MAKALAH STRUKTUR DATA

Di Susun Oleh :

NIDN 0406096803
Nama : A n u s Wuryanto,Mkom

Akademi Manajemen Informatika dan Komputer

Bina Sarana Informatika

2020

DAFTAR ISI

KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, dengan ini kami panjatkan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, taufiq serata inayah-Nya kepada kami, sehingga kami dapat menyelesaikan makalah Struktur Data.

Adapun makalah Struktur ini telah kami usahakan semaksimal mungkin. Dan kami ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu kami dalam pembuatan makalah ini, antara lain:

1. Kepada BSI Group yang telah banyak membantu.
2. Istri yang selalu mendukung dan mensuport tercapai makalah ini.

Akhirnya penyusun mengharapkan semoga dari makalah Struktur ini dapat diambil manfaatnya sehingga dapat memberikan inspirasi terhadap pembaca. Selain itu, kritik dan saran dari Anda kami tunggu untuk perbaikan makalah ini nantinya.

Bekasi, 20 Nopember 2020

Penyusun

BAB 1

1.1 Latar Belakang

Pemrograman dalam struktur data ada beberapa macam. Salah satunya adalah pemrograman C++. Pemakaian struktur data yang tepat di dalam proses pemrograman akan menghasilkan algoritma yang lebih jelas dan tepat, Struktur Data adalah : suatu koleksi atau kelompok data yang dapat dikarakteristikan oleh organisasi serta operasi yang didefinisikan terhadapnya.

Pemakaian Struktur Data yang tepat didalam proses pemrograman, akan menghasilkan Algoritma yang lebih jelas dan tepat sehingga menjadikan program secara keseluruhan lebih sederhana.

1.2 Kontrak Perkulihan

- ✓ Struktur Data merupakan Mata Kuliah yang diambil oleh mahasiswa di semester 2 (dua) dengan 4 (empat) sks dalam bentuk tatap muka
- ✓ Pertemuan 1-6 dilakukan seperti biasanya dimana dosen menyampaikan materi kepada mahasiswa
- ✓ Pertemuan 7 diadakan QUIZ / review materi
- ✓ Pertemuan 8 diadakan UTS dimana materi diambil dari pertemuan 1-6
- ✓ Pertemuan 9-11 yang diadakan setelah UTS dilakukan sama seperti Pertemuan 1-6 dimana dosen menyampaikan materi kepada mahasiswa.
- ✓ Pertemuan 12-14 mahasiswa secara berkelompok diwajibkan untuk membuat dan mempresntasikan Project program sebagai implementasi dari materi yang sudah diberikan.
- ✓ Pertemuan 15 diadakan QUIZ / review materi
- ✓ Pertemuan 16 diadakan UAS dimana materi diambil dari pertemuan 9-11 sebanyak 80% dan pertemuan 1-6 sebanyak 20%

1.3 Batasan Masalah

Dalam pembatasan masalah penuyusun hanya menulis makalah ini sebatas untuk mata kuliah struktut data

1.4 Tujuan makalah

Mahasiswa diharapkan mampu:

- Memahami sistem pengorganisasian data pada memori komputer dan file (berkas) pada media penyimpanan.
- Mengimplementasikannya dalam program dengan menggunakan salah satu bahasa pemrograman generasi ke-3 (Bahasa C) untuk membuat berbagai macam struktur data (array, tree, struct) dengan teknik-teknik tertentu (linked list, stack, dan queue) serta manipulasinya (sorting dan searching) secara baik, efisien, dan cepat.

BAB II

PEMBAHASAN

2.1 Arti Strukur data

Pemrograman dalam struktur data ada beberapa macam. Salah satunya adalah pemrograman C++. Pemakaian struktur data yang tepat di dalam proses pemrograman akan menghasilkan algoritma yang lebih jelas dan tepat, sehingga program menjadi lebih efisien dan sederhana

Pada garis besarnya, Data dapat dikategorikan menjadi :

A. Type Data Sederhana / Data Sederhana
   Terdiri dari :
   1. Data Sederhana Tunggal
      Misalnya : Integer, Real/Float, Boolean dan Character
   2. Data Sederhana Majemuk
      Misalnya : String

B. Struktur Data
   Terdiri dari :
   1. Struktur Data Sederhana
      Misalnya Array dan Record
   2. Struktur Data Majemuk
      Terdiri dari :
      a. Linier
         Misalnya : Stack, Queue dan Linear Linked List.
      b. Non Linier
         Misalnya : Pohon (Tree), Pohon Biner (Binary Tree), Pohon Cari Biner (Binary Search Tree), General Tree serta Graph.

TYPE DATA SEDERHANA
(Dalam Program C++)

1. **INTEGER**
   Merupakan Bilangan Bulat dan tidak mengandung pecahan. seperti : ...-3,-2,-1,0,1,2,3,....

Type data Integer

Type	Range	Ukuran (Byte)
Integer	- 32768..32767	2
Long	- 2147483648..2147483647	4

2. **FLOAT**

Type data yang merupakan bilangan pecahan.
Jenis Data float ditulis dgn menggunakan titik(koma) desimal.
Misalnya : 0.32 4,35 -131.128

Type Real dapat juga ditulis dengan Rumus :

$$M * R^e = X$$

M = Pecahan, R = Radix,
e = Exponen, X = Hasil Bilangan,

Misalnya : $3.2 * 10^{-1} = 0.32$
$4.35 * 10^{2} = 435$

Type data FLOAT

Type	Range	Ukuran (Byte)
Float	3.4×10^{-38} s/d $3.4 \times 10^{+38}$	4
Double	1.7×10^{-308} s/d $1.7 \times 10^{+308}$	8
Long Double	3.4×10^{-4932} s/d $1.1 \times 10^{+4932}$	10

3. **BOOL ATAU LOGICAL**

Type data yang hanya mempunyai dua bentuk keluaran yaitu nilai True dan False (Benar dan Salah) yang dinyatakan dengan 1 dan 0, Sehingga satuan data yang terpakai cukup satu bit saja. Operator yang digunakan adalah : And, Or dan Not.

Input			NOT (!)			AND (&&)	OR (||)
A	B	C	!A	!B	!C	A&&B&&C	A||B||C
0	0	0	1	1	1	0	0
0	0	1	1	1	0	0	1
0	1	0	1	0	1	0	1
0	1	1	1	0	0	0	1
1	0	0	0	1	1	0	1
1	0	1	0	1	0	0	1
1	1	0	0	0	1	0	1
1	1	1	0	0	0	1	1

4. CHARACTER

Type data yang terdiri dari aksara (simbol) yang meliputi *digit numerik*, *character alfabetik* dan *spesial character*. Untuk menuliskan tipe char, karakter perlu ditulis di dalam tanda petik tunggal (‘)

Contoh :

‘A’ → karakter berupa huruf A

‘1’ → karakter berupa angka 1

‘*’ → karakter simbol *

5. STRING

Merupakan type data majemuk yang terbentuk dari kumpulan character sebanyak 256 (default) dengan jangkauan niai 0 - 255. Kumpulan character yang digunakan untuk membentuk String dinamakan ***alfabet.*** Pemberian nilai String diapit dengan tanda petik ganda (“)

Bentuk umum penulisan tipe data ini adalah :

tipe_data pengenal [panjang] ;

pengenal = nama variabel

panjang = bilangan bulat yg menunjukan jumlah karakter

Contoh : char nama[15] ;

Fungsi pada Operasi STRING

1. Strcpy()
   untuk menyalin nilai string.
   Contoh dalam penggalan program c++:
   Cout<<“Masukan Kata ? “;gets(kata);
   Strcpy(copy,kata);
   Cout<<“Hasilnya ? “<<copy;
2. Strcat()
   untuk menggabungkan nilai string.
   Contoh dlm penggalan program c++:
   Cout<<“Kata Pertama ? “;gets(a);
   Cout<<“Kata Kedua ? “;cin(b);
   Strcat(a,b);
   Cout<<“Hasil Gabungan : “<<a;
3. Strcmp()
   untuk membandingkan 2 nilai string.
   Contoh dalam penggalan program c++:

```
char sa[]="Logika";
char sb[]="Logika Algoritma";
char sc[]="Logika Algoritma & Pemprograman";
/*Melakukan perbandingan terhadap dua string dan penampilan nilainya*/
printf("Nilai Yang dibandingkan sa,sb : %d\n",strcmp(sa,sb));
printf("Nilai Yang dibandingkan sa,sc : %d\n",strcmp(sa,sc));
printf("Nilai Yang dibandingkan sb,sa : %d\n",strcmp(sb,sa));
getch();
return 0;
}
```

4. Strlen()
   untuk mengetahui panjang nilai string

Contoh dalam penggalan program c++:

```
cout<<"Masukkan Kata = ";
gets(angka);
cout<<"Panjang Kata yang telah diinput = ";
cout<<strlen(angka);
```

5. Strchr ()

untuk mencari nilai karakter dalam string.

Contoh dalam penggalan program C++:

```
int main(void){
char str [100]="Aisyah Zahra";
char karakter='Z';
char *hasil;
hasil=strchr(str,karakter);
printf("Hasil Peubah :%s\n",hasil);
printf("Karakter %c ditemukan pada indeks ke-%d",karakter,(hasil-str));
getch();
return 0; }
```

Contoh Type data Sederhana

```
#include <conio.h>
#include <iostream.h>
#include <math.h>
void main()
        {
        int x,y,z;
        clrscr();
        cout <<“\n input nilai X=“; cin >> x;
        cout <<“\n input nilai Y=“; cin >> y;
        z = x + y;
        cout <<“\n hasil penjumlahan =“ << z;
        getch();
}
```

Latihan Soal :

- Buatlah dengan perkalian dan pengurangan dari Source program di atas kemudian anda jumlah dari ke perklain,penjumlahan dan bagi menjadi total seluruhnya

2.2 Pengertian Stack (Tumpukan)

Pengertian Stack atau Tumpukan adalah suatu stuktur data yang penting dalam pemrograman yang mempunyai sifat LIFO (Last In First Out), Benda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari stack. Stack (Tumpukan) adalah list linier yang dikenali elemen puncaknya (TOP) dan Aturan penyisipan dan penghapusan elemennya tertentu. Penyisipan selalu dilakukan "di atas" TOP dan Penghapusan selalu dilakukan pada TOP.

Dalam Kehidupan sehari-hari kita terdapat banyak kejadian yang mempunyai sifat seperti stack. Untuk lebih memahami pengertian stack kita ambil contih sebagai berikut,;

Perhatikan sebuah tumpukan piring disebuah warung makan,Piring-piring tersebut tersususn rapat dari atas ke bawah (membentuk barisan berurutan). Setiap kali ada pembeli datang maka piring yang paling atas akan diambil (menghapus elemen) yang berarti mengurangi jumlah piring dalam tumpukan. Bila tumpukan tersebut sudah habis atau tinggal sedikit, maka pegawai waru g akan menambahkan piring lain yang masih bersih (menambah elemen). Piring yang terakhir kali dimasukan pasti akan terletak ditumpukan paling atas, dan piring yang terletak dalam tumpukan paling itu pasti merupakan piring yang teakhir kali dimasukan.

Kesimpulannya adalah Penambahandan Penghapusan elemen hanya dapat dilakukan diatas ujung tumpukan piring. Proses seperti ini biasa disebut Last In-First Out (LIFO). Tumpukan hanya bisa menambah atau mengambil dari sebuah kotak lewat satu ujung, yaitu ujung bagian atas. Tumpukan merupakan kumpulan data yang sifatnya dinamis, artinya kita bisa menambah dan mengambil data darinya.

Dalam pemorgraman, koleksi data yang berstruktur stack dapat ditempatkan dalam aray satu dimensi atau dalam linear singly linked list. Untuk stack yang menggunakan aray satu dimensi kita mengenal single stack dan double stack.

1. Single Stack (Stack Tunggal)

   Adalah Stack ysng terdiri dari Satu collection. Bila stack tersebut menggunakan aray satu dimensi.
2. Double Stack (Stack Ganda)

   Satu aray digunakan untuk dua stack dimana dasar stack 1 berada pada sisi indeks yang terkecil dan dasar stack 2 berada pada sisi indeks stack yang tebesar.

A. OPERASI DASAR STACK

Adapun operasi-operasi dasar dari suatu stack adalah :

a) Create(Stack)

Operasi Create(Stack) digunakan untuk membuat suatu stack baru dengan nama stack, yang nilai elemen saat stack tersebut dibuat adalah NOEL(S) = 0, TOP(S) = NULL (tidak terdefinisikan)

b) IsEmpty(Stack)

Operasi ini merupakan operasi untuk mencek isi dari suatu stack dalam keadaan kosong atau berisi. Operasi ini memiliki 2 (dua) kondisi boolean yaitu :

- .True jika stack tersebut kosong atau dapat dikatakan NOEL(S) = 0
- .False jika stack tersebut tidak dalam kondisi kosong atau dapat dikatakan NOEL(S) > 0

c) Push(Stack, Elemen)

Operasi ini merupakan operasi untuk menambahkan satu elemen dengan nilai X pada puncak suatu stack, sehingga posisi TOP(S) akan bernilai X, penerapan operasi push pasa suatu stack S akan berakibat overflow jika NOEL(S) dari stack tersebut telah bernilai maksimum.

d) Pop(Stack)

Operasi ini berfungsi untuk menghapus satu elemen dari stack S, sehingga posisi NOEL(S) akan berkurang satu elemen, dan TOP(S) akan berubah. Operasi pop dapat menyebabkan kondisi underflow jika suatu stack S yang berada dalam kondisi minimum dikenakan operasi pop.

e) IsFULL
Operasi ini untuk memeriksa apakah Stack sudah penuh.

f) CLEAR
Operasi ini untuk mengosongkan stack.

B. NOTASI ARITMATIK (Infix, Prefix dan Postfix)

Salah satu manfaat tumpukan adalah menulis notasi aritmatik, yaitu Notasi Infix, Prefix dan notasi Postfix)

1. Notasi Infix

   Yaitu notasi yang mudah dimengerti oleh manusia.
   Contoh A + B (Operand Operator Operand)
   Hirarki Operator :
   a) Tanda kurng : (....)
   b) Ekponensial atau tanda pangkat : ^
   c) Perkalian, Pembagian : *, /
   d) Penjumlahan, Pengurangan : +, -

   Contoh (A-B) * (C + D)
   Prioritas pengerjaannya adalah :
   a. Dalam kurung yang pertama : (A – B)
   b. Dalam kurung yang kedua : (C + D)
   c. Perkalian hasil pengurangan dengan hasil penjumlahan

2. Notasi Prefix

   Yaitu notasi yang simbul Operatir diletakkan sebelum dua operand.
   Contoh : Infix à A + B
   Prefix à + AB (Operator Operand Operand)

3. Notasi Postfix

   Yaitu Notasi yang simbol operator diletakkan sesudah dua operand.
   Contoh : Infix à A + B
   POSTfix à AB – (Operand Operand Operator)
   Contoh : Infix ke Prefix

1. A+B

   Pengerjaan 1 : A + B Prefixnya : + AB

2. (A + B) – (C * D)

   a. Pengerjaan dalam kurung 1 : (A + B) Prefixnya : + AB
   b. Pengerjaan dalam kurung 2 : (C * D) Prefixnya : * CD
   c. Terakhir adalah Operator – : +AB – *CD Prefixnya : – +AB*CD

Contoh : Infix ke Postfix

1. A+B
   a. Pengerjaan 1 : A + B Prefixnya : AB+
2. (A + B) – (C * D)
   a. Pengerjaan dalam kurung 1 : (A + B) Postfixnya : AB+
   b. Pengerjaan dalam kurung 2 : (C * D) Postfixnya : CD*
   c. Terakhir adalah Operator – : AB+ – CD* Postfixnya : AB+ CD* –

INFIX	PREFIX	POSTFIX
A + B	+AB	AB+
A + B – C	-+ A B C	A B + C -
(A + B) * (C – D)	*+AB-CD	AB + CD-*
A – B / (C * D ^ E)	-A / B * C ^ D E	A B C D ^ */-

C. Contoh stack dalam borland c++

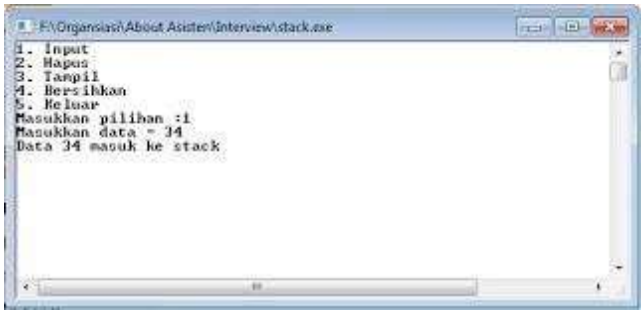
Koding program c++:

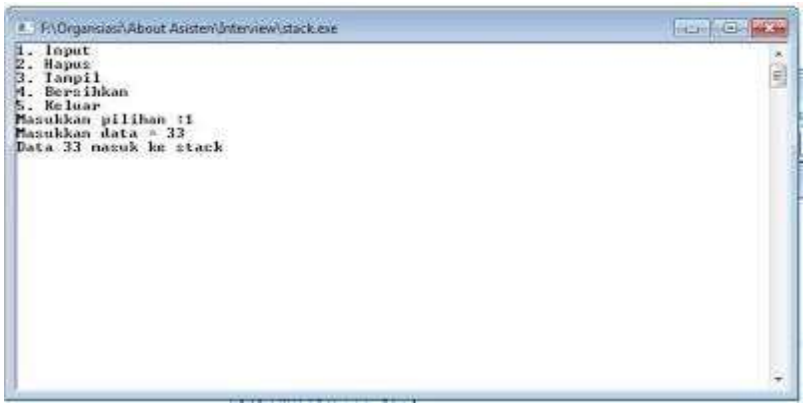
```
#include <iostream.h>
#include <conio.h>
#define max 10

struct Tumpukan{
int atas;
int data[max];
}T;

void awal(){
 T.atas=-1;
}
```

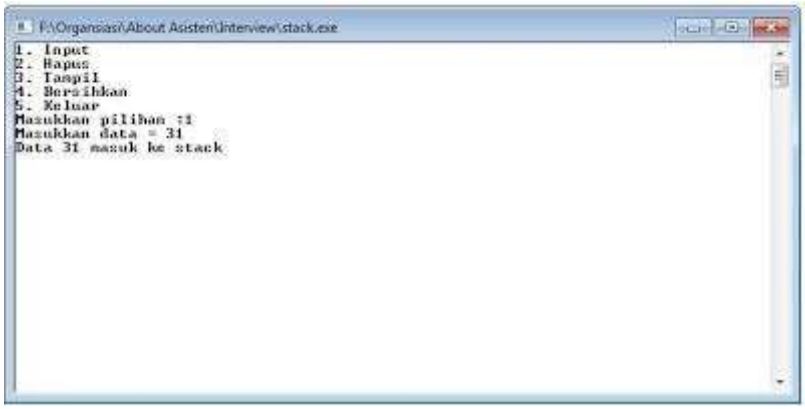
```
    int kosong(){
    if(T.atas==-1)
    return 1;
    else
    return 0;
    }

    int penuh(){
    if(T.atas==max-1)
    return 1;
    else
    return 0;
    }

    void input(int data){
    if(kosong()==1)
    {T.atas++;
    T.data[T.atas]=data;
    cout<<"Data "<<T.data[T.atas]<<" masuk ke stack";}
    else if(penuh()==0)
    {
    T.atas++;
    T.data[T.atas]=data;
    cout<<"Data "<<T.data[T.atas]<<" masuk ke stack";}
else
    cout<<"Tumpukan penuh";
    }

    void hapus(){
    if(kosong()==0){
    cout<<"Data teratas sudah terambil";
    T.atas--;
    }
     else
    cout<<"Data kosong";
    }

    void tampil(){
     if(kosong()==0)
    {for(int  i=T.atas;i>=0;i--)
    {cout<<"\nTumpukan ke "<<i<<"="<<T.data[i];}
    }
    else
    cout<<"Tumpukan kosong";
    }
void bersih(){
    T.atas=-1;
    cout<<"Tumpukan kosong!";
```

```
}
void main(){
int  pil,data;
awal();
do
{
clrscr();
cout<<"1. Input\n2. Hapus\n3. Tampil\n4. Bersihkan\n5. Keluar\nMasukkan pilihan :";
cin>>pil;
switch(pil)
{case 1:cout<<"Masukkan data = ";cin>>data;
input(data);
break;
 case 2:hapus();
 break;
 case 3:tampil();
 break;
 case 4:bersih();
 break;
 case 5: cout<<"Terimakasih, tekan enter untuk keluar";
 }
 getch();     }
 while(pil!=5); [=./}
```

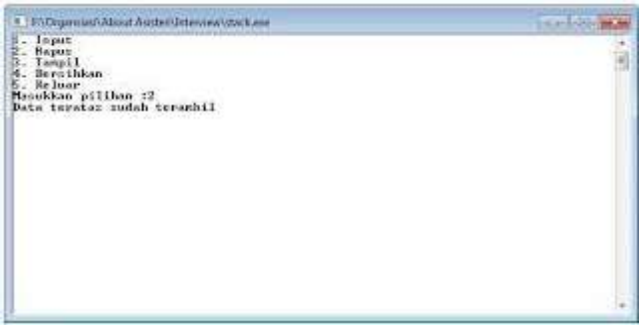
Screen Shot program sebagai berikut:




F:\Organisasi\About Asisten\Interview\stack.exe
1. Input
2. Hapus
3. Tampil
4. Bersihkan
5. Keluar
Masukkan pilihan :1
Masukkan data = 33
Data 33 masuk ke stack


F:\Organisasi\About Asisten\Interview\stack.exe
1. Input
2. Hapus
3. Tampil
4. Bersihkan
5. Keluar
Masukkan pilihan :1
Masukkan data = 31
Data 31 masuk ke stack


1. Input
2. Hapus
3. Tampil
4. Bersihkan
5. Keluar
Masukkan pilihan :3
Tumpukan ke 2=31
Tumpukan ke 1=33
Tumpukan ke 0=34


1. Input
2. Hapus
3. Tampil
4. Bersihkan
5. Keluar
Masukkan pilihan :2
Data teratas sudah terambil

2.3 Pengertian Queue (Antrian)

Struktur Data Antrean (Queue) adalah suatu bentuk khusus dari List Linier dengan operasi pemasukan data hanya diperbolehkan pada salah satu sisi, yang disebut sisi Belakang / ekor (Tail) dan operasi penghapusan hanya diperbolehkan pada sisi lainnya yang disebut sisi Depan / kepala (Head) dari LinkedList.

Prinsip Antrean :

- FIFO (First In First Out)
- FCFS (First Come First Serve)

"Yang Tiba lebih awal Maka akan dilayani Terlebih Dahulu"

Contoh Elustrasi :

- Contoh antrian :
  - Antrian printer
  - Antrian tiket bioskop
  - Antrian pada kasir sebuah bank.

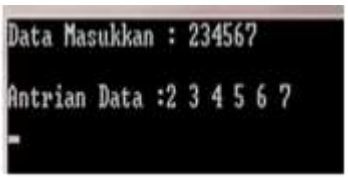
Keterangan :

Ketika seorang pelanggan datang, akan menuju ke belakang dari antrian. Setelah pelanggan dilayani, antrian yang berada di depan akan maju. Terdapat pada gambar di bawah ini :

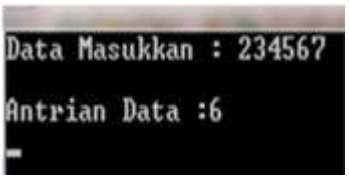
1. Gambar antrian di Bank

2. Gambar antrian di Loket pembayaran

3. Source program C++ tentang antrian



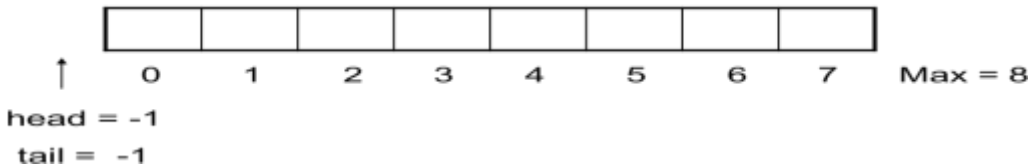
RUBAH PROGRAM TERSEBUT MENJADI ANTRIAN




```
#include<iostream.h>
#include<conio.h>
main()
{
char queue[5];
int i;

//Inputan Data
cout<<"Data Masukkan : ";
cin>>queue;

cout<<endl;

//Menampilkan data
cout<<"Pembalikan Data :";
for(i=5;i>=0;i++)
{
cout<<queue[i]<<" ";
}
cout<<endl;
getch();
}
```

DEKLARASI QUEUE

```
#define MAX 8
typedef    struct{
                    int data[MAX];
                    int head;
                    int tail;
                } Queue;

Queue antrian;
```



OPERASI QUEUE

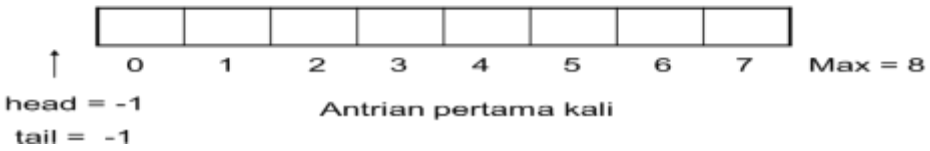
- **CREATE**
  Untuk menciptakan dan menginisialisasi Queue
  Dengan cara membuat Head dan Tail = -1
- **ISEMPTY**
  Untuk memeriksa apakah queue kosong
- **ISFULL**
  Untuk memeriksa apakah queue sudah penuh
- **ENQUEUE**
  Untuk menambahkan item pada posisi paling belakang
- **DEQUEUE**
  Untuk menghapus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

a). Fungsi Create

Fungsi Create

- Digunakan untuk membentuk dan menunjukan awal terbentuknya suatu Antrean / Queue

```
Void Create()
{
   antrian.head = antrian.tail = -1
}
```



b). Fungsi IsEmpty

Fungsi IsEmpty

- Untuk memeriksa apakah Antrian penuh atau kosong
- Dengan cara memeriksa nilai Tail, jika Tail = -1 maka antrian kosong (empty)
- 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

Contoh Elustrasi :

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

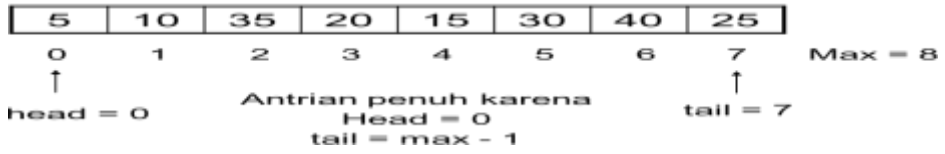
c). Fungsi isFull

Fungsi IsFull

- Untuk mengecek apakah Antrian sudah penuh atau belum
- Dengan cara :
  - Mengecek nilai Tail
  - Jika tail = MAX-1 berarti antrian sudah penuh (MAX-1 adalah batas elemen array dalam program C++)

Contoh : Elustrasi isfull

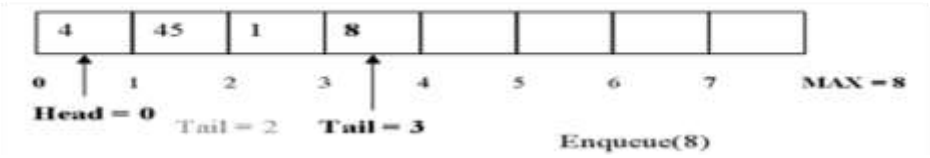
```
Int IsFull()
{
   if (antrian.tail == Max-1)
        return 1;
   else
        return 0;
}
```



d). Fungsi Engueue

Fungsi Enqueue

- Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu dilakukan pada elemen paling **belakang**
- Penambahan elemen selalu menggerakan variabel Tail dengan cara menambahkan Tail terlebih dahulu

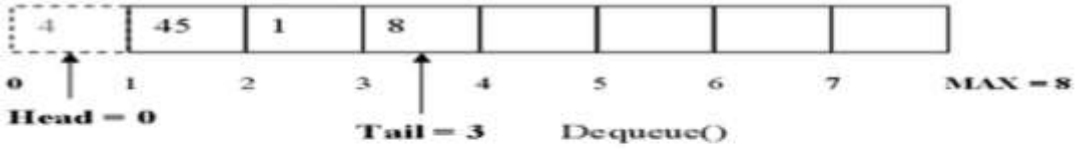


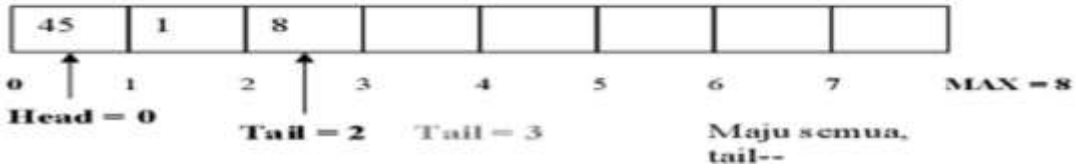
e). Fungsi Dequeue

Fungsi Dequeue

- Digunakan untuk menghapus elemen terdepan (head) dari Antrian
- Dengan cara : menggeser semua elemen antrian kedepan dan mengurangi Tail dgn 1. Penggeseran dilakukan dengan menggunakan looping

```
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;
}
```





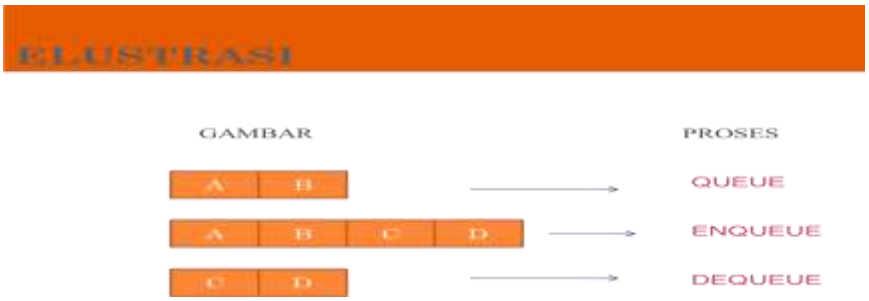
f). Fungsi Clear

Fungsi 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 sehingga mengembalikan antrian seperti keadaan semula

```
void Clear(){
      antrian.head=antrian.tail=-1;
      printf("data clear");
}
```

Antrian setelah di lakukan Clear



1. Contoh Soure Code pemograman Antriaan

```
#include<iostream.h>
#include<conio.h>
main()
{
char queue[6];
int i;

//Inputan Data
cout<<"Data Masukkan : ";
cin>>queue;

cout<<endl;

//Menampilkan data
```

```
cout<<"Antrian Data :";
for(i>0;i<=6;i++)
{
 i=(i+1)+3;
// i=(i+2)+2;

 cout<<queue[i]<<" ";
}
cout<<endl;
getch();
}
```

2. Contoh Soure Code pemograman Antriaan

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define MAX 6

typedef struct{
int data[MAX];
int head;
int tail;
}queue;
queue antrian;

void create()
{
antrian.head=antrian.tail=-1;
}

// kosong

int isempty() {
if(antrian.tail==-1)
return 1;
else
return 0;
}

// Penuh

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

```
// tambah
void enqueue(int data)
{
if(isempty()==1)
{
antrian.head=antrian.tail=0;
antrian.data[antrian.tail]=data;
printf("%d, Sudah Masuk!",antrian.data[antrian.tail]);


void tampil();
{
int i;
if (isempty()==0)
{
 for(i=antrian.head;i<=antrian.tail;i++)
 {
  printf(" %d ",antrian.data[i]);
  }
}
else
 printf("\n**** QUEUE IS EMPTY ****\n");
}
}
else
if(isfull()==0)
{
antrian.tail++;
antrian.data[antrian.tail]=data;
printf("%d , Sudah Masuk!",antrian.data[antrian.tail]);
}
  else{
  if(isfull()==1)
  {
  cout<<"\n\n**** QUEUE IS FULL , data TIDAK dapat masuk ****";
  }
  }
gotoxy(25,8);cout<<"PRESS any key for back to MENU";
}

// hapus

int dequeue()
{
if (isempty()==1){
cout<<"\n**** ERROR :: QUEUE IS EMPTY ****";
}else
```

```
if(isempty()==0){
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--;
cout<<"\n\nData Yang Keluar => "<<e;
}
gotoxy(25,8);cout<<"PRESS any key for back to MENU";
}

// Exit

void clear()
{
antrian.head=antrian.tail=-1;
printf("\n\n**** DATA CLEAR ****");
gotoxy(25,8);cout<<"PRESS any key for back to MENU";
}

// tampil

void tampil()
{
int i;
if(isempty()==0)
{
cout<<"Data Yang ada Dalam QUEUE : "<<endl<<endl;
for(i=antrian.head;i<=antrian.tail;i++)
{
printf("| %d |",antrian.data[i]);
}
}
else
{
printf("\n**** QUEUE IS EMPTY ****\n");
}
gotoxy(25,8);cout<<"PRESS any key for back to MENU";
}

void main()
{
int pil;
int data;
create();
do
```

```
{
clrscr();
gotoxy(25,2);cout<<"========MENU PILIHAN========"<<endl<<endl;
gotoxy(25,4);cout<<"============================"<<endl;
gotoxy(30,6);cout<<" 1. ENQUEUE "<<endl;
gotoxy(30,7);cout<<" 2. DEQUEUE "<<endl;
gotoxy(30,8);cout<<" 3. TAMPILAN "<<endl;
gotoxy(30,9);cout<<" 4. CLEAR "<<endl;
gotoxy(30,10);cout<<"5. KELUAR "<<endl;
gotoxy(25,12);cout<<"============================"<<endl;
gotoxy(25,14);cout<<" Masukan Pilihan Anda => ";cin>>pil;

switch(pil){
case 1:
clrscr();
printf("\n\n Masukan Data => "); scanf("%d",&data);
enqueue(data);
break;

case 2:
clrscr();
dequeue();
break;

case 3:
clrscr();
cout<<endl;
tampil();
break;

case 4:
clrscr();
clear();
break;

case 5:
clrscr();
gotoxy(25,8);cout<<"**** TERIMA KASIH ****"<<endl;
break;
}
getch();
} while(pil!=5);
}
```

2.3 Pengertian Kunjungan Pohon Biner

Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan

hubungan yang bersifat hirarkis (hubungan one to many) antara elemen-elemen. Tree bisa didefinisikan sebagai kumpulan simpul/node dengan satu elemen khusus yang disebut Root dan node lainnya. Tree juga adalah suatu graph yang acyclic, simple, connected yang tidak mengandung loop.

Sebuah binary search tree (bst) adalah sebuah pohon biner yang boleh kosong, dan setiap nodenya harus memiliki identifier/value. Value pada semua node subpohon sebelah kiiri adalah selalu lebih kecil dari value dari root, sedangkan value subpohon di sebelah kanan adalah sama atau lebih besar dari value pada root, masing-masing subpohon tersebut (kiri dan kanan) itu sendiri adalah juga binary search tree.

2.2.1 Kunjungan pohon biner

Kunjungan pada Pohon Binar merupakan salah satu operasi yang sering dilakukan pada suatu Pohon Binar tepat satu Kali (Binary Tree Traversal). Operasi ini terbagi menjadi 3 bentuk :

1. Kunjungan secara Preorder (Depth First Order), mempunyai urutan :
   a. Cetak isi simpul yang dikunjungi (Simpul Akar)
   b. Kunjungi Cabang Kiri
   c. Kunjungi Cabang Kanan

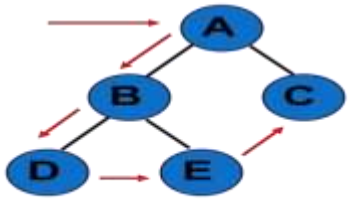
2. Kunjungan secara Inorder (Symetric Order), mempunyai urutan :
   a. Kunjungi Cabang Kiri
   b. Cetak isi simpul yang dikunjungi (Simpul Akar)
   c. Kunjungi Cabang Kanan

3. Kunjungan secara Postorder, mempunyai urutan :
   a. Kunjungi Cabang Kiri
   b. Kunjungi Cabang Kanan
   c. Cetak isi simpul yang dikunjungi (Simpul Akar)

Pada ketiga cara kunjungan diatas, kunjungan ke Cabang Kiri dilakukan terlebih dahulu, baru kemudian kunjungan ke Cabang Kanan. Dengan orientasi semacam ini, Ketiga kunjungan diatas disebut dengan Left To Right Oriented (LRO).

Jika kunjungan ke Cabang Kanan dilakukan lebih dahulu baru kemudian kunjungan ke Cabang Kiri, maka Orientasi semacam ini disebut Right To Left Oriented (RLO).

1. Kunjungan secara Preorder (Depth First Order), mempunyai urutan :
   a. Cetak isi simpul yang dikunjungi (Simpul Akar)
   b. Kunjungi Cabang Kiri
   c. Kunjungi Cabang Kanan

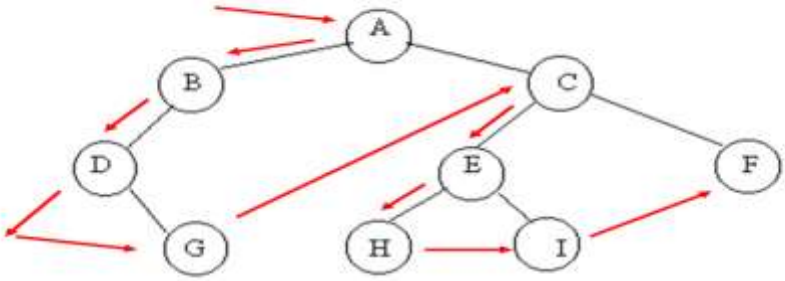


A B D E C

Klik Animasi

Contoh :

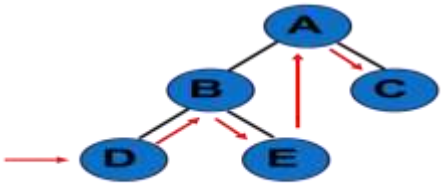
Kunjungan PreOrder



Hasil : A B D G C E H I F

Klik Animasi

2. Kunjungan secara Inorder (Symetric Order), mempunyai urutan :
   a. Kunjungi Cabang Kiri
   b. Cetak isi simpul yang dikunjungi (Simpul Akar)
   c. Kunjungi Cabang Kanan

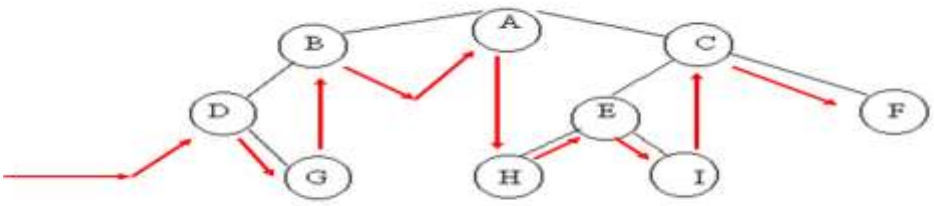


D B E A C

Klik Animasi

Contoh : Kunjungan in Order

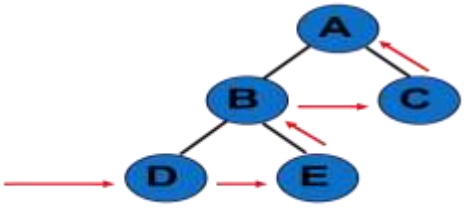
Kunjungan InOrder



Hasil : D G B A H E I C F

Klik Animasi

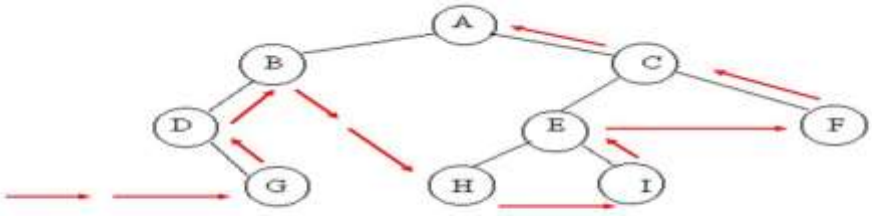
3. Kunjungan secara Postorder, mempunyai urutan :
   a. Kunjungi Cabang Kiri
   b. Kunjungi Cabang Kanan
   c. Cetak isi simpul yang dikunjungi (Simpul Akar)



D E B C A

Klik Animasi

Kunjungan PostOrder



Hasil : G D B H I E F C A

Klik Animasi

2.2.2 Kunjungan LevelOrder

Selain kunjungan yang dijelaskan diatas, masih ada satu macam kunjungan masih ada satu macam kunjungan lagi yaitu kunjungan LevelOrder.
Kunjungan dimulai dari simpul yang ada pada tingkat 1 (Akar), diteruskan pada simpul di tingkat 2, tingkat 3 dan seterusnya.

Secara singkat kunjungan Level Order ini dapat dijelaskan sebagai berikut:

1. Dimulai dengan memasukkan Akar kedalam antrean.
2. Kemudian mengeluarkan Akar tersebut keluar dari antrean.

Pada saat Akar tersebut dikeluarkan dari antrean, cabang kiri dan cabang kanan secara berturut-turut dimasukkan dalam antrean.

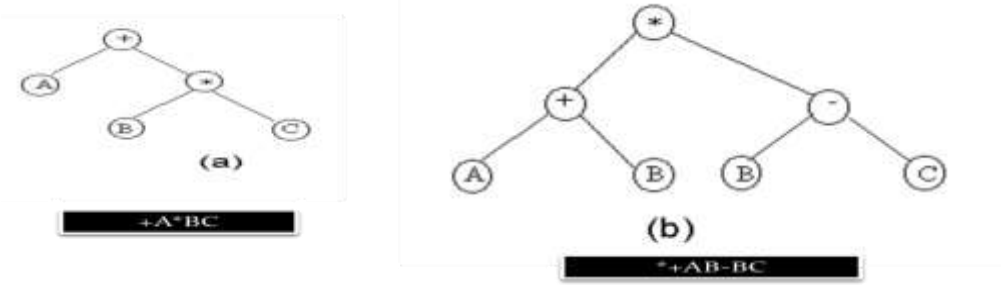
Dengan kata lain jika suatu elemen dikeluarkan dari antrean, maka cabang kiri dan kanan dari elemen yang baru saja dikeluarkan dimasukkan kedalam antrean.

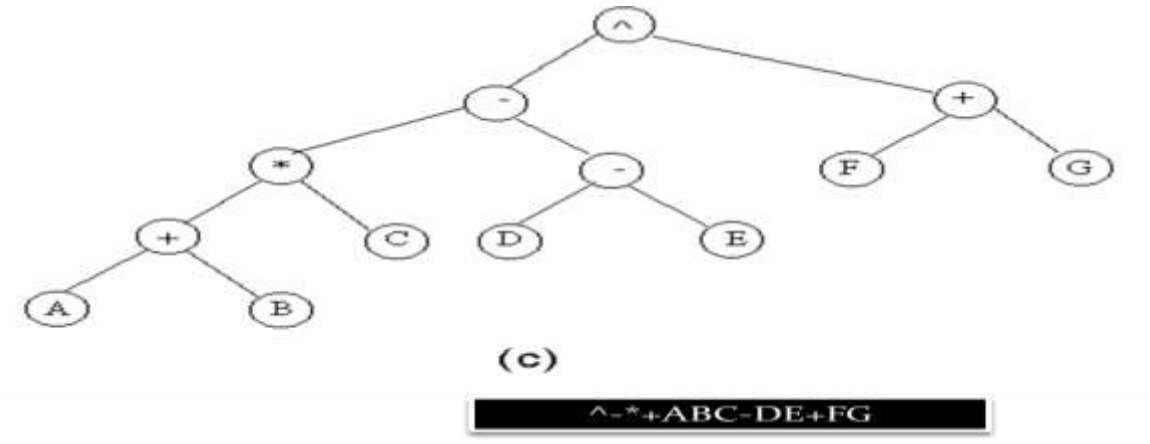
2.3.3 APLIKASI POHON BINER

1). NOTASI PREFIX, INFIX DAN POSTFIX

Pada bagian ini akan dibahas tentang bagaimana menyusun sebuah Pohon Binar yang apabila dikunjungi secara PreOrder akan menghasilkan Notasi Prefix, kunjungan secara InOrder menghasilkan Notasi Infix, dan kunjungan PostOrder menghasilkan Notasi Postfix.

Contoh :





Keterangan diatas sebagai berikut :

Berdasarkan Gambar diatas, apabila dilakukan kunjungan secara PreOrder, maka akan diperoleh Notasi Prefix dari persamaan-persamaan yang digambarkan tersebut, yaitu :

+A*BC	(Gambar.a)
*+AB-BC	(Gambar.b)
^-*+ABC-DE+FG	(Gambar.c)

Jika dilakukan kunjungan secara InOrder, akan diperoleh Notasi Infixnya, yaitu :

(A+(B*C))	(Gambar.a)
((A+B) * (B-C))	(Gambar.b)
((((A+B) * C) – (D-E))^(F+G))	(Gambar.c)

Jika dilakukan kunjungan secara PostOrder, akan diperoleh Notasi Postfixnya, yaitu :

ABC*+	(Gambar.a)
AB+BC-*	(Gambar.b)
AB+C*DE--FG+^	(Gambar.c)

Contoh Program Source Kunjungan Pohon biner

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

struct node
{
        struct node *kiri;
   char data;
   struct node *kanan;
};

node *root=NULL;
TambahNode(node **root, char isi)
{
        if((*root)==NULL)
   {
        node *baru;
      baru= new node;
      baru->data = isi;
                baru->kiri = NULL;
      baru->kanan = NULL;
      (*root)=baru;
   }
}

void preOrder (node *root)
{
        if(root !=NULL)
   {
        cout<<" "<<root->data;
      preOrder(root->kiri);
      preOrder(root->kanan);
   }
}
void inOrder (node *root)
{
        if(root !=NULL)
   {
        inOrder(root ->kiri);
        cout<<" "<<root->data;
      inOrder(root->kanan);
   }
}
void postOrder (node *root)
{
        if(root !=NULL)
   {
        preOrder(root->kiri);
```

```
    preOrder(root->kanan);
    cout<<" "<<root->data;
  }
}

//Program Utama
int main()
{
  char data, lagi;
  int pilihan;

  TambahNode(&root, data='A');
  TambahNode(&root->kiri, data='B');
  TambahNode(&root->kiri->kiri, data='D');
  TambahNode(&root->kiri->kanan, data='E');
  TambahNode(&root->kiri->kiri->kanan, data='H');
  TambahNode(&root->kanan, data='C');
  TambahNode(&root->kanan->kiri, data='F');
  TambahNode(&root->kanan->kanan, data='G');
  TambahNode(&root->kanan->kiri->kanan, data='I');

  awal:
  clrscr();
  cout<<endl<<endl;
  cout<<"                    POHON BINER"<<endl<<endl;
  cout<<"                 __________A _________"<<endl;
  cout<<"                 |                   |"<<endl;
  cout<<"                 |                   |"<<endl;
  cout<<"              _____B _____         _____C ____"<<endl;
  cout<<"              |         |         |         |"<<endl;
  cout<<"              |         |         |         |"<<endl;
  cout<<"              D ____    E         F ____    G"<<endl;
  cout<<"                 |                   |"<<endl;
  cout<<"                 |                   |"<<endl;
  cout<<"                 H                   I"<<endl;
  cout<<"\n\n                    PILIHAN\n"<<endl;
  cout<<" ------------------------------ "<<endl;
  cout<<"                    1. Pre-Order\n"<<endl;
  cout<<"                    2. In-order\n"<<endl;
  cout<<"                    3. Post-Order\n"<<endl;
  cout<<"                    4. Semua\n"<<endl;
  cout<<"                    5. Exit\n"<<endl<<endl<<endl<<endl;
  printf("                   Input : ");scanf("%d",&pilihan);
  cout<<endl<<endl;

  switch(pilihan)
    {
      case 1 :
```

```
                cout<<"          Pre-Order  : ";
                        preOrder(root);
                                cout<<endl;
                break;
        case 2:
                        cout<<"          In-Order   : ";
                        inOrder(root);
                        cout<<endl;
                break;
        case 3:
                        cout<<"          Post-Order : ";
                        postOrder(root);
                cout<<endl;
                break;
        case 4:
                cout<<"          Pre-Order  : ";
                        preOrder(root);cout<<endl<<endl;
                cout<<"          In-Order   : ";
                        inOrder(root);cout<<endl<<endl;
                        cout<<"          Post-Order : ";
                        postOrder(root);cout<<endl<<endl;
                break;
        case 5:
                return 0;
                break;
        }

  printf("\n\n                Ulangi ??? [Y/T] : ");
        lagi = getche() ;
        if (lagi == 'Y' || lagi == 'y')
        goto awal;
  getch();
  return 0;
}
```

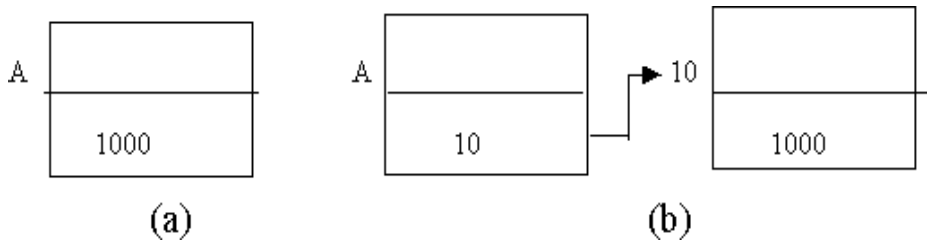
2.4 Single Linked List

Untuk mengolah data yang banyaknya tidak bisa ditentukan sebelumnya, maka disediakan satu fasilitas yang memungkinan untuk menggunakan suatu perubah yang disebut dengan perubah dinamis (Dinamic variable).

Perubah Dinamis (Dinamic variable) suatu perubah yang akan dialokasikan hanya pada saat diperlukan, yaitu setelah program dieksekusi.

2.4.1 Perbedaan Perubah Statis & Dinamis

Pada perubah statis, isi Memory pada lokasi tertentu (nilai perubah) adalah data sesungguhnya yang akan diolah. Pada perubah dinamis, nilai perubah adalah alamat lokasi lain yang menyimpan data sesungguhnya. Dengan demikian data yang sesungguhnya dapat dimasukkan secara langsung.Dalam hal cara pemasukkan data dapat diilustrasikan seperti dibawah ini.



2.4.2. Deklarasi pointer

Pointer digunakan sebagai penunjuk ke suatu alamat memori Dalam pemrograman C++, type Data Pointer dideklarasikan dengan bentuk umum :

Type Data * Nama Variabel;

Type Data dapat berupa sembarang type data, misalnya char, int atau float. Sedangkan Nama veriabel merupakan nama variabel pointer.

a. Type data pointer

Mensubstitusikan address sebuah variabel
ke pointer dengan memakai address
operator &

```
int x;
int *ptr;
ptr = &x
```

Mensubstitusikan address awal
sebuah array ke pointer

```
char  t[5];
char *ptr;
ptr = t;
```

Mensubstitusikan address salah satu
elemen array dengan address operator

```
char t[5] ;
char *ptr;
ptr = &t[3];
```

1. Contoh penggunaan pointer dalam program C++:

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

void main()
{
  int x,y,*z;
  x = 75;      //nilai x = 75
```

```
  y = x;       //nilai y diambil dari nilai x
  z = &x;     //nilai z menunjuk kealamat pointer dari nilai x
  getch();
}
```

2. POINTER PENAMBAHAN & PENGURANGAN

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
void main()
{
 int nilai [3],*petunjuk;
 clrscr();
 nilai [0]=124;
 nilai [1]=134;
 nilai [2]=750;
 petunjuk=&nilai[0];
 printf("nilai %i ada di alamat memory %p\n",*petunjuk,petunjuk);
 printf("nilai %i ada di alamat memory %p\n",*(petunjuk+1),petunjuk+1);
 printf("nilai %i ada di alamat memory %p\n",*(petunjuk+2),petunjuk+2);
 getch();
 }
```

3. Contoh Program Poiter

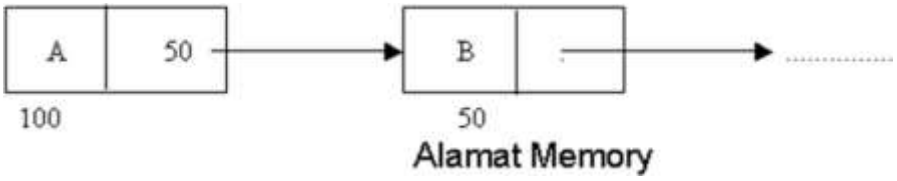
```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
void main()
{
 int a=100, b=200,*pa,*pb ;
 clrscr();
 pa=&a;
```

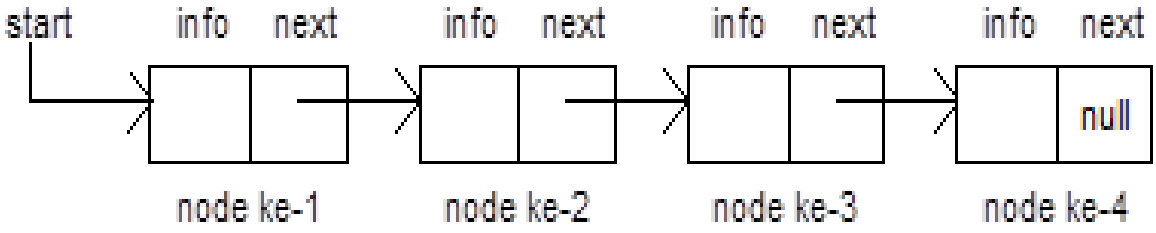
```
pb=&b;
if (pa < pb)
  printf("pa menunjuk ke memory lebih rendah dari pb\n");
if (pa==pb)
 printf("pa menunjuk  ke memory sama dengan pb\n");
if (pb < pa)
  printf("pa menunjuk ke memory lebih besar dari pb\n");
getch();
}
```

2.4.3 **LINKED LIST (LINKED LIST)**

Salah satu Struktur Data Dinamis yang paling sederhana adalah Linked List atau Struktur Berkait atau Senarai Berantai, *yaitu suatu kumpulan komponen yang disusun secara berurutan dengan bantuan Pointer*.

Linked List (Senarai Berantai) disebut juga dengan Senarai Satu Arah (One-Way List). Masing-masing komponen dinamakan dengan Simpul (Node).





1. **Perbedaan Karakteristik Array dan Linked List**

ARRAY	LINKED LIST
Statis	Dinamis
Penambahan / penghapusan data terbatas	Penambahan / penghapusan data tidak terbatas
Random access	Sequential access
Penghapusan array tidak mungkin	Penghapusan linked list mudah

Setiap simpul dalam suatu Linked List terbagi menjadi dua bagian,yaitu :

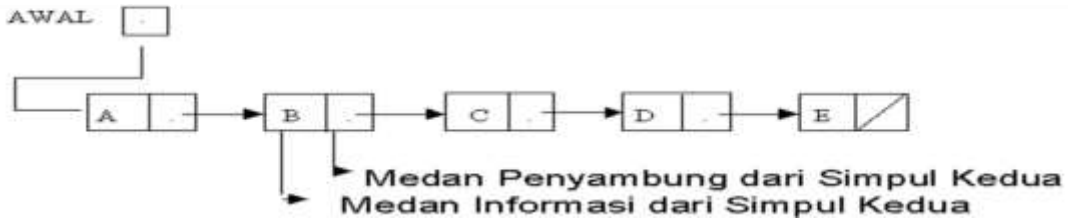
1. Medan Informasi

   Berisi informasi yang akan disimpan dan diolah.

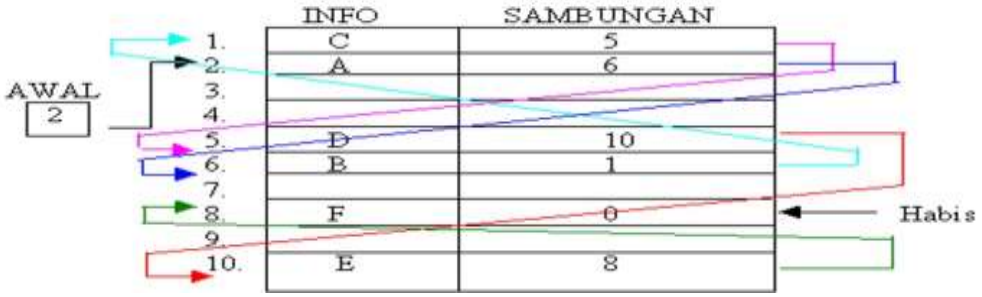
2. Medan Penyambung (Link Field)

   Berisi alamat berikutnya. Bernilai 0, Jika Link tersebut tidak menunjuk ke Data (Simpul) lainnya. Penunjuk ini disebut Penunjuk Nol.

Linked List juga mengandung sebuah variabel Penunjuk List, yang biasanya diberi nama START(AWAL), yang berisi alamat dari simpul pertama dalam List



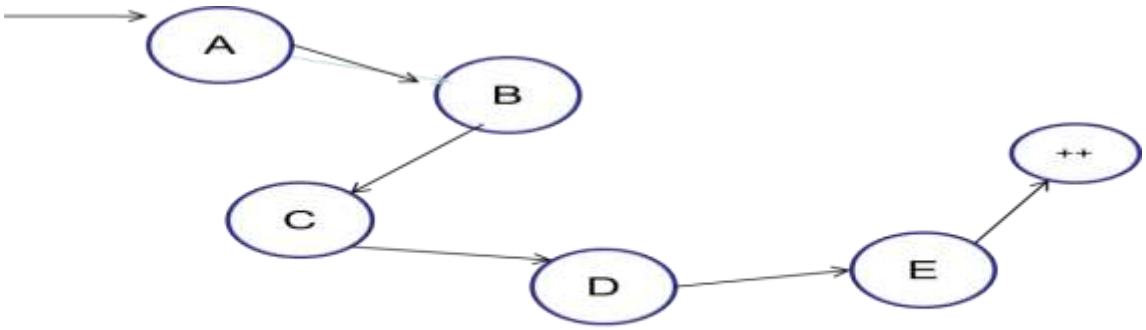
PENYAJIAN LINKED LIST DALAM MEMORY



Keterangan :

AWAL	= 2	,	Maka	INFO[2]	=	'A'
SAMBUNGAN[2]	= 6	,	Maka	INFO[6]	=	'B'
SAMBUNGAN[6]	= 1	,	Maka	INFO[1]	=	'C'
SAMBUNGAN[1]	= 5	,	Maka	INFO[5]	=	'D'
SAMBUNGAN[5]	= 10	,	Maka	INFO[10]	=	'E'
SAMBUNGAN[10]	= 8	,	Maka	INFO[8]	=	'F'
SAMBUNGAN[8]	= 0	,	Maka	Akhir Linked List		

Dari contoh diatas diperoleh untai **'ABCDEF'**



Contoh Linked List

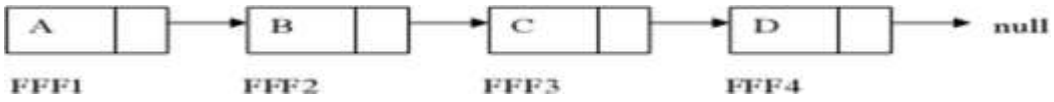
2. Bentuk Node Single Linked List no Circular

- Single : field pointer-nya hanya satu dan satu arah,pada akhir node pointernya menunjuk NULL

Menempati alamat memori tertentu

- Linked List : node-node tersebut saling terhubung satu
- Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data.
- Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

Contoh



3. Pembuatan Single Linked List non Circular

Deklarasi Node :

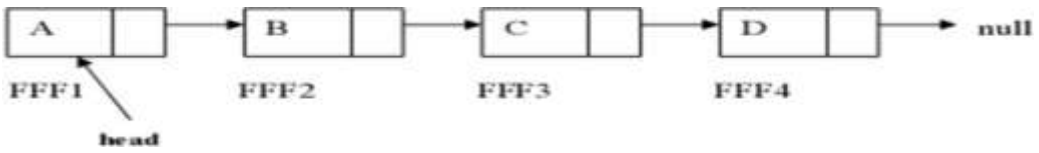
```
typedef struct TNode{
        int data;
        TNode *next;
};
```

Keterangan:

- Pembuatan struct bernama TNode yang berisi 2 field, yaitu field data bertipe integer dan field next yang bertipe pointer dari TNode
- Setelah pembuatan struct, buat variabel head yang bertipe pointer dari TNode yang berguna sebagai kepala linked list
- Digunakan perintah new untuk mempersiapkan sebuah node baru berserta alokasi memorinya, kemudian node tersebut diisi data dan pointer nextnya ditunjuk ke NULL.

