Modul Praktikum

Algoritma dan Struktur Data

Double Linked List

Tenia Wahyuningrum, S.Kom., MT

Sisilia Thya Safitri, ST., MT

ST3 Telkom Purwokerto

Jl. Dl Panjaitan 128 Purwokerto

* Untuk kalangan sendiri

Praktikum 8
Materi : Double Linked List
Waktu : 100 menit

Dasar Teori

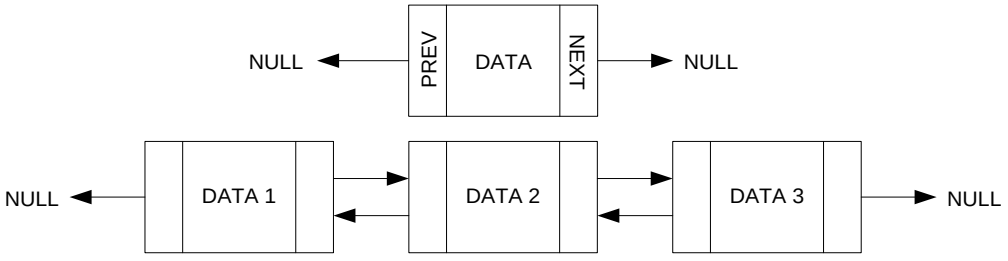
- Pada dasarnya, penggunaan Double Linked List hampir sama dengan penggunaan Single Linked List yang telah kita pelajari pada materi sebelumnya. Hanya saja Double Linked List menerapkan sebuah pointer baru, yaitu **prev**, yang digunakan untuk menggeser mundur selain tetap mempertahankan pointer **next**.
- Keberadaan 2 pointer penunjuk (**next** dan **prev**) menjadikan Double Linked List menjadi lebih fleksibel dibandingkan Single Linked List, namun dengan mengorbankan adanya memori tambahan dengan adanya pointer tambahan tersebut.
- Ada 2 jenis Double Linked List, yaitu: Double Linked List Non Circular dan Double Linked List Circular.

1. DOUBLE LINKED LIST NON CIRCULAR (DLLNC)

a. DLLNC

- DLLNC adalah sebuah Linked List yang terdiri dari dua arah pointer, dengan node yang saling terhubung, namun kedua pointernya menunjuk ke NULL.
- Setiap node pada linked list mempunyai field yang berisi data dan pointer yang saling berhubungan dengan node yang lainnya.

b. GAMBARAN NODE DLLNC



c. PEMBUATAN DLLNC

Buatlah sebuah project dengan nama Guided1_DLLNC_NIM. Kemudian ikuti dan cermati langkah-langkah berikut ini:

- **Deklarasi Node**

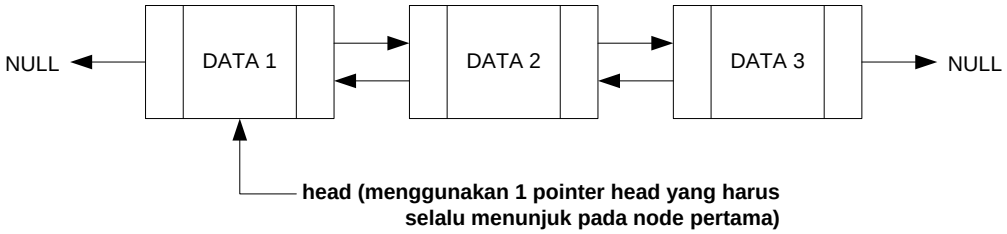
```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

using namespace std;

typedef struct TNode
{
    int data;
    TNode *next;
    TNode *prev;
};

TNode *head;
```

- **Pembuatan DLLNC dengan Head**
  - Ilustrasi :



  - Fungsi-fungsi yang biasa digunakan :
    - ✓ Fungsi untuk inisialisasi awal

```
void init() // inisialisasi awal
{
    head = NULL;
}
```

    - ✓ Perlu diperhatikan :
      - Fungsi ini harus ada, untuk memunculkan node awal.
      - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini. Jangan lupa untuk mendeklarasikan node-nya terlebih dahulu.

