MODUL

PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

PROGRAM STUDI S1 TEKNIK INFORMATIKA

ST3 TELKOM PURWOKERTO

2015

Algoritma dan Struktur Data

Tenia Wahyuningrum, S.Kom., MT

ST3 Telkom Purwokerto

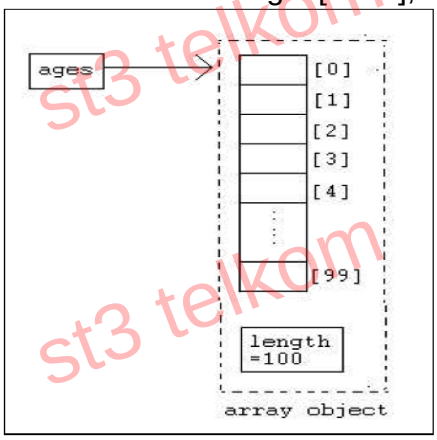
Jl. DI Panjaitan 128 Purwokerto

* Untuk kalangan sendiri

Praktikum 1
Materi : Array
Waktu : 100 menit

Dasar Teori

- Sebuah larik/array akan menyimpan beberapa item data dengan tipe data yang sama di dalam sebuah blok memori yang berdekatan yang kemudian dibagai menjadi beberapa slot. Cara penyimpanan [struktur data] inilah yang disebut sebagai array.
- Tipe data yang sama, disimpan dalam satu tempat yang sama dan diberi nomor indeks.
- Untuk mendeklarasikan array :
  tulis tipe datanya, diikuti dengan tanda kurung [],
  Contoh : int ages[100];



- Contoh :
  //memberikan nilai d3010 kepada elemen pertama array
  nim[0] = d3010;

  //mencetak elemen array yang ke 3
  cout<<nim[3];

Praktik

1. Buatlah variabel array untuk menampung kode, nama barang, jumlah, harga dan total yang sudah ditentukan sebagai berikut! Total mula-mula adalah 0, dan didapatkan dari jumlah * harga.

Kode	**Nama**	**Jumlah**	**Harga**	**Total**
001	Penghapus	4	1000	4000
002	Pensil	3	1500	4500
003	Buku	2	2000	4000
004	Rautan	3	1000	3000
005	Penggaris	5	500	2500

Jumlah item = 17
Total pembelian = 18000

2. Buatlah program untuk menghitung banyak data, rata-rata, jumlah dari sekumpulan data yang dimasukkan!

Tampilan

Masukan banyaknya data = 5
Data [1] = 5
Data [2] = 2
Data [3] = 4
Data [4] = 4
Data [5] = 5
Banyaknya data = 5
Rata-rata= 4
Jumlah = 20

3. Tambahkan standar deviasi pada soal no 3!
   Rumus standar deviasi (sd)

   Sigma= sigma+sqr(data[i]-rata);

   Sd=sqrt(sigma/n);

4. Buatlah program untuk mengalikan matriks !
   Syarat perkalian matriks :
   Jika matriks Am x n dan matriks Bp x q dikalikan, maka :
   Banyaknya kolom matriks A harus sama dengan banyaknya baris matriks B, sehingga n = p
   Matriks hasil perkalian antara A dan B adalah matriks dengan ordo m x q
   Perkalian dilakukan dengan menjumlahkan hasil kali setiap elemen baris matriks A dengan setiap elemen kolom matriks B yang sesuai