```
TNode *baru;
baru = new TNode;
baru->data = databaru;
baru->next = NULL;
```

- Dibutuhkan satu buah variabel pointer : head yang akan selalu menunjuk pada node pertama



4. Fungsi Inisialisasi Single Linked List

```
void init()
{
head = NULL;
}
```

5. Function untuk mengetahui kondisi Single Linked List

- Jika pointer head tidak menunjuk pada suatu node maka kosong

```
int isEmpty()
{
   if (head == NULL) return 1;
   else return 0;
 }
```

6. Deklarasi Pointer Penunjuk Head Single Linked List

Manipulasi linked list tidak dapat dilakukan langsung ke node yang dituju, melainkan harus menggunakan suatu pointer penunjuk ke node pertama (Head) dalam linked list

Deklarasinya sebagai berikut:

```
TNode *head;
```

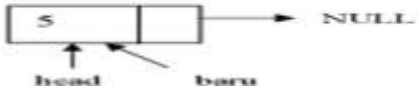
a. Menambah Node di Depan

- Penambahan node baru akan dikaitan di node paling depan, namun pada saat pertama kali (data masih kosong), maka penambahan data dilakukan dengan cara: node head ditunjukkan ke node baru tersebut.
- Prinsipnya adalah mengkaitkan node baru dengan head, kemudian head akan menunjuk pada data baru tersebut sehingga head akan tetap selalu menjadi data terdepan.

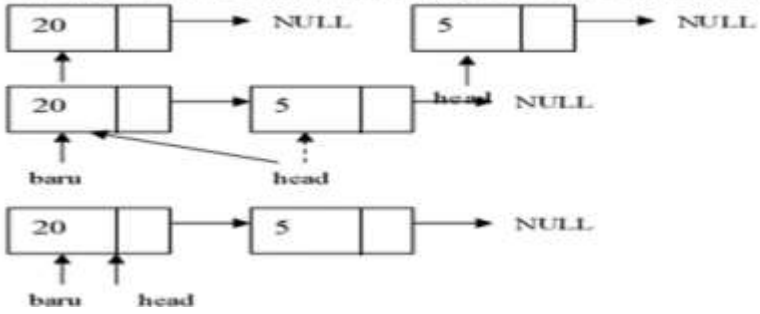
```
void insertDepan(int databaru)
{
 TNode *baru;
 baru = new TNode;
 baru->data = databaru;
 baru->next = NULL;
 if(isEmpty()==1)
  {
      head=baru;
      head->next = NULL;
  }
 else
  {
      baru->next = head;
      head = baru;
  }
 printf("Data masuk\n");
}
```

1. List masih kosong (head=NULL)

2. Masuk data baru, misalnya 5



3. Datang data baru, misalnya 20 (penambahan di depan)



b. Menambah Node di Belakang

- Penambahan data dilakukan di belakang, namun pada saat pertama kali, node langsung

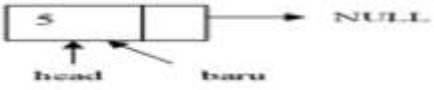
ditunjuk oleh head.

- Penambahan di belakang membutuhkan pointer bantu untuk mengetahui node terbelakang. Kemudian, dikaitkan dengan node baru.
- Untuk mengetahui data terbelakang perlu digunakan perulangan.

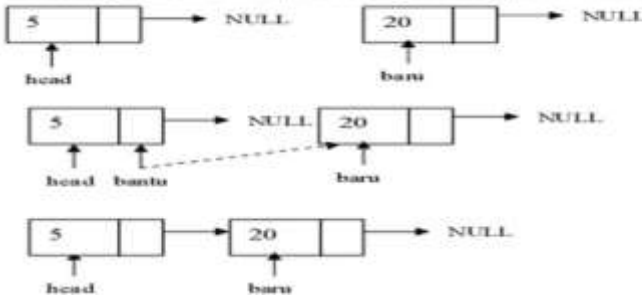
```
void insertBelakang (int databaru)
{
  TNode *baru,*bantu;
  baru = new TNode;
  baru->data = databaru;
  baru->next = NULL;
  if(isEmpty()==1) {
    head=baru;
    head->next = NULL;
  }
  else {
    bantu=head;
    while(bantu->next!=NULL){
      bantu=bantu->next;
    }
    bantu->next = baru;
  }
  printf("Data masuk\n");
}
```

1. List masih kosong (head=NULL)

2. Masuk data baru, misalnya 5



3. Datang data baru, misalnya 20 (penambahan di belakang)



BAB III

PENUTUP

3.1 Kesimpulan

Dari penjelasan diatas dapat disimpulkan Struktur data merupakan suatau mata kuliah yang harus benar – benar di pahami guna menciptakan mahasiswa mampu medesigned secara data tersruktur, serta Memahami sistem pengorganisasian data pada memori komputer dan file (berkas) pada media penyimpanan dan Mengimplementasikannya dalam program dengan menggunakan salah satu bahasa pemrograman generasi ke-3 (Bahasa C) untuk membuat berbagai macam struktur data (array, tree, struct) dengan teknik-teknik tertentu (linked list, stack, dan queue) serta manipulasinya (sorting dan searching) secara baik, efisien, dan cepat

3.2 Saran

Dari uraian di atas penulis berharap, pembaca dapat memahami tentang pengertian tentang macam struktur data (array, tree, struct) dengan teknik-teknik tertentu (linked list,

stack, dan queue) serta manipulasinya (sorting dan searching) Penulis menyadari bahwa dalam pembuatan makalah ini masih banyak kesalahan dan kekurangan, baik dari segi penulisan maupun dari segi pemaparan. Maka dari itu pemakalah menerima dan mengharapkan kritik dan saran dari pembaca yang sifatnya membangun, agar makalah ini dapat menjadi lebih sempurna dan menjadi rujukan bagi pembaca nantinya.