    - ✓ Fungsi untuk mengecek kosong tidaknya Linked List

```
int isEmpty() // mengecek kosong tidaknya Linked List
{
    if(head == NULL)
        return 1;
    else
        return 0;
}
```

- ✓ Perlu diperhatikan :
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menambahkan data di depan

```
void insertDepan(int value) // penambahan data di depan
{
    TNode *baru;
    baru = new TNode; // pembentukan node baru

    baru->data = value;  // pemberian nilai terhadap data baru
    baru->next = NULL;   // data pertama harus menunjuk ke NULL
    baru->prev = NULL;   // data pertama harus menunjuk ke NULL

    if(isEmpty() == 1)      // jika Linked List kosong
    {
        head = baru;        // head harus selalu berada di depan
        head->next = NULL;
        head->prev = NULL;
    }
    else                    // jika Linked List sudah ada datanya
    {
        baru->next = head;   // node baru dihubungkan ke head
        head->prev = baru;   // node head dihubungkan ke node baru
        head = baru;         // head harus selalu berada di depan
    }

    printf("Data Masuk\n");
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di depan. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menambahkan data di belakang

```
void insertBelakang(int value)  // penambahan data di belakang
    {
    TNode *baru, *bantu;
    baru = new TNode;           // pembentukan node baru

    baru->data = value;         // pemberian nilai terhadap data baru
    baru->next = NULL;          // data pertama harus menunjuk ke NULL
    baru->prev = NULL;          // data pertama harus menunjuk ke NULL

    if(isEmpty() == 1)          // jika Linked List kosong
    {
        head = baru;            //head harus selalu berada di depan
        head->next = NULL;
        head->prev = NULL;
    }
    else
    {
        bantu = head;           // bantu diletakan di head dulu
        while(bantu->next != NULL)
    {
        bantu = bantu->next;    // menggeser hingga node terakhir
    }

    baru->next = baru;          // node baru dihubungkan ke head
    head->prev = bantu;         // node head dihubungkan ke node baru
    }
    printf("Data Masuk\n");
}
```

- ✓ Perlu diperhatikan :
  - Jika Linked List hanya menggunakan head, maka dibutuhkan satu pointer untuk membantu mengetahui node terakhir dari Linked List. Dalam code di atas digunakan pointer bantu.
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di belakang. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menambahkan data di tengah (menyisipkan data)

```
void insertTengah(int value, int cari) //penambahan data di tengah
{
    TNode *baru, *bantu, *bantu2;
    baru = new TNode;             // pembentukan node baru

    baru->data = value;           // pemberian nilai terhadap data baru
    baru->next = NULL;            // data pertama harus menunjuk ke NULL
    baru->prev = NULL;            // data pertama harus menunjuk ke NULL
    bantu = head;                 // bantu diletakan di head dulu

    while(bantu->data != cari)
    {
        bantu = bantu->next;      //menggeser hingga didapat data cari
    }

    bantu2 = bantu->next;         // menghubungkan ke node setelah yang dicari
    baru->next = bantu2;          // menghubungkan node baru
    bantu2->prev = baru;
    bantu->next = baru;           // menghubungkan ke node sebelum yang dicari
    baru->prev = bantu;
}
```

- ✓ Perlu diperhatikan :
  - Dibutuhkan satu pointer untuk membantu mencari node di mana data yang ingin disisipkan ditempatkan. Dalam code di atas digunakan pointer bantu.
  - Penggunaan pointer bantu2 pada code di atas sebenarnya bisa digantikan dengan pemanfaatan pointer bantu. Bagaimana caranya?
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di tengah. Misalkan saja data pada Linked List ada 4, lalu sisipkan data baru setelah node kedua.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menghapus data di depan

```
void deleteDepan()              // penghapusan data di depan
{
    TNode *hapus;

    if(isEmpty() == 0)          // jika data belum kosong
    {
        if(head->next != NULL)  // jika data masih lebih dari 1
    {
        hapus = head;           // letakan hapus pada head
        head = head->next;      // menggeser head (karena head harus ada)
        head->prev = NULL;      // head harus menuju ke NULL
        delete hapus;           //proses delete tidak boleh dilakukan jika node masih ditunjuk oleh pointer
    }
    else                        // jika data tinggal head
        {
        head = NULL;            // langsung diberi nilai NULL saja
        }
        printf("data terhapus\n");
    }
    else                        // jika data sudah kosong
    printf("DATA KOSONG\n");
}
```

- ✓ Perlu diperhatikan :
  - Dibutuhkan satu pointer untuk membantu memindahkan head ke node berikutnya. Dalam code di atas digunakan pointer hapus. Mengapa head harus dipindahkan?
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di depan. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.
- ✓ Fungsi untuk menghapus data di belakang

```
void deleteBelakang()      // penghapusan data di belakang
{
    TNode *hapus;

    if(isEmpty() == 0)           // jika data belum kosong
    {
        if(head->next != NULL)    // jika data masih lebih dari 1
    {
        hapus = head;                  // letakan hapus pada head
        while(hapus->next != NULL)
    {
        hapus = hapus->next;       // menggeser hingga node akhir
    }

        hapus->prev->next = NULL; // menghubungkan node sebelumnya dengan NULL
        delete hapus;//proses delete tidak boleh dilakukan jika node sedang ditunjuk oleh pointer
    }
    else            // jika data tinggal head
    {
        head = NULL;     // langsung diberi nilai NULL saja
    }
        printf("data terhapus\n");
    }
    else                // jika data sudah kosong
        printf("data kosong\n");

}
```

- ✓ Perlu diperhatikan :
  - Jika Linked List hanya menggunakan head, maka dibutuhkan satu pointer untuk membantu mengetahui node terakhir dari Linked List. Dalam code di atas digunakan pointer hapus.
  - Jangan lupa untuk tetap mengaitkan node terakhir ke NULL.
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di belakang. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menghapus data di tengah

```
void deleteTengah(int cari)          // penghapusan data di tengah
{
    TNode *hapus, *bantu, *bantu2;

    hapus = head;                  // letakan hapus pada hea
    while(hapus->data != cari)
    {
            hapus = hapus->next;      // menggeser hingga data cari
    }
    bantu2 = hapus->next;           // mengkaitkan node sebelum dan sesudahnya
    bantu = hapus->prev;
    bantu->next = bantu2;
    bantu2->prev = bantu;
    printf("data terhapus\n");
    delete hapus;                 //proses delete tidak boleh dilakukan jika node sedang ditunjuk oleh pointer
}
```

- ✓ Perlu diperhatikan :
  - Dibutuhkan satu pointer untuk membantu mencari node di mana data yang ingin dihapus ditempatkan. Dalam code di atas digunakan pointer hapus.
  - Penggunaan pointer bantu dan bantu2 pada code di atas sebenarnya bisa digantikan dengan pemanfaatan pointer hapus. Bagaimana caranya?
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di tengah. Misalkan saja data pada Linked List ada 4, lalu hapus data pada node ketiga.
  - Setelah memahami penggunaan fungsi ini, bukalah Codeblocks Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menghapus semua data

```
void clear()                    // penghapusan semua data
{
    TNode *bantu, *hapus;
    bantu = head;               // letakan bantu pada head
    while (bantu != NULL)       // geser bantu hingga akhir
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;          // delete satu persatu node
    }

    head = NULL;               // jika sudah habis berikan nilai NULL pada head
}
```

✓ Perlu diperhatikan :

- Dibutuhkan dua pointer untuk membantu menggeser dan menghapus, di mana dalam code di atas digunakan pointer bantu dan hapus.
- Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

✓ Fungsi untuk menampilkan semua data