5. Buat resume praktikumnya, dan kumpulkan pada asisten / kirim via email !

Praktikum 2

Materi : Array 2 dimensi
Waktu : 100 menit

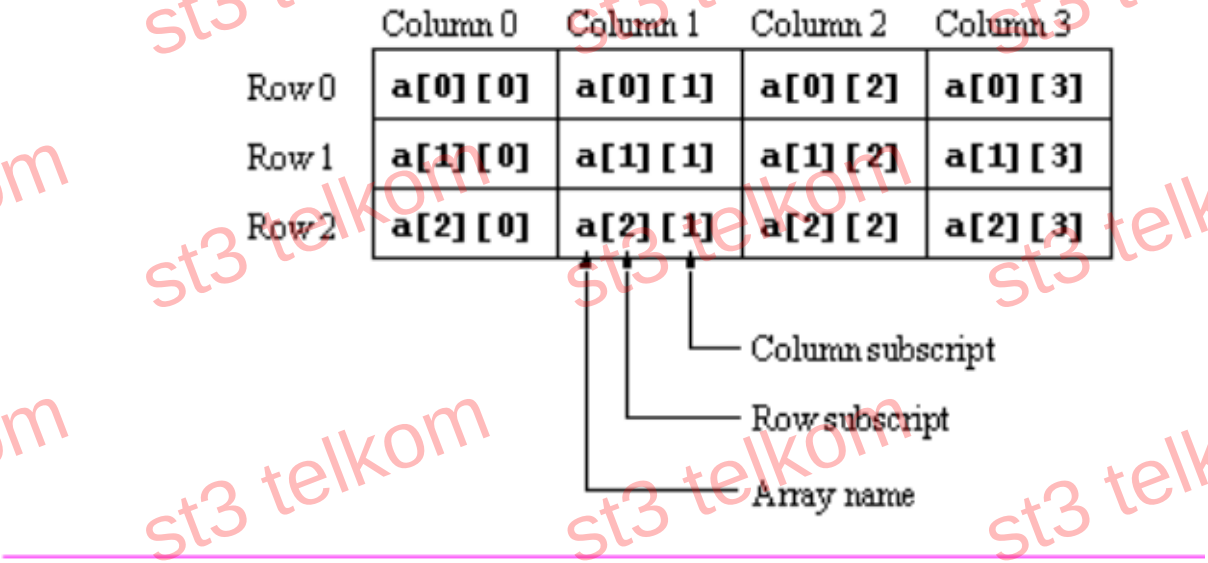
Dasar Teori

Array dua dimensi atau array multidimensi pada dasarnya sama dengan array satu dimensi, hanya saja, pada array multidimensi, indeksnya bisa lebih dari 1. Merupakan sebuah variabel yang menyimpan sekumpulan data yang memiliki tipe sama dan elemen yang akan diakses melalui banyak indeks atau subskrip. Array seperti ini biasa digunakan untuk matik, array 2 dimensi juga termasuk kedalam array multidimensi.

Array dua dimensi biasanya digunakan untuk merepresentasikan nilai dari sebuah tabel. mengidentifikasi tiap elemen array harus dispesifikasikan nilai baris dan kolom. . Array multidimensi sebenarnya adalah array dari array. Deklarasi array multidimensi dilakukan dengan adanya lebih dari satu pasangan kurung siku di dalam deklarasi array. [1]
Syntax untuk mendeklarasikan array 2 dimensi adalah: **type[,] <namaVariabel>;**

Type adalah tipe data dari array dan *<namaVariabel>* adalah nama dari array, sedangkan tanda [,] memberitahu C# untuk membuat variabel array 2 dimensi. Contoh : **byte[,] matrix;**

Pada contoh tersebut artinya membuat sebuah array 2 dimensi dengan nama matrix yang mempunyai tipe data byte. Sebagai ilustrasi, dapat dilihat pada gambar berikut.



Praktik
Buatlah algoritma dan program dengan struktur data array untuk menambah, mengurangi dan mengalikan 2 buah matriks !

Syarat perkalian matriks :
Jika matriks $A_{m \times n}$ dan matriks $B_{p \times q}$ dikalikan, maka :

- ✓ Banyaknya kolom matriks A harus sama dengan banyaknya baris matriks B, sehingga $n = p$
- ✓ Matriks hasil perkalian antara A dan B adalah matriks dengan ordo $m \times q$
- ✓ Perkalian dilakukan dengan menjumlahkan hasil kali setiap elemen baris matriks A dengan setiap elemen kolom matriks B yang sesuai

Praktikum 3
Materi : Pointer
Waktu : 100 menit

Dasar Teori
Pointer adalah variable yang berisi alamat memory sebagai nilainya dan berbeda dengan variable biasa yang berisi nilai tertentu. Dengan kata lain, pointer berisi alamat dari variable yang mempunyai nilai tertentu.

Dengan demikian, ada variabel yang secara langsung menunjuk ke suatu nilai tertentu, dan variabel yang secara tidak langsung menunjuk ke nilai.
Adapun bentuk umum dari pernyataan variabel pointer dalam C++ adalah :

Type *variabel-name

Dengan :

- Type adalah tipe dasar pointer
- Variabel name adalah nama variabel pointer
- * adalah variabel pada alamatnya yang ditentukan oleh operand.

Contoh :

```
Int *int_pointer;          // pointer to integer
Float *float_pointer;    // pointer to float
```

Contoh :

//Program : pointer.cpp

```
#include <stdio.h>

main()
{
        int a, *b;
        a=20;
        b=&a;
        printf (" Pointer b menunjukkan alamat =%p\n",b);
        printf (" Alamat tersebut berisi nilai :%d\n",*b);
}
```

//Program : pointer1.cpp

```
#include <iostream .h>

// cetak p dan *p
void main(void)
{
      int v = 7, *p;
      p = &v;
      cout << " Nilai v = " << v << " dan *p = " << *p
      << "\nAlamatnya = " << p << '\n';
}
```

```
Bila program diatas dijalankan, maka hasilnya adalah sebagai
berikut :
            Nilai v = 7 dan *p = 7
            Alamatnya = efffb24
```

//Program:pointer2.cpp

```
#include <iostream.h>

int main ()
{
      int value1 = 5, value2 = 15;
      int * mypointer;

      mypointer = &value1;
      *mypointer = 10;
      mypointer = &value2;
      *mypointer = 20;
      cout << "value1==" << value1 << "/ value2==" << value2;
      return 0;
}
```

Bila program diatas dijalankan, maka hasilnya adalah sebagai berikut :

“value1==” 10 << “/ value2==20

Praktik

1. Buatlah program untuk menghitung banyaknya karakter yang dimasukkan dengan menggunakan pointer.
2. Buatlah program untuk merubah karakter yang dimasukkan dari huruf kecil menjadi huruf besar.

Praktikum 4

Materi : Structure
Waktu : 100 menit

Dasar Teori

Bahasa pemrograman bisa memiliki tipe data:

– **Built-in : sudah tersedia oleh bahasa pemrograman tersebut**
  - Tidak berorientasi pada persoalan yang dihadapi

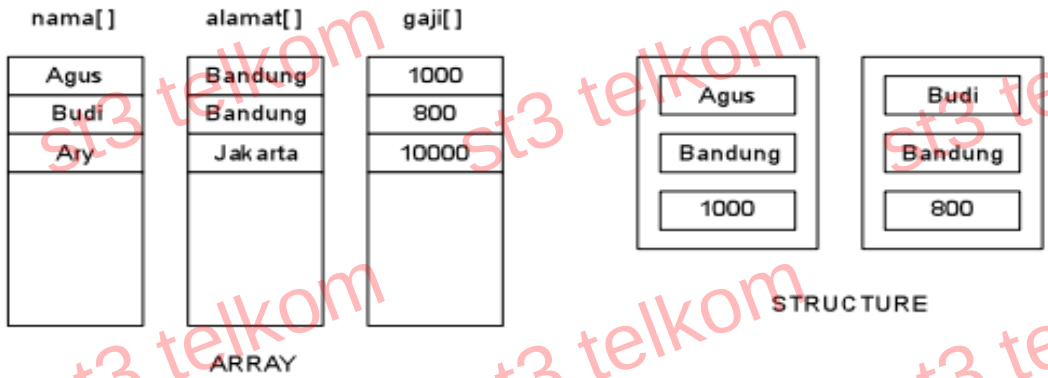
– **UDT : User Defined Type, dibuat oleh pemrogram.**
  - Mendekati penyelesaian persoalan yang dihadapi.
  - Contoh: record pada Pascal, struct pada C/C++, class pada Java

– **ADT : Abstract Data Type**
  - Memperluas konsep UDT dengan menambahkan pengkapsulan atau enkapsulasi, berisi sifat-sifat dan operasi-operasi yang bisa dilakukan terhadap kelas tersebut.
  - Contoh: class pada Java

adalah kumpulan data yang saling berhubungan, yang disimpan dalam satu unit penyimpanan.
Contoh : Data pegawai,
  - nama,
  - alamat ,
  - gaji.

- Bila menggunakan array biasa, maka diperlukan tiga variable yang bebas satu dengan yang lain, yaitu variabel nama, alamat dan gaji.
- Dengan menggunakan structure, data tersebut diorganisasikan dalam satu kesatuan.

Array vs Structure



Praktik

a. Buatlah 2 buah structure yang dapat menyimpan data pembeli dan barang. Isikan dalam structure data-data sbb :

- Data pembeli

Kode_pbl	Status	Nama
P001	M	Diana
P002	M	Rina
P003	BM	Lina
P004	BM	Doni
P005	M	Dodi

- M= Member, pembeli yang telah memiliki kartu member
- BM = Bukan Member, pembeli yang belum memiliki kartu member

- Data barang

Kode_brg	Nama_brg	Harga
BRG001	Pensil	2000
BRG002	Buku tulis	3500
BRG003	Penghapus	1000
BRG004	Penggaris	1500
BRG005	Ballpoint	2500

b. Buatlah input sebagai berikut :
Kode pembeli : P001
Nama pembeli : _____(otomatis tampil di layar)
Status pembeli : _____(otomatis tampil di layar)

Kode barang : BRG001
Nama barang : _____ (otomatis tampil di layar)
Harga barang : _____ (otomatis tampil di layar)
Jumlah barang : 2
Sub Total : jumlah barang x harga barang

Total pembelian : jumlah seluruh total
Diskon : jika pelanggan adalah member, maka diskon 10% dari total pembelian
Pembelian diatas 3 pcs mendapat potongan harga Rp. 300
Kembali = jumlah bayar-total

c. Output : print out nota pembelian
TOKO INDO APRIL
Jl. DI Panjaitan 128 Purwokerto

Kode barang	Nama barang	Jumlah	Harga	Total
BRG001	Pensil	2	2000	4000
BRG002	Buku tulis	2	3500	7000
Subtotal				11000
Diskon				1100
Total				12100

Jumlah bayar	13000
Kembali	900

Data pembeli
Kode pembeli : p001
Nama pembeli : Diana
Status : Member

Praktikum 5
Materi : Linked List
Waktu : 100 menit

Dasar Teori

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Linked List sering disebut sebagai senarai berantai. Untuk menghubungkan satu node dengan node lainnya maka Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang menempati suatu lokasi memori secara dinamis yang terdiri dari beberapa field, minimal 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitari bertipe pointer sebagai penunjuk node selanjutnya. Array dan Linked List memiliki perbedaan sebagai berikut :

Array
Statis
Penambahan dan penghapusan data Terbatas
Random access
Penghapusan array tidak mungkin

Linked List
Dinamis
Penambahan dan penghapusan data tidak terbatas
Sequential access
Penghapusan mudah

Salah satu tipe Linked List yang sederhana yaitu Single Linked List. Single Linked List merupakan Linked List yang memiliki hanya satu pointer penunjuk dengan arah data hanya satu arah juga. Single Linked List memiliki 2 macam bentuk yaitu Non Circular dan Circular. Non Circular Linked List merupakan Linked List di mana antara kepala dan node terakhir tidak memiliki hubungan. Pada Linked List ini maka pointer terakhir selalu menunjuk NULL sebagai pertanda data terakhir dalam list-nya. Single Linked List Non Circular dapat digambarkan sebagai gerbong kereta api seperti berikut ini :

Langkah membuat dan operasi pada sebuah Linked List adalah sebagai berikut :
1. Mendeklarasikan struct node
2. Membuat node head
3. Menginisialisasi node head
4. Menambah node baru baik di depan maupun di belakang
5. Menghapus node

Linked List banyak dimanfaatkan pada pemrograman kecerdasan buatan, fuzzy, maze solving, dan sebagainya.

PROSEDUR PERCOBAAN

Kompile program berikut ini dan amati outputnya pada layar Anda. Perhatikan baik-baik pemanggilan dan penggunaan fungsi-fungsi serta prosedurnya agar dapat mengerjakan tugas yang diberikan !

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

using namespace std;

//global var/const
typedef struct TNode{
int data;
TNode *next;
};

TNode *head;  //head node
//proto func/proc

void initHead();
int isEmpty();
void insertDepan(int databaru);
void insertBelakang (int databaru);
void tampilList();
void hapusDepan();
void hapusBelakang();
void clearList();


//detil func/proc
//init head

void initHead()
{
 head = NULL;  //NULL <> null!!!
}

//cek list kosong atau tdk
int isEmpty()
{
    return (head == NULL) ? 1:0;
}
//tambah data di depan

void insertDepan(int databaru)
{
    TNode *baru;
    baru = new TNode;
    baru->data = databaru;
    baru->next = NULL;
if(isEmpty()==1)
    {
        head=baru;
        head->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
    cout<<"Data baru telah dimasukkan di depan\n";
```

```
}

//tambah data di belakang
void insertBelakang (int databaru)
{
    TNode *baru,*bantu;
    baru = new TNode;
    baru->data = databaru;
    baru->next = NULL;
    if(isEmpty()==1)
    {
    head=baru;
    head->next = NULL;
}
    else
    {
    bantu=head;
    while(bantu->next!=NULL)
    {
    bantu=bantu->next;
    }
    bantu->next = baru;
    }
    cout<<"Data baru telah dimasukkan di belakang\n";
}

//menampilkan list
void tampilList()
{
    TNode *bantu;
    bantu = head;
    if(isEmpty()==0)
    {
while(bantu!=NULL)
{
cout<<bantu->data<<"  ";
bantu=bantu->next;
}
cout<<"\n";
    }
    else
         cout<<"Masih kosong\n";
}

