dibaca dari depan akan sama maknanya dengan jika dibaca dari belakang. Contoh: "KASUR NABABAN RUSAK" maka jika kalimat tersebut dibalik akan mempunyai makna yang sama yaitu: "KASUR NABABAN RUSAK"!

   Jawab :

```
#include<iostream>
#include<string>

using namespace std;

int main()
{
 cout << "Nama \t: Nova Ardiansyah\n";
 cout << "NIM \t: 211011401309\n";
 cout << "========================\n\n";

 string kalimat;
 cout << "Masukkan Kalimat : ";
 getline(cin, kalimat);

 int len = kalimat.length();
 bool flag = true;

 for(int i = 0, j = len - 1; i < len / 2; ++i, --j)
 {
  if(kalimat[j] != kalimat[i])
  {
   flag = false;
   break;
  }
 }

 cout << "\nKata Setelah Dibalik: ";
 for(int i = len - 1; i >= 0; --i)
 {
  cout << kalimat[i];
 }

 if(flag)
  cout << "\nKalimat Ini adalah Kalimat Palindrome";
 else
  cout << "\nKalimat Ini Bukan Kalimat Palindrome";

 return 0;
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

Masukkan Kalimat : kasur nababan rusak

Kata Setelah Dibalik: kasur nababan rusak
Kalimat Ini adalah Kalimat Palindrome
--------------------------------
Process exited after 11.29 seconds with return value 0
Press any key to continue . . .
```

2. KESIMPULAN

Stack adalah suatu tumpukan. Konsep utama dari stack adalah LIFO (Last In First Out), yaitubenda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari tumpukan. Dalam C++ ada dua cara penerapan stack. Sesuai dengan sifat stack, maka pengambilan/penghapusan elemen dalam stack haru s dimulai dari elemen teratas. Deklarasi konstanta, tipe, dan variable yang akan di pakai dalam penjelasan operasi-operasi stack dengan array.

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AWAL PRAKTIKUM

Pertemuan ke-13

Queue

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Queue atau Antrian merupakan kumpulan elemen dengan penyisipan dan penghapusan elemen yang dilakukan dari sisi/gerbang yang berbeda. Penyisipan dilakukan dari gerbang belakang dan penghapusan dilakukan dari gerbang depan. Hal ini menunjukan bahwa untuk Queue mempunyai dua gerbang yaitu gerbang depan dan gerbang belakang. Dengan demikian dapat dilihat bahwa Queue mempunyai sifat FIFO (first In Firs Out), yaitu elemen yang pertama masuk akan keluar pertama juga. Queue dapat direpresentasikan dengan menggunakan Array atau Linked List.

B. TUGAS PENDAHULUAN

1. Apa yang dimaksud dengan Queue!

   Jawab :

   Queue atau Antrian merupakan kumpulan elemen dengan penyisipan dan penghapusan elemen yang dilakukan dari sisi/gerbang yang berbeda.

2. Tuliskan Deklarasi sintaks Queue!

   Jawab :

Deklarasi Awal :

```
#define max 7;
int data[max];
int head = -1, tail = - 1;
```

IsEmpty :

```
bool IsEmpty() {
  if (head == -1 && tail == -1)
    return true;
 else
   return false;
}
```

IsFull :

```
bool IsFull() {
  if (tail == max-1)
    return true;
  else
    return false;
}
```

Enqueue :

```
void Enqueue() {
  if (IsFull()) {
    cout << "Antrian penuh!" << endl;
  } else {
```

```
    if (IsEmpty()) {
      head = tail = 0;
      cout << "Masukkan data: ";
      cin >> data[head];
    } else {
      tail++;
      cout << "Masukkan data: ";
      cin >> data[tail];
    }
  }
}
```

Dequeue :

```
void Dequeue() {
  if (IsEmpty()) {
    cout << "Antrian kosong!" << endl;
  } else {
    for (int i = 0; i < tail; i++) {
      data[i] = data[i + 1];
    }
    tail--;
  }
}
```

Clear :

```
void Clear() {
  if (IsEmpty()) {
    cout << "Antrian kosong!" << endl;
  } else {
    tail = -1;
    cout << "Antrian berhasil dikosongkan!" << endl;
  }
}
```