```
void cetak()            // menampilkan semua data
{
    TNode *bantu;
    bantu = head;       // letakan bantu pada head

    if(isEmpty() == 0)
    {
        while (bantu != NULL)
    {
        printf("%d ", bantu->data); // cetak data pada setiap node
        bantu = bantu->next;        // geser bantu hingga akhir
    }
        printf("\n");
    }
    else
        printf("data kosong");    // jika data sudah kosong
}
```

✓ Perlu diperhatikan :

- Dibutuhkan satu pointer untuk membantu menggeser, di mana dalam code di atas digunakan pointer bantu.
- Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

2. DOUBLE LINKED LIST CIRCULAR

- Menggunakan 1 pointer head
- Head selalu menunjuk node pertama

Buatlah sebuah project baru dengan nama Guided2_DLLC_NIM. Kemudian ikuti dan cermati langkah-langkah berikut ini

Sebelumnya kita harus mendeklarasikan dulu pointer head :

```
TNode *head;
```

Setelah kita mendeklarasikan pointer head, kita belum bisa secara langsung mendeklarasikan node yang dituju. Sehingga pointer head harus dibuat bernilai *null* terlebih dahulu :

```
head = NULL;
```

untuk mengetahui apakah suatu Linked List kosong atau tidak, kita dapat mengetahuinya dengan mengecek nilai dari pointer Head-nya.

```
int isEmpty() {

      if(head==NULL) return 1;

      else return 0;

}
```

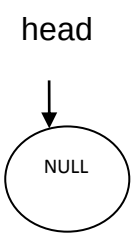
Contoh program :

- Penambahan di depan

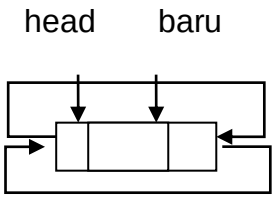
```
void tambahdata (int databaru){
    TNode *baru,*bantu;
//pointer bantu digunakan untuk menunjuk node terakhir (head->prev)
    baru = new TNode;
    baru -> data = databaru;
    baru -> next = baru;
    baru -> prev = baru;

    if (isEmpty()==1) {
        head=baru;
        head->next=head;
        head->prev=head;
    }
    else {
        bantu=head->prev;
        baru->next=head;
        head->prev=baru;
        head=baru;
        head->prev=bantu;
        bantu->next=head;
    }
    cout<<"data depan masuk";
}
```

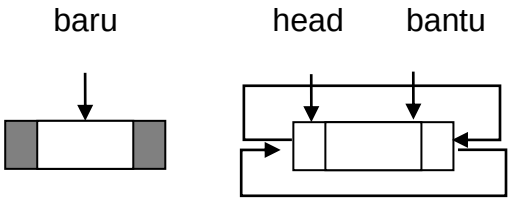
Penggambaran :

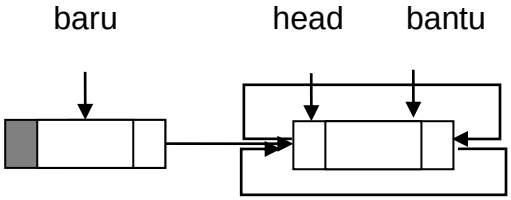


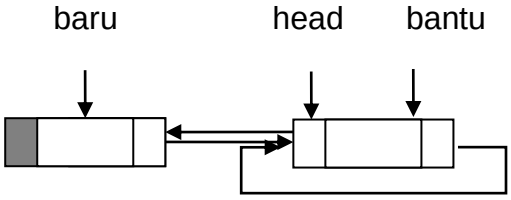
Setelah dibuat node baru dan jika diketahui head==NULL :

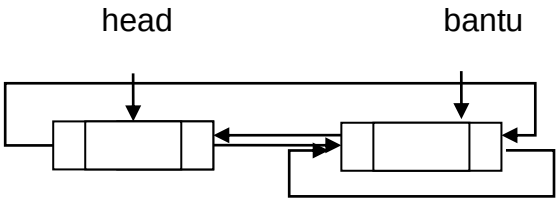


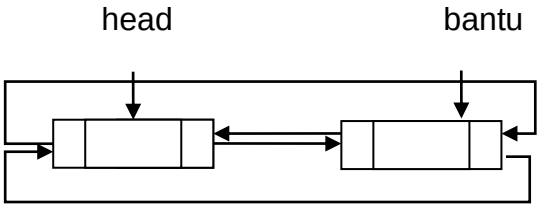
Bila kita membuat node baru lagi maka :











- Penambahan di belakang

```
void insertBelakang (int databaru){
    TNode *baru,*bantu;
    baru = new TNode;
    baru->data = databaru;
    baru->next = baru;
    baru->prev = baru;
        if(isEmpty()==1){
    head=baru;
    head->next = head;
    head->prev = head;
}
else {
    bantu=head->prev;
    bantu->next = baru;
    baru->prev = bantu;
    baru->next = head;
    head->prev = baru;
}
        cout<<"data belakang masuk";
}
```

- Tampil

```
void tampil(){
    TNode *bantu;
    bantu = head;
    if(isEmpty()==0){
        do{
            cout<<" i" <<bantu->data;
    bantu=bantu->next;
    }while(bantu!=head);
    printf("\n");
} else
    cout<<"Masih kosong\n";
}
```

- Hapus di depan

```
void hapusDepan (){
    TNode *hapus,*bantu;
    int d;
    if (isEmpty()==0){
    if(head->next != head){
    hapus = head;
    d = hapus->data;
    bantu = head->prev;
    head = head->next;
    bantu->next = head;
    head->prev = bantu;
    delete hapus;
} else {
    d = head->data;
    head = NULL;
}
        printf(" data depan terhapus",d);
} else
            printf("Masih kosong\n");
}
```

- Hapus di belakang

```
void hapusBelakang(){
    TNode *hapus,*bantu;
    int d;
    if (isEmpty()==0){
    if(head->next != head){
    bantu = head;
        while(bantu->next->next != head){
        bantu = bantu->next;
}
        hapus = bantu->next;
        d = hapus->data;
        bantu->next = head;
        delete hapus;
} else {
    d = head->data;
    head = NULL;
}
        printf(" data belakang terhapus\n",d);
} else
        cout<<"Masih Kosong";
}
```

GUIDED

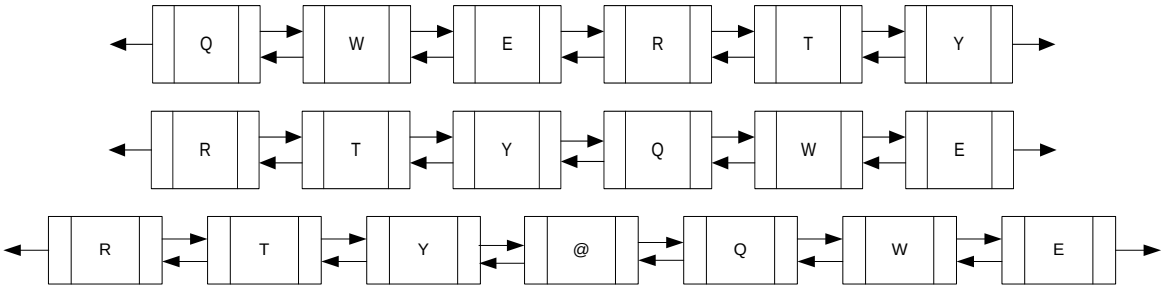
1. Setelah deklarasi node dilakukan pada project **Guided1_DLLNC_NIM**, dan semua fungsi sudah tersedia. Sekarang gabungkan setiap fungsi yang ada pada sebuah program penuh dengan spesifikasi :

- Pada program utama (main) berisi sebuah menu yang berisi fitur-fitur yang terdapat dari setiap fungsi yang sudah ada ada sebelumnya, yaitu : tambah data, hapus data, cek data kosong, dan cetak semua data.
- Pada struct hanya terdapat 1 tipe data saja yaitu integer.
- Sesuaikan fungsi-fungsi yang ada dengan program yang Anda buat (jangan langsung copy-paste dan digunakan).

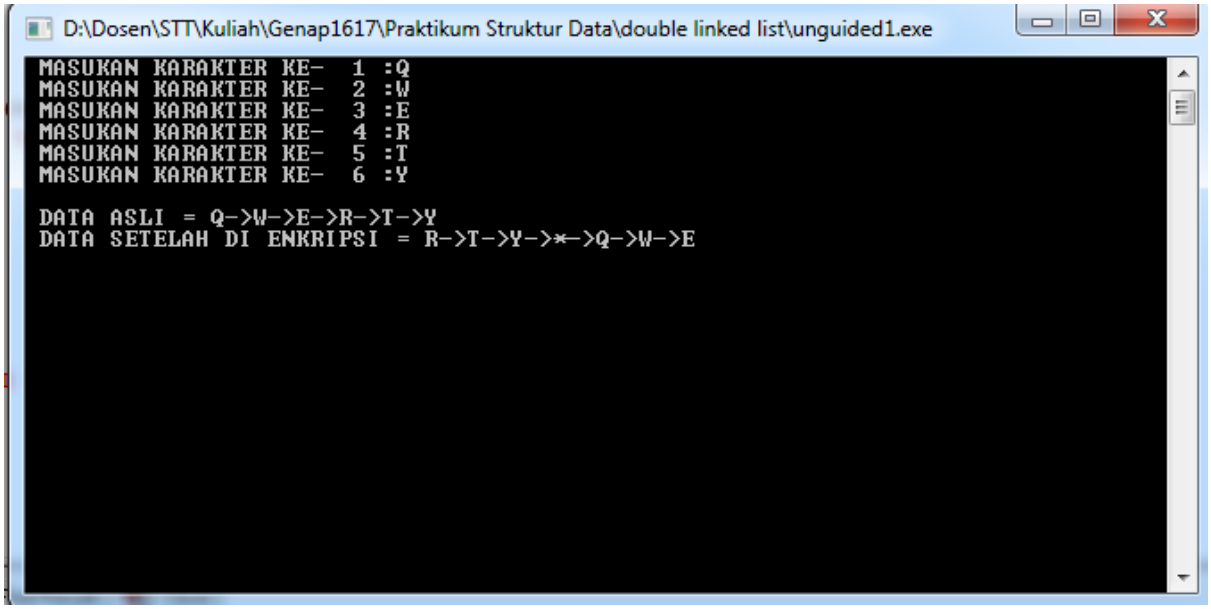
2. Dari project **Guided2_DLLC_NIM** Lanjutkan dengan membuat program lengkap dari potongan-potongan program yang ada diatas! Buat agar menjadi seperti menu.

Unguided

1. Buat program untuk enkripsi dan dekripsi password yang memanfaatkan Linked List, dengan spesifikasi :
   - Panjang password minimal 6 digit.
   - Isi password terserah dari user dan password diinputkan terlebih dahulu sebelumnya (penambahan data di belakang).
   - Enkripsi dilakukan dengan memindahkan 3 node terakhir, menjadi node terdepan. Kemudian sisipkan 1 karakter baru sebagai kunci setelah node ketiga dari yang dipindahkan tersebut.
   - Ilustrasi :



   - Lakukan juga proses dekripsi-nya.
   - Berikan juga fitur untuk menampilkan password



Resume

PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

S1 TEKNIK INFORMATIKA

Hari/Tanggal Praktikum : .....

Modul : .....

NIM : .....

Nama Praktikan : .....

Nama Asistant : 1.....

2.....

Nilai dan Parat : .....

Hasil Analisa Praktikum