//hapus data terdepan
void hapusDepan()
{
TNode *hapus;
int d;
if (isEmpty()==0)
    {
        if(head->next != NULL)
        {
            hapus = head;
            d = hapus->data;
            head = head->next;
            delete hapus;
        }
        else
        {
            d = head->data;
            head = NULL;
}
        cout<<d<<" terhapus\n";
}
    else
         cout<<"Masih kosong\n";
}
```

```
//hapus data terakhir
void hapusBelakang()
{
       TNode *hapus,*bantu;
       int d;
       if (isEmpty()==0)
           {
        if(head->next != NULL)
            {
       bantu = head;
       while(bantu->next->next!=NULL)
       {

      bantu = bantu->next;
}
       hapus = bantu->next;
       d = hapus->data;
       bantu->next = NULL;
       delete hapus;
 }
  else
     {
       d = head->data;
       head = NULL;
       }
        cout<<d<<"%d terhapus\n";
}
    else
         cout<<"Masih kosong\n";
}

//clear semua node
void clearList()
{
       TNode *bantu,*hapus;
       bantu = head;
       while(bantu!=NULL)
           {
       hapus = bantu;
       bantu = bantu->next;
       delete hapus;
}
       head = NULL;}

//main prog
int main()
{
    cout<<"single linked list non circular\n1. inisialisasi head ... \t";
    initHead();
    cout<<"done\ntampilkan isi list :\n";
    tampilList() ;

    //entry data di depan
    cout<<"\n entri data di depan list\n";
    int data_baru;
    for(int i=1;i<=5;i++)
    {
     cout<<"masukkan data ke-"<<i<<" : ";
     cin>> data_baru;
     insertDepan(data_baru);
    }
    cout<<"tampilkan isi list :\n";
    tampilList() ;

    //entry data di belakang
    cout<<"\n entri data di belakang list\n";
    for(int i=1;i<=5;i++)
```

```
{
 cout<<"masukkan data ke- "<<i<<" : ";
 cin>>data_baru;
 insertBelakang(data_baru);
}


cout<<"tampilkan isi list :\n";
tampilList() ;

//hapus data di depan
cout<<"\nhapus 2 data terdepan\n";

for(int i=1;i<=2;i++)
{
     hapusDepan();
}
cout<<"tampilkan isi list :\n";
tampilList() ;

//hapus data di belakang
cout<<"\nhapus 2 data terakhir\n";

for(int i=1;i<=2;i++)
{
     hapusBelakang();
}
cout<<"tampilkan isi list :\n";

tampilList();
//clear smua list
cout<<"\n hapus semua node\n";
clearList();
cout<<"tampilkan isi list :\n";
tampilList();
getch();
return 0;
}
```

Praktik

1. Buatlah program menggunakan Single Linked List Non Circular untuk menyimpan

   Nama dan NIM data mahasiswa berikut ini :

Nama	NIM
Dede	12347867
Kiki	98765674
Nina	67453279
Andi	83450120

2. Hapus list Andi !
3. Tampilkan di layar hasilnya sbb :

PROGRAM SENARAI BERANTAI

Masukkan nama ke-1 : Dede
Masukkan NIM ke-1 : 12347867

Dst

DATA MAHASISWA

Nama	NIM
Dede	12347867
Kiki	98765674
Nina	67453279
Andi	83450120

Andi terhapus

DATA MAHASISWA

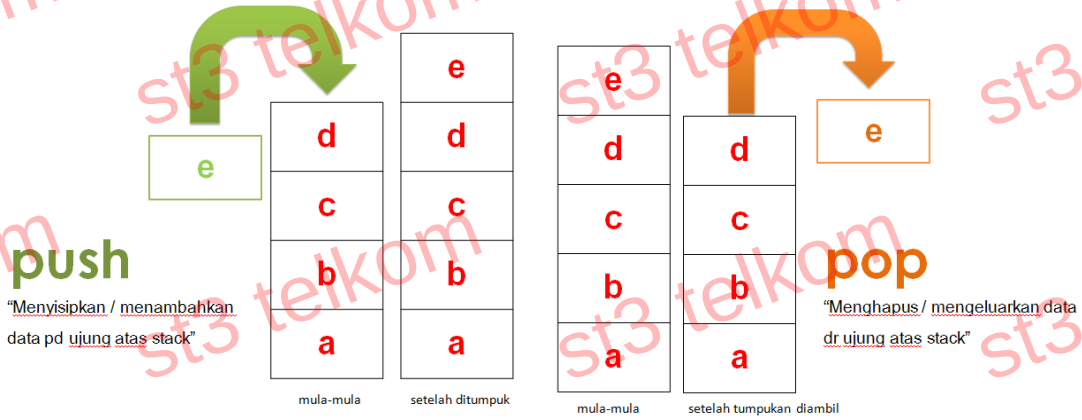
Nama	NIM
Dede	12347867
Kiki	98765674
Nina	67453279

Praktikum 6
Materi : Stack / Tumpukan

Waktu : 100 menit

Dasar Teori

"A stack is an ordered collection of items into which new items may be inserted and from which items may be deleted at one end, called top of the stack" * Yedidyah L, Moshe J. A., and Aaron M. Tenenbaum; Data Structures Using C and C++. Secara sederhana, tumpukan bisa diartikan sebagai suatu kumpulan data yang seolah-olah ada data yang diletakan diatas data yang lain. Satu hal yang perlu kita ingat adalah bahwa kita bisa menambah (menyisipkan) data, dan mengambil (menghapus) data lewat ujung yang sama, yang disebut sebagai ujung atas tumpukan (*top of stack)*. **LIFO (Last In First Out)** adalah sifat dari stack data yang disimpan terakhir akan diambil lebih dahulu, data yang disimpan pertama kali akan diambil paling akhir"



Operasi Stack, Push and Pop

Praktik :

1. Buatlah program untuk melakukan pembalikan terhadap kalimat dengan menggunakan stack.

   Contoh:
   Kalimat : Struktur Data
   Hasil setelah dibalik : ataD rutkurtS

2. Dari soal no 1, buatlah program untuk menentukan apakah sebuah kalimat yang diinputkan dalam program (dengan menggunakan stack) adalah sebuah palindrom atau bukan. Palindrom adalah kalimat yang jika dibaca dari depan dan dari belakang, maka bunyinya sama.

   Contoh:
   Kalimat : sugus
   Kalimat tersebut adalah palindrom

   Kalimat : tenia
   Kalimat tersebut bukan palindrom

Algoritma :

1. Mulai
2. Masukkan kata
3. Hitung jumlah hurufnya
4. Masukkan ke dalam stack (push)
5. Bandingkan elemen 1 dalam stack dengan elemen terakhir (pop)
6. Perbandingan dilakukan berulang sebanyak jumlah huruf
7. Jika huruf yang dibandingkan semuanya sama, maka kata tersebut adalah palindrome
8. Selesai

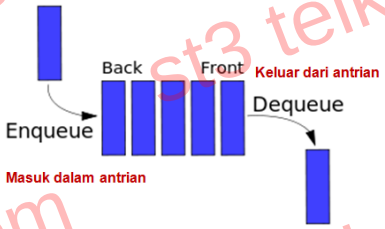
3. Buatlah program dengan stack untuk mengubah notasi matematika infix menjadi postfix !

Praktikum 7

Materi : Queue / Antrian
Waktu : 100 menit

Dasar Teori

Queue bersifat FIFO (First In First Out) yaitu elemen pertama yang ditempatkan pada queue adalah yang pertama dipindahkan.



Representasi Antrian

Operasi-operasi antrian

- CREATE
  Untuk menciptakan dan menginisialisasi queue dengan cara membuat Head dan Tail = -1
- ISEMPTY
  Untuk memeriksa apakah queue kosong
- ISFULL
  Untuk memeriksa apakah queue sudah penuh
- ENQUEUE
  Untuk menambahkan item pada posisi paling belakang
- DEQUEUE
  Untuk menghapus item dari posisi paling depan
- CLEAR
  Untuk mengosongkan queue

Praktik

1. Compile program dibawah ini !
2. Berikan penjelasan pada masing-masing fungsi yang terdapat pada program. Jelaskan apa kegunaan masing-masing fungsi !
3. Tambahkan fasilitas untuk menghitung banyaknya data, jumlah data, dan rata-rata dari keseluruhan data yang masuk ke dalam antrian !

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 8
typedef struct{
int data[MAX];
int head;
int tail;
} Queue;

Queue antrian;
void Create(){
antrian.head=antrian.tail=-1;
```

```
}
int IsEmpty(){
if(antrian.tail==-1)
return 1;
else
return 0;
}
int IsFull(){
 if(antrian.tail==MAX-1) return 1;   else return 0;  }


void Enqueue(int data){
if(IsEmpty()==1){
antrian.head=antrian.tail=0;
antrian.data[antrian.tail]=data;
printf("%d masuk!",antrian.data[antrian.tail]);
void Tampil();
{

if(IsEmpty()==0){
for(int i=antrian.head;i<=antrian.tail;i++){
printf("%d ",antrian.data[i]);
}


 }else printf("data kosong!\n");  }

} else
if(IsFull()==0){
antrian.tail++;
antrian.data[antrian.tail]=data;
printf("%d masuk!",antrian.data[antrian.tail]);
}
}

int Dequeue(){
int i;
 int e = antrian.data[antrian.head];  for(i=antrian.head;i<=antrian.tail-
1;i++){
antrian.data[i] = antrian.data[i+1];
}
antrian.tail--;
return e;
}
void Clear(){
antrian.head=antrian.tail=-1;
printf("data clear");
}
void Tampil(){


if(IsEmpty()==0){
for(int i=antrian.head;i<=antrian.tail;i++){
printf("%d ",antrian.data[i]);
jum=jum+antrian.data[i];

}
```

```
 }else printf("data kosong!\n");  }
int main(){
int pil;
int data;
Create();
do{
system ("CLS");

printf("1. Enqueue\n");
printf("2. Dequeue\n");
printf("3. Tampil\n");
printf("4. Clear\n");
printf("5. Exit\n");
 printf("Pilihan = ");scanf("%d",&pil);  switch(pil){
  case 1: printf("Data = ");scanf("%d",&data);  Enqueue(data);
break;
  case 2: printf("Elemen yang keluar : %d",Dequeue());  break;
  case 3: Tampil();  break;
  case 4: Clear();  break;
}
getch();
} while(pil!=5);
}
```

Praktikum 8
Materi : Double Linked List
Waktu : 100 menit

Dasar Teori

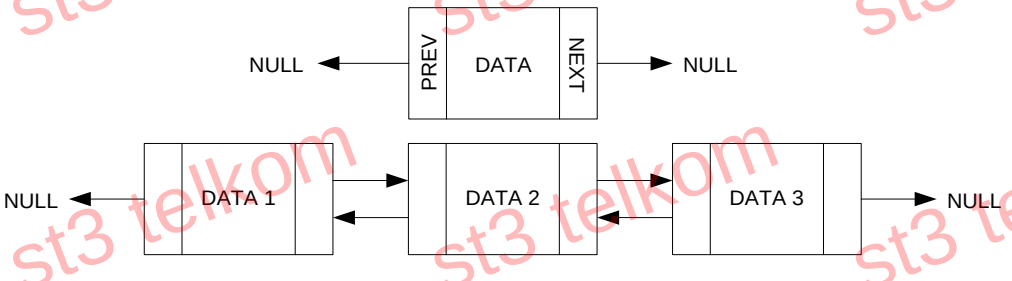
- Pada dasarnya, penggunaan Double Linked List hampir sama dengan penggunaan Single Linked List yang telah kita pelajari pada materi sebelumnya. Hanya saja Double Linked List menerapkan sebuah pointer baru, yaitu **prev**, yang digunakan untuk menggeser mundur selain tetap mempertahankan pointer **next**.
- 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.

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

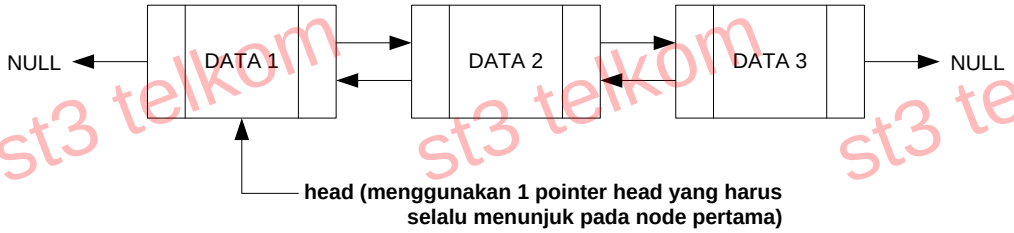


c. PEMBUATAN DLLNC

- **Deklarasi Node**

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

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



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

```
void init()                                          // inisialisasi awal
{
      TNode *head;
      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 head
      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 Turbo C++ 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                                            // jika data sudah kosong
          printf("data kosong");
}
```

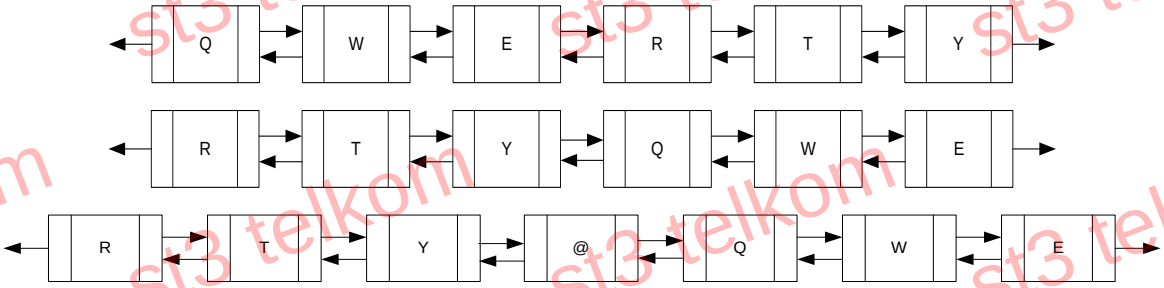
✓ Perlu diperhatikan :

- 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.

Praktik

1. Setelah deklarasi node dilakukan, 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. 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 (kunci) setelah node ketiga dari yang dipindahkan tersebut.
- Ilustrasi :



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

Praktikum 9
Materi : Double Linked List Circular
Waktu : 100 menit

Dasar Teori
DOUBLE LINKED LIST CIRCULAR

- Menggunakan 1 pointer head
- Head selalu menunjuk node pertama

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;
}
```

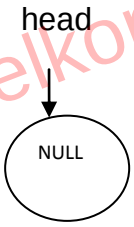
Contoh program :

- Penambahan di depan

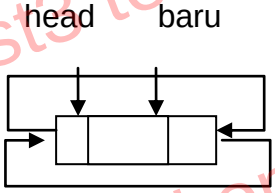
```
void tambahdata (int databaru){
  TNode *baru,*bantu;
  //pointer bantu digunakan untuk menunjuk node terakhir (head->prev)
  baru = new TNode;
  baru -> data = databaru;
  baru -> next = baru;
  baru -> prev = baru;

  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;
        }
        printf("data masuk");
    }
```

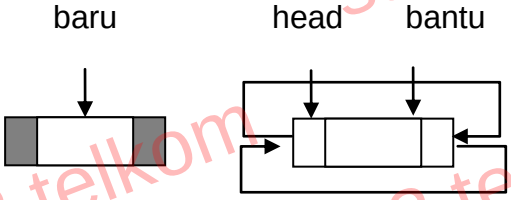
Penggambaran :

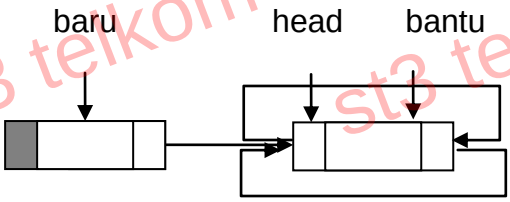


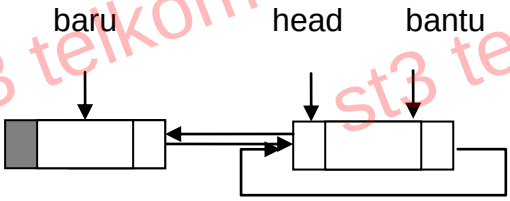
Setelah dibuat node baru dan jika diketahui head==NULL :

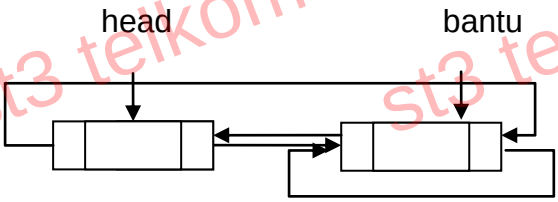


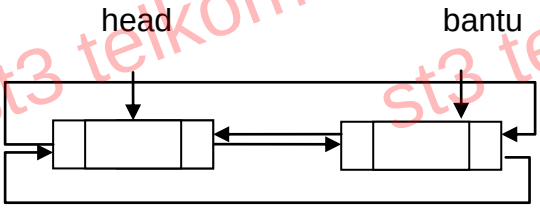
Bila kita membuat node baru lagi maka :











- 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;
    }
    printf("data masuk");
}
```

- Tampil

```
void tampil(){
    TNode *bantu;
    bantu = head;
    if(isEmpty()==0){
        do{
            printf("%i ",Bantu->data);
            bantu=bantu->next;
        }while(bantu!=head);
        printf("\n");
    } else printf("masih Kosong");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("%i 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("%i terhapus\n",d);
    } else printf("Masih Kosong");
}
```

Praktik

- Buatlah ilustrasi dari masing-masing potongan program.
- Buat program lengkap dari potongan-potongan program yang ada diatas! Buat agar menjadi seperti menu.
- Buat program untuk memasukkan node baru tetapi diantara node yang sudah ada. Tentukan node yang baru akan berada pada antrian keberapa.