Cetak :

```
void Cetak() {
  if (IsEmpty()) {
    cout << "Antrian kosong!" << endl;
  } else {
    cout << "Isi antrian: ";
    for (int i = 0; i <= tail; i++) {
      cout << data[i] << " ";
    }
    cout << endl;
  }
}
```

3. Sebutkan dan Jelaskan operasi-operasi pada Queue!

   Jawab :

   - **Create()** Untuk menciptakan dan menginisialisasi Queue Dengan cara membuat Head dan Tail = -1.
   - **IsEmpty()** Untuk memeriksa apakah Antrian sudah penuh atau belum Dengan cara memeriksa nilai Tail, jika Tail = -1 maka empty Kita tidak memeriksa Head, karena Head adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah Pergerakan pada

Antrian terjadi dengan penambahan elemen Antrian kebelakang, yaitu menggunakan nilai Tail.

- **IsFull()** Untuk mengecek apakah Antrian sudah penuh atau belum Dengan cara mengecek nilai Tail, jika Tail >= MAX-1 (karena MAX-1 adalah batas elemen array pada C) berarti sudah penuh.
- **Enqueue()** Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang Penambahan elemen selalu menggerakan variabel Tail dengan cara increment counter Tail terlebih dahulu.
- **Dequeue()** Digunakan untuk menghapus elemen terdepan/pertama (head) dari Antrian Dengan cara menggeser semua elemen antrian kedepan dan mengurangi Tail dgn 1 Penggeseran dilakukan dengan menggunakan looping.
- **Clear()** Untuk menghapus elemen-elemen Antrian dengan cara membuat Tail dan Head = -1. Penghapusan elemen-elemen Antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesan-nya ke nilai -1 sehingga elemen- elemen Antrian tidak lagi terbaca.
- **Cetak()** Untuk menampilkan nilai-nilai elemen Antrian Menggunakan looping dari head s/d tail.

4. Jelaskan Aplikasi-Aplikasi Queue dalam dunia nyata!
   Jawab :

Antrian pada penjualan tiket kereta api, dimana orang yang pertama datang adalah orang yang pertama kali dilayani untuk membeli tiket. Jika ada orang baru yang datang akan membeli tiket, maka posisinya berada pada urutan paling belakang dalam antrian tersebut. Orang yang berada pada posisi terakhir dalam antrian adalah yang terakhir kali dapat dilayani dan memperoleh tiket kereta api (kalau kurang beruntung, maka akan kehabisan tiket).

Contoh lain adalah nasabah yang antri di teller bank, paket data yang menunggu untuk di transmisikan lewat internet, antrian printer dimana terdapat antrian print job yang menunggu giliran untuk menggunakan printer dan sebagainya.

Contoh aplikasi Queue dalam dunia nyata :

- Aplikasi antrian di jalan Tol.
- Aplikasi antrian saat mengantri di loket
- Aplikasi antraian reservasi tiket kereta api, dll

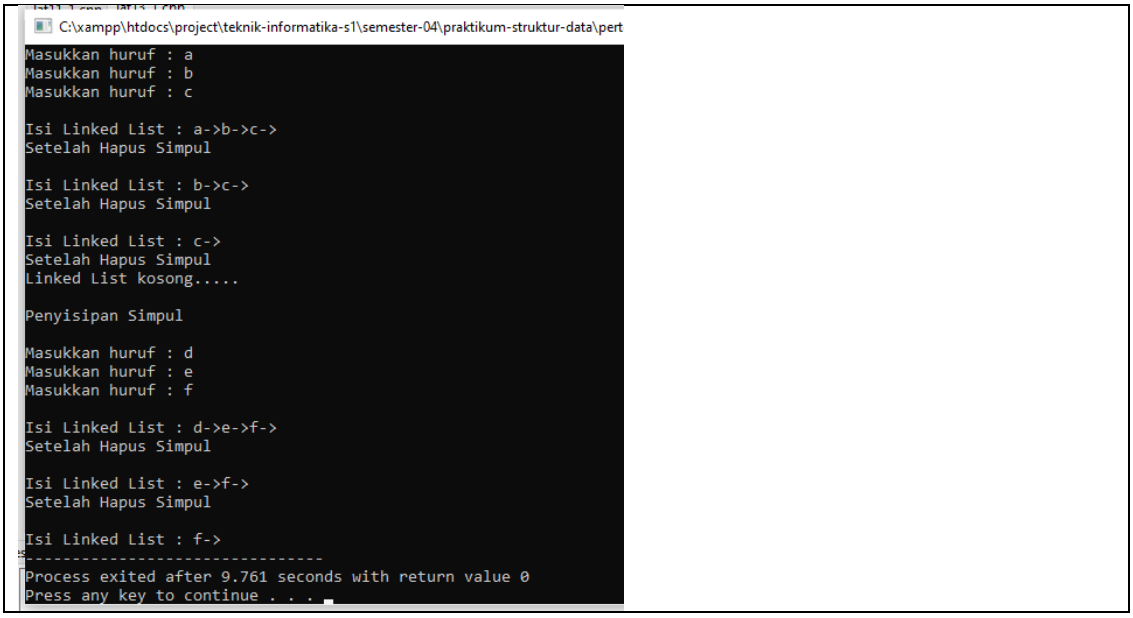
Semua itu menggunakan aturan FIFO (First In, First Out).

C. TUGAS PRAKTIKUM

- Lat13_1

```
#include <iostream>
using namespace std;

typedef struct node *simpul;
struct node
{
  char Isi;
  simpul Next;
};

void Sisip_Belakang(simpul &L, char elemen);
void Hapus_Depan(simpul &L);
void Cetak(simpul L);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  char huruf;
  simpul L = NULL;
  int i;

  cout << "== Operasi pada Single Linked List ==" << endl;

  cout << "\nPenyisipan Simpul\n\n";
  for (i = 1; i <= 3; i++)
  {
    cout << "Masukkan huruf : ";
    cin >> huruf;
    Sisip_Belakang(L, huruf);
  }
  Cetak(L);

  cout << "\nSetelah Hapus Simpul\n";
  Hapus_Depan(L);
  Cetak(L);

  cout << "\nSetelah Hapus Simpul\n";
  Hapus_Depan(L);
  Cetak(L);

  cout << "\nSetelah Hapus Simpul\n";
  Hapus_Depan(L);
  Cetak(L);

  cout << "\nPenyisipan Simpul\n\n";
  for (i = 1; i <= 3; i++)
  {
    cout << "Masukkan huruf : ";
    cin >> huruf;
    Sisip_Belakang(L, huruf);
  }
  Cetak(L);

  cout << "\nSetelah Hapus Simpul\n";
  Hapus_Depan(L);
```

```
  Cetak(L);

  cout << "\nSetelah Hapus Simpul\n";
  Hapus_Depan(L);
  Cetak(L);

  return 0;
}

void Sisip_Belakang(simpul &L, char elemen)
{
  simpul baru = new node;
  baru->Isi = elemen;
  baru->Next = NULL;

  if (L == NULL)
  {
    L = baru;
  }
  else
  {
    simpul bantu = L;
    while (bantu->Next != NULL)
    {
      bantu = bantu->Next;
    }
    bantu->Next = baru;
  }
}

void Cetak(simpul L)
{
  simpul bantu = L;
  if (L == NULL)
  {
    cout << "Linked List kosong....." << endl;
  }
  else
  {
    cout << "\nIsi Linked List : ";
    while (bantu != NULL)
    {
      cout << bantu->Isi << "->";
      bantu = bantu->Next;
    }
  }
}

void Hapus_Depan(simpul &L)
{
  if (L == NULL)
  {
    cout << "Linked List kosong....." << endl;
  }
  else
  {
    simpul Hapus = L;
    L = L->Next;
    Hapus->Next = NULL;
    delete Hapus;
  }
}
```


C:\xampp\htdocs\project\teknik-informatika-s1\semester-04\praktikum-struktur-data\pert
Masukkan huruf : a
Masukkan huruf : b
Masukkan huruf : c
Isi Linked List : a->b->c->
Setelah Hapus Simpul
Isi Linked List : b->c->
Setelah Hapus Simpul
Isi Linked List : c->
Setelah Hapus Simpul
Linked List kosong.....
Penyisipan Simpul
Masukkan huruf : d
Masukkan huruf : e
Masukkan huruf : f
Isi Linked List : d->e->f->
Setelah Hapus Simpul
Isi Linked List : e->f->
Setelah Hapus Simpul
Isi Linked List : f->

Process exited after 9.761 seconds with return value 0
Press any key to continue . . .

LAPORAN PRAKTIKUM STRUKTUR DATA
LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-13

Queue

Disusun Oleh:

Nama Lengkap	:	Nova Ardiansyah
NIM	:	211011401309
Kelas	:	04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Buatlah program lat13_1.cpp diatas, dengan menggunakan system menu!

   Jawab :

```
#include <iostream>
using namespace std;

typedef struct node *simpul;
struct node
{
  char Isi;
  simpul Next;
};

void Sisip_Belakang(simpul &L, char elemen);
void Hapus_Depan(simpul &L);
void Cetak(simpul L);
void Menu(simpul &L);

int main()
{
  cout << "Nama \t: Nova Ardiansyah\n";
  cout << "NIM \t: 211011401309\n";
  cout << "=========================\n\n";

  simpul L = NULL;

  cout << "== Operasi pada Single Linked List ==" << endl;

  Menu(L);

  return 0;
}

void Menu(simpul &L)
{
  int pilihan;
  char huruf;

  while (true)
  {
    cout << "\nMenu Pilihan:" << endl;
    cout << "1. Sisipkan huruf" << endl;
    cout << "2. Hapus huruf dari depan" << endl;
    cout << "3. Cetak linked list" << endl;
    cout << "4. Keluar" << endl;
    cout << "Pilihan Anda: ";
    cin >> pilihan;

    switch (pilihan)
    {
    case 1:
      cout << "Masukkan huruf: ";
      cin >> huruf;
      Sisip_Belakang(L, huruf);
      break;
    case 2:
      Hapus_Depan(L);
      break;
    case 3:
```

```
      Cetak(L);
      break;
    case 4:
      return;
    default:
      cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
      break;
    }
  }
}

void Sisip_Belakang(simpul &L, char elemen)
{
  simpul baru = new node;
  baru->Isi = elemen;
  baru->Next = NULL;

  if (L == NULL)
  {
    L = baru;
  }
  else
  {
    simpul bantu = L;
    while (bantu->Next != NULL)
    {
      bantu = bantu->Next;
    }
    bantu->Next = baru;
  }
}

void Cetak(simpul L)
{
  simpul bantu = L;
  if (L == NULL)
  {
    cout << "Linked List kosong....." << endl;
  }
  else
  {
    cout << "\nIsi Linked List : ";
    while (bantu != NULL)
    {
      cout << bantu->Isi << "->";
      bantu = bantu->Next;
    }
  }
}

void Hapus_Depan(simpul &L)
{
  if (L == NULL)
  {
    cout << "Linked List kosong....." << endl;
  }
  else
  {
    simpul Hapus = L;
    L = L->Next;
    Hapus->Next = NULL;
    delete Hapus;
  }
```

```
}
```

```
Nama    : Nova Ardiansyah
NIM     : 211011401309
========================

== Operasi pada Single Linked List ==

Menu Pilihan:
1. Sisipkan huruf
2. Hapus huruf dari depan
3. Cetak linked list
4. Keluar
Pilihan Anda: 1
Masukkan huruf: a

Menu Pilihan:
1. Sisipkan huruf
2. Hapus huruf dari depan
3. Cetak linked list
4. Keluar
Pilihan Anda: 1
Masukkan huruf: b

Menu Pilihan:
1. Sisipkan huruf
2. Hapus huruf dari depan
3. Cetak linked list
4. Keluar
Pilihan Anda: 3

Isi Linked List : a->b->
Menu Pilihan:
1. Sisipkan huruf
2. Hapus huruf dari depan
3. Cetak linked list
4. Keluar
Pilihan Anda: 4

--------------------------------
Process exited after 20.8 seconds with return value 0
Press any key to continue . . .
```

2. KESIMPULAN

Queue mempunyai dua gerbang yaitu gerbang depan dan gerbang belakang. Dengan demikian dapat dilihat bahwa Queue mempunyai sifat FIFO (first In Firs Out), yaitu elemen yang pertama masuk akan keluar pertama juga. Queue dapat direpresentasikan dengan menggunakan Array atau Linked List..