MODUL I

ARRAY 1 DIMENSI

1. TUJUAN

- Memahami array 1 dimensi
- Dapat mendeklarasikan array 1 dimensi
- Dapat menerapkan konsep array 1 dimensi

2. TEORI DASAR

Array adalah sekelompok data sejenis yang disimpan ke dalam variabel dengan nama yang sama, dengan memberi indeks pada variabel untuk membedakan antara yang satu dengan yang lain.
Array 1 Dimensi , array adalah hal yang paling penting dalam setiap bahasa pemrograman. Menurut definisi, array adalah alokasi memori statis. Ini mengalokasikan memori untuk tipe data yang sama secara berurutan.
Ini berisi beberapa nilai jenis yang sama. Hal ini juga menyimpan nilai-nilai dalam memori pada ukuran tetap. Array juga mempunyai definisi lain yaitu struktur data yang statis yang mempunyai 1 nama tetapi memiliki banyak tempat. Setiap tempat harus dibedakan, untuk membedakannya dibutuhkan penunjuk, pentunjuk dapat berupa karakter(char) atau integer.
Sekali disimpan dalam penunjuk yang sama(berbentuk indeks), maka isinya tidak akan hilang kecuali indeksnya diisi oleh nilai yang lain.
Cara pendeklarasian Array setiap bahasa Program berbeda tapi semuanya memiliki karakter yang sama .
di C++
tipedata namaVariabel [jumlahElemen];

Contoh implementasi array

Program1

```
#include <iostream.h>
#include <conio.h>
void main ()
{ int y [] = {1, 2, 7, 4, 5};
  int n, r=0;
  for ( n=0 ; n<5 ; n++ )
  {r += y[n];}
  cout<<" "<<r;
  getch();
}
```

Program2

```
 #include <iostream.h>
#include <conio.h>
void main ()
{ int nilai[5],x;
  cout<<"Masukkan nilai :\n\n";
  for(x=0;x<5;x++)
```

```
  { cout<<"Nilai Angka  : ";cin>>nilai[x];}
  cout<<"\n\nOutput nilai   : \n";
  for(x=0;x<5;x++)
  { cout<<"\nNilai Angka  : "<<nilai[x];}
getch();
 }
```

Program3

```
#include <iostream.h>
#include <conio.h>
void main ()
{ int A [5]={20,9,1986,200,13};
  Int n;
  clrscr();
  cout<<"Data yang lama\n";
  for (n=0;n<5;n++)
  { cout<<" "<<A[n];}
  cout<<"\nData yang baru : \n";
  A[0]=4;
  A[1]=2;
  A[2]=1;
  A[3]=3;
  A[4]=5;
  for (n=0;n<5;n++)
  { cout<<" "<<A[n]; }
  getch();
}
```

Program4

```
#include <iostream.h>
#include <conio.h>
void main ()
{ int A [5]={20,9,1986,200,13};
  int n,hapus;
  clrscr();
  cout<<"Data yang lama\n";
  for (n=0;n<5;n++)
    { cout<<" "<<A[n];}
  cout<<" data yang ingin dihapus : ";
  cin>>hapus;
  cout<<"\nData yang baru : \n";
  for (n=hapus-1;n<5-1;n++)
    { A[n]=A[n+1];}
  for (n=0;n<4;n++)
    { cout<<" "<<A[n]; }
  getch()}
```

3. LATIHAN

1. Buatlah program untuk menginputkan elemen array secara dinamis dgn batas elemen array 10
2. Tambahkan perintah untuk cari, edit dan hapus elemen array

MODUL II
ARRAY 2 DIMENSI

1. TUJUAN

- Memahami array 2 dimensi
- Dapat mendeklarasikan array 2 dimensi
- Dapat menerapkan konsep array 2 dimensi

2. TEORI DASAR

Pengertian Array 2 Dimensi

Pengertian sederhananya, "*Array 2 Dimensi adalah Sebuah array yang menampung array lain sebagai data nilai dari setiap indeks array penampung*". Jika kita sudah mengerti mengenai array 1 dimensi, kita bisa langsung membayangkan dan mungkin bisa langsung mengerti dari **definisi**array 2 dimensi diatas. Oke, untuk lebih memudahkan langkah kita, mari mencoba membahas sedikit mengenai array 1 dimensi dan dasar-dasarnya di pembahasan deklarasi dan inisialisasi array sebagai langkah untuk lebih memahami array 2 dimensi ini.

Deklarasi dan Inisialisasi Array 2 Dimensi

Sewaktu mempelajari array 1 dimensi, tentunya kita sudah tahu bahwa untuk setiap nilai di dalam array haruslah bertipe data yang sama. Pada bahasa C / C++, tidak dibenarkan untuk melakukan pencampuran tipe data untuk setiap nilai yang ditampungnya. Pencampuran tipe data untuk setiap nilai array seperti ini dapat ditemukan pada bahasa pemrograman JavaScript, PHP, Phyton, dan bahasa pemrograman lain yang mendukung fitur ini. Nah, untuk pendeklarasian array 1 dimensi tentu saja bukan hal yang sulit lagi jika kita sudah memiliki sedikit pemahaman mengenai array 1 dimensi.
Berikut ini merupakan cara pendeklarasian dan inisialisasi array 1 dimensi.

```
1.  int arrayBilangan[] = {1, 2, 3}; //array memiliki 3 buah data / nilai
2.  char arrayKarakter[] = {'a','b','c'}; //array memiliki 3 buah data / nilai
```

Berdasarkan pendeklarasian dan inisialisasi nilai array 1 dimensi di atas, kita dapat menyimpulkan bahwa setiap kali mendeklarasikan array dan array langsung diinisialisasi, kita tidak perlu memberikan lebar dari data array dan proses inisialisasi nilai array ditandai dengan tanda kurung kurawal "{ }", nilai / data dari setiap array diletakkan di dalam kurung kurawal tersebut dan untuk setiap nilai / data dipisahkan dengan tanda koma ','.

```
1.  int arrayBilangan[][2] = {{1, 2}, {3, 4}, {5, 6}};
2.  char arrayKar[][2] = {{'A', 'a'}, {'B','b'}, {'C','c'}};
```

Berdasarkan pendeklarasian dan inisialisasi array 2 dimensi bahasa C di atas, untuk melihat perbedaannya dengan pendeklarasian dan inisialisasi array 1 dimensi sebelum array 2 dimensi ini lihat pernyataan berikut ini.

1. // int arrayBilangan[][2]
variabel ini memiliki 3 buah data, dan setiap indeks array tersebut menampung sebuah array yang memiliki 2 buah elemen anggota maksimal
2. // char arrayKar[][2]
variabel ini sama seperti variabel pertama, yang membedakan hanya tipe datanya saja

Untuk memahami lebih lanjut lagi, kita akan melihat bagaimana array 2 dimensi diimplementasikan pada sebuah matriks.

3. LATIHAN :

Buatlah program matrik dengan operasi-operasi sbb:

1. Input dan output matrik
2. Penjumlahan, pengurangan dan perkalian matrik
3. Transpose dan diagonal matrik
4. Menampilkan nilai maksimal dan minimal matrik

MODUL III

SORTING

1. TUJUAN

- Dapat memahami teori tentang sorting
- Dapat menerapkan konsep sorting baik dengan metode bubble sort, selection sort dan insertion sort

2. TEORI DASAR

Bab ini akan membahas tentang sorting dan masih berkaitan dengan array.

Ada 3 jenis metode sorting yang akan dibahas yaitu :

1. Bubble sort (gelembung) secara ascending

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

25	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:

K=N=6					2	7
					1	6
K=5				8	2	7
					1	6
K=4			8	1	2	7
				0	1	6
K=3		8	2	1	2	7
			7	0	1	6
K=2		2	2	1	2	7
		5	7	0	1	6

Hasil akhir langkah 1:

8	25	27	10	21	76
1	2	3	4	5	6

Langkah 2:

K=N=6					2	7
					1	6
K=5				1	2	7
				0	1	6
K=4			1	2	2	7
			0	7	1	6
K=3		1	2	2	2	7
		0	5	7	1	6

Hasil akhir langkah 2 :

8	10	25	27	21	76
1	2	3	4	5	6

Langkah 3:

K=N=6 2 7
1 6
K=5 2 2 7
1 7 6
K=4 2 2 2 7
1 5 7 6

Hasil akhir langkah 3 :

8	10	21	25	27	76
1	2	3	4	5	6

Langkah 4:

K=N=6 2 7
7 6
K=5 2 2 7
5 7 6

Hasil akhir langkah 4:

8	10	21	25	27	76
1	2	3	4	5	6

Langkah 5:

K=N=6 27 76

Hasil akhir langkah 5:

8	10	21	25	27	76
1	2	3	4	5	6

Sorting sudah selesai, karena data sudah terurutkan.

Contoh pengurutan data menggunakan metode bubble sort

```
//Jumlah elemen dalam array ada 5
L[0]=1;
L[1]=50;
L[2]=10;
L[3]=3;
L[4]=2;
```

```
//Proses secara Ascending(naik)
for(i=0;i<=4;i++)
  for(k=0;k<=4;k++)
   if (L[k]>L[k+1])
    {temp=L[k];
     L[k]=L[k+1];
     L[k+1]=temp;  }
for(i=1;i<=5;i++)
cout<<L[i]<<endl;
```

2. Selection sort (maksimum/minimun)

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

29	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:

Cari elemen maksimum di dalam larik L[1..6]→ maks = L[5] = 76

Tukar maks dengan L[N],hasil akhir langkah 1:

29	27	10	8	21	**76**
1	2	3	4	5	6

Langkah 2:

(berdasarkan susunan larik hasil langkah 1)

Cari elemen maksimum di dalam larik L[1..5]→ maks = L[1] = 29

Tukar maks dengan L[5],hasil akhir langkah 2:

21	27	10	8	**29**	**76**
1	2	3	4	5	6

Langkah 3:

(berdasarkan susunan larik hasil langkah 2)

Cari elemen maksimum di dalam larik L[1..4]→ maks = L[2] = 27

Tukar maks dengan L[4],hasil akhir langkah 3:

21	8	10	**27**	**29**	**76**
1	2	3	4	5	6

Langkah 4:

(berdasarkan susunan larik hasil langkah 3)

Cari elemen maksimum di dalam larik L[1..3]→ maks = L[1] = 21

Tukar maks dengan L[3],hasil akhir langkah 4:

10	8	**21**	**27**	**29**	**76**

1	2	3	4	5	6

Langkah 5:

(berdasarkan susunan larik hasil langkah 4)

Cari elemen maksimum di dalam larik L[1..2]→ maks = L[1] = 10

Tukar maks dengan L[2],hasil akhir langkah 5:

8	**10**	**21**	**27**	**29**	**76**
1	2	3	4	5	6

Jika nilai pada array/larik sudah terurutkan maka proses sorting sudah selesai.

Contoh pengurutan data menggunakan metode selection sort (maksimum /minimum)

```
int main(){
//deklarasi array dengan 7 elemen
int A[7];
int j,k,i,temp;
int jmax,u=6;

//memasukkan nilai sebelum diurutkan
cout<<"Masukkan nilai pada elemen array :"<<endl;
for(i=0;i<7;i++)
{ cout<<"A["<<i<<"]=";
  cin>>A[i];}

//Proses pengurutan secara menaik (Ascending)
for(j=0;j<7;j++)
{ jmax=0;
  for(k=1;k<=u;k++)
  if (A[k] > A[jmax])
     jmax=k;
  temp=A[u];
  A[u]=A[jmax];
  A[jmax]=temp;
  u--;}

//menampilkan nilai setelah diurutkan
cout<<"\nNilai setelah diurutkan ="<<endl;
for(i=0;i<7;i++)
 cout<<A[i]<<"  ";
```

3. Insertion sort (sisip)

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

29	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:

Elemen L[1] dianggap sudah terurut

29	27	10	8	76	21
1	2	3	4	5	6

Langkah 2:

(berdasarkan susunan larik pada langkah 1)

Cari posisi yang tepat untuk L[2] pada L[1..2],diperoleh :

29	**27**	10	8	76	21
1	2	3	4	5	6

Langkah 3:

(berdasarkan susunan larik pada langkah 2)

Cari posisi yang tepat untuk L[3] pada L[1..3],diperoleh :

10	**27**	**29**	8	76	21
1	2	3	4	5	6

Langkah 4:

(berdasarkan susunan larik pada langkah 3)

Cari posisi yang tepat untuk L[4] pada L[1..4],diperoleh :

8	**10**	**27**	**29**	76	21
1	2	3	4	5	6

Langkah 5:

(berdasarkan susunan larik pada langkah 4)

Cari posisi yang tepat untuk L[5] pada L[1..5],diperoleh :

8	**10**	**27**	**29**	**76**	21
1	2	3	4	5	6

Langkah 6:

(berdasarkan susunan larik pada langkah 5)

Cari posisi yang tepat untuk L[6] pada L[1..6],diperoleh :

8	**10**	**21**	**27**	**29**	**76**
1	2	3	4	5	6

Jika nilai pada array/larik sudah terurutkan maka proses sorting sudah selesai.

Contoh pengurutan data menggunakan metode insertion sort (sisip)

```
L[1]=1;
L[2]=25;
```

```
L[3]=10;
L[4]=30;
L[5]=2;
for(k=2;k<=5;k++){
temp=L[k];/* ambil elemen L[k] supaya tidak tertimpa
             penggeseran*/

/* Cari Posisi Yang tepat dalam L[1..k-1] sambil menggeser*/
 j=k-1;
    while(temp<=L[j])
    {L[j+1]=L[j];
     j--;}
 if((temp >= L[j])|| ( j=1))
   L[j+1]=temp; /*posisi yg tepat untuk L[k] ditemukan*/
 else
   { L[j+1]=L[j];
     L[j]=temp; }
}
for(k=1;k<=5;k++)
 cout<< L[k]<<"   ";
```

3. LATIHAN

Ketik ulang semua contoh kode pengurutan lalu analisa hasilnya.

MODUL IV
SEARCHING

1. TUJUAN

- Memahami konsep tentang searching
- Dapat menerapkan konsep searching ke dalam bentuk program

2. DASAR TEORI

Pada bab ini akan menjelaskan materi tentang searching yaitu pencarian data dengan membandingkan setiap setiap elemen larik satu per satu secara urut (beruntun), mulai dari elemen pertama sampai dengan elemen yang terakhir. Ada 2 macam pencarian beruntun,yaitu pencarian pada array yang **sudah** terurut, dan pencarian pada array yang **belum** terurut.

Ada 3 metode pencarian yang akan kita bahas disini:

1. Pencarian Beruntun (Sekuensial Search)

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

13	16	14	21	76	21
1	2	3	4	5	6

X = 21

Maka, elemen yang di cek adalah 13, 16, 14, 21 (ditemukan), berarti data ditemukan pada indeks ke 3 elemen ke 4.

Contoh Pencarian data dengan metode beruntun

```
int larik[9]={1,12,3,4,10,6,7,11,9}, i,n, x, posisi;

cout<<"data yang ingin dicari ? ";cin>>x;
i=0;posisi=0;

while(i<8 && larik[i]!=x)
{i++;}
if (larik[i]!=x)
   cout<<"maaf data yang dicari tidak ada";
else if (larik[i]==x)
   { posisi=i+1;
     cout<<"pada posisi ke "<<posisi;
}
```

2. Pencarian Beruntun dengan sentinel

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

13	16	14	21	76	21
1	2	3	4	5	6

a. Misalkan elemen yang dicari adalah X=21

Maka, tambahkan 21 sebagai elemen sentinel di L[N+1]:

13	16	14	21	76	21	21
1	2	3	4	5	6	7

Elemen yang diperiksa selama pencarian : 13, 16, 14, 21

Elemen larik yang dikembalikan : 4, karena 4 != N+1, berarti X =21 terdapat di dalam larik semula.

b. Misalkan elemen yang dicari adalah X=13

Maka, tambahkan 13 sebagai elemen sentinel di L[N+1]:

13	16	14	21	76	21	13
1	2	3	4	5	6	7

Elemen yang diperiksa selama pencarian : 13

Elemen larik yang dikembalikan : 1, karena 1 != N+1, berarti X = 13 terdapat di dalam larik semula.

Contoh Pencarian data dengan metode sentinel

```
int larik[9]={1,12,3,4,10,6,7,11,9}, i,n, x, posisi;

cout<<"data yang ingin dicari ? ";cin>>x;
i=0;posisi=0;

while(i<8 && larik[i]!=x)
{i++;}
if (larik[i]!=x)
   cout<<"maaf data yang dicari tidak ada";
else if (larik[i]==x)
   { posisi=i+1;
     cout<<"pada posisi ke "<<posisi;
}
```

3. Pencarian Bagi dua (Binary Search)

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

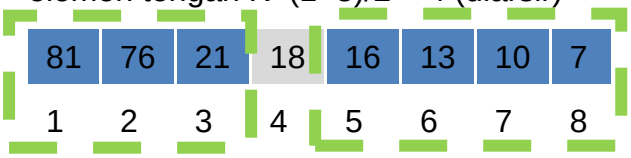
81	76	21	18	16	13	10	7
Ia=1	2	3	4	5	6	7	8=ib

a. Misal elemen yang dicari adalah X = 18

langkah 1:

ia=1 dan ib=8

elemen tengah K=(1+8)/2 = 4 (diarsir)



Langkah 2:

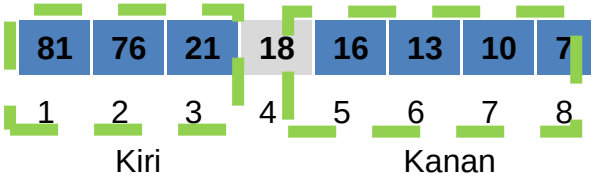
L[4]=18? Ya! (X ditemukan, pencarian dihentikan)

b. Misal elemen yang dicari adalah X = 16

langkah 1:

ia=1 dan ib=8

elemen tengah K=(1+8)/2 = 4 (diarsir)

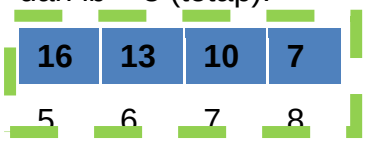


Langkah 2:

L[4]=18? Tidak !

harus diputuskan apakan pencarian akan dilakukan di bagian kiri atau kanan dengan pemeriksaan sbb:

L[4] > 16 ? Ya ! Lakukan pencarian pada larik bagian kanan dengan ia=k+1 = 5 dan ib = 8 (tetap).



Langkah 2.1:

ia = 5 dan ib = 8

elemen tengah K = (5+8)/2 = 6 (diarsir)

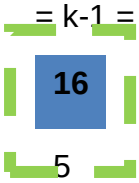
Kiri Kanan

Langkah 2.2:

L[6] = 16? Tidak !

harus diputuskan apakan pencarian akan dilakukan di bagian kiri atau kanan dengan pemeriksaan sbb:

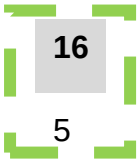
L[6] > 16 ? Tidak ! Lakukan pencarian pada larik bagian kiri dengan ia= 5 dan ib = k-1 = 5



Langkah 2.1.1:

ia = 5 dan ib = 5

elemen tengah K = (5+5)/2 = 5 (diarsir)



Langkah 2.1.2:

L[5] = 16? Ya ! (X ditemukan,pencarian dihentikan)

Contoh Pencarian data dengan metode bagi dua

```
int array[arraySize] = {1, 2, 3, 4, 5};
int first, mid, last,target;

cout << "Enter a target to be found: "; cin >> target;

first = 0;   last = 4;

while(first <= last)
{ mid = (first + last)/2;
  if(target > array[mid])
   { first = mid + 1; }
  else if(target < array[mid])
   { last = mid - 1;  }
  else
   { first = last+1; }
}
if(target == array[mid])
  { cout << "Target found in elemen"<<mid+1 << endl; }
else
  { cout << "Target not found." << endl;  }
```

3. LATIHAN

Ketik ulang semua contoh kode pengurutan lalu analisa hasilnya.

MODUL V
STRUCT

1. TUJUAN

- Memahami struct
- Dapat mendeklarasikan struct
- Dapat menerapkan konsep array dalam struct

2. TEORI DASAR

Structure (struct) merupakan kumpulan variable-variable yang bertempat di suatu tempat yang sama. Berbeda dengan array yang berisi kumpulan variable-variable yang bertipe sama dalam satu nama, maka suatu structure dapat terdiri dari variable-variable yang berbeda tipenya yang tersimpan dalam satu nama structure.
Dalam hal ini, dalam suatu structure bisa terdapat berbeda-beda type data seperti char, int, float, dll.
Pola Structure di C++ :

```
struct himpunan
{
    int n;
    char nama[20];
};
```

Dalam pola di atas , setiap akhir pendeklarasian variable di dalam struct, harus di akhiri dengan tanda } dan ; .
Dalam penggunaannya di dunia nyata, structure digunakan untuk program antrian dan tumpukan.

Contoh implementasi STRUCT

```
enum j_kel{pria,wanita} ;

int main() {
struct siswa
{ int nis;
  char nama[20];
  j_kel kelamin;}A;

  //input data
  A.nis=123;
  A.nama=="Yuli";
  A.kelamin=wanita;

  //output data
  cout<<"\nNis        : "<<A.nis<<endl;
  cout<<"Nama      : "<<A.nama;
```

```
  cout<<"\nKelamin   : "<<A.kelamin;
  getch();
}
```

Contoh kasus :

Buat Program untuk menghitung spp mahasiswa menggunakan struktur, diketahui :

a. D3
   - spp tetap Rp 500.000
   - spp var Rp 25.000/sks

b. S1
   - spp tetap Rp 750.000
   - spp var Rp 50.000/sks

Penyelesaian :

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

struct mhs
{ char nama[20],nim[10],jurusan[2];
  int sks,program; };
  struct mhs bayar;
main ()
{
  int bts,var,tetap;

    //input data
    cout<<"\nNama mhs        = ";cin>>bayar.nama;
    cout<<"NIM              = ";cin>>bayar.nim;
    cout<<"Jurusan[TI,MI,SI] = ";
    cin>>bayar.jurusan;
    input:
    cout<<"Program[1=D3,2=S1]= ";
    cin>>bayar.program;
    if (bayar.program < 0 || bayar.program > 2)
       {cout<<"Program tidak sesuai\n";
        goto input;}
    cout<<"Jumlah sks        = ";cin>>bayar.sks;

    if (bayar.program==1)
       {tetap=500000;
        var=bayar.sks*25000;}
    else if (bayar.program==2)
       {tetap=750000;
        var=bayar.sks*50000;}
    cout<<"";

  //output data
    cout<<"\n\n-----------------------\n";
```

```
  cout<<"       Output  ";
  cout<<"\n----------------------\n";
  cout<<"\nNama mhs     = "<<bayar.nama;
  cout<<"\nNIM          = "<<bayar.nim;
  cout<<"\nJurusan      = "<<bayar.jurusan;
  cout<<"\nProgram      = "<<bayar.program;
  cout<<"\nJumlah sks   = "<<bayar.sks;
  cout<<"\nSpp tetap    = "<<tetap;
  cout<<"\nSpp variabel = "<<var;
  cout<<endl;
getch();}
```

3. LATIHAN

1. Buat program untuk menghitung jumlah nilai akhir mahasiswa menggunakan structure dengan ketentuan:

   Nilai akhir = 10%*tugas + 20%*kuis + 30%*mid + 40%*uas

   Nilai Huruf:

   Nilai akhir >85 : A

   85 >= nilai akhir > 70 : B

   70 >= nilai akhir > 55 : C

   55 >= nilai akhir > 40 : D

   Nilai akhir <=40 : E

2. Buat sebuah program untuk menghitung gaji harian pegawai, bila diketahui ketentuannya sebagai berikut :

   Gaji per jam = 500

   Bila jumlah jam kerja hari itu > 7 jam, maka kelebihannya dihitung lembur yang besarnya 15 x gaji per jam.

   Input : jumlah jam kerja

   Output : gaji harian pegawai

   Tentukan sendiri variabel-variabel yang dibutuhkan pada program ini.

MODUL VI

STACK

1. TUJUAN

- Mamahami konsep stack
- Memahami operasi dasar stack
- Dapat mengaplikasikan stack dalam kasus nyata

2. TEORI DASAR

Stack dibuat menggunakan array dan sebuah variabel bertipe integer yang menunjukkan posisi puncak stack. Dalam stack terdapat beberapa fungsi untuk operasi dasar, yaitu push() dan pop(). Percobaan kali ini menerangkan stack dalam potongan – potongan kode program. Langkah pertama adalah membuat array untuk stack. Dalam kode ini menggunakan asumsi

bahwa stack digunakan untuk tipe data integer saja.

Contoh deklarasi stack

```
int stack[MAX];  // array stack
int top;     // penunjuk posisi atas stack
```

Kemudian membuat fungsi push() untuk operasi memasukkan elemen baru ke dalam stack.

Dalam fungsi ini terdapat pemeriksaan, bila nilai top sama dengan atau lebih dari MAX

berarti stack sudah penuh.

Contoh fungsi push()

```
void push(int e) {
  if(top >= MAX) {
    printf("Stack penuh\n");
  } else {
    stack[top] = e;
    top++;
  }}
```

Fungsi yang kedua adalah pop() yang digunakan untuk operasi mengeluarkan isi stack. Fungsi akan mengembalikan nilai stack paling atas. Dalam fungsi ini juga terdapat pemeriksaan bila nilai top sama dengan atau kurang dari 0 berarti stack kosong.

Contoh fungsi pop()

```
void push(int e) {
  if(top >= MAX) {
    printf("Stack penuh\n");
  } else {
    stack[top] = e;
```

```
    top++;
  }}
```

Selain dua fungsi di atas bisa pula ditambahkan dengan fungsi show() untuk menampilkan isi

stack dan fungsi menu() untuk menampilkan daftar menu pilihan.

Contoh program stack

```
#include <conio.h>
#include <string.h>
#include <iostream.h>
#define MAX_STACK 10

typedef struct stack
{
     int top;
     char data[10][10];
} tumpuk1;

stack tumpuk;

void inisialisasi(){
tumpuk.top = -1;
}

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

void Push(char d[10]){
tumpuk.top++;
strcpy(tumpuk.data[tumpuk.top],d);}

void Pop(){
cout<<"\nData yang terambil = "<<tumpuk.data[tumpuk.top]<<endl;
tumpuk.top--;}

void Clear(){
tumpuk.top=-1;}

void TampilStack(){
for(int i=tumpuk.top;i>=0;i--)
   {cout<<"Data : "<<tumpuk.data[i]<<endl;}}
```

```
int main(){
 int pil;
 inisialisasi();
 char dt[10];
 do{
     cout<<"1. push\n";
     cout<<"2. pop\n";
     cout<<"3. print\n";
     cout<<"4. clear\n";
     cout<<"5. exit\n";
     cout<<"\nPilihan : ";cin>>pil;
switch(pil){
     case 1: if(IsFull() != 1){
               cout<<"\nData = ";cin>>dt;
               cout<<endl;}
             else
               cout<<"\nSudah penuh!\n";
             break;
     case 2: if(IsEmpty() != 1)
             else
               cout<<"\nMasih kosong!\n";
               break;
     case 3: if(IsEmpty() != 1)
             else
               cout<<"\nMasih kosong!\n";
               break;
     case 4: Clear();
             cout<<"\nSudah kosong!\n";
             break;}
             getch(); }
     while(pil != 5);
     getch();}
```

3. LATIHAN

Dengan menggunakan fungsi push() dan pop() pada stack, buatlah program untuk membalik kalimat.

MODUL VII
QUEUE

1. TUJUAN

- Mamahami konsep queue
- Memahami operasi dasar queue
- Dapat mengilustrasikan cara kerja queue menggunakan model circular array

2. TEORI DASAR

Queue (antrian) dibuat menggunakan array dan dua buah variabel bertipe integer yang menunjukkan posisi awal dan akhir antrian. Dalam queue terdapat beberapa fungsi untuk operasi dasar, yaitu store() dan retrieve(). Percobaan kali ini menerangkan queue dalam potongan – potongan kode program.

Langkah pertama adalah membuat array untuk queue. Dalam kode ini menggunakan asumsi bahwa queue digunakan untuk tipe data integer saja.

Contoh deklarasi queue

```
int queue[MAX];     // antrian
int spos = 0;     // index posisi simpan (store)
int rpos = 0;     // index posisi ambil (retrieve)
```

Index posisi ambil biasanya terletak di awal, dan index posisi simpan terletak di akhir. Kemudian membuat fungsi store() untuk operasi memasukkan elemen baru ke dalam queue. Dalam fungsi ini terdapat pemeriksaan, bila nilai spos sama dengan atau lebih dari MAX berarti queue sudah penuh.

Contoh fungsi store

```
void store(int e) {
  if(spos == MAX) {
    printf("antrian penuh\n");
    return;  }
queue[spos++] = e; }
```

Fungsi yang kedua adalah retrieve() yang digunakan untuk operasi mengambil isi queue. Fungsi akan mengembalikan nilai elemen antrian pada posisi **rpos**. Dalam fungsi ini juga terdapat pemeriksaan bila nilai spos sama dengan rpos berarti queue kosong.

Contoh fungsi retrieve

```
int qretrieve(void) {
  if(rpos == spos) {
    printf("antrian kosong\n");
    return 0; }
return queue[rpos++]; }
```

Selain dua fungsi di atas bisa pula ditambahkan dengan fungsi show() untuk menampilkan isi queue dan fungsi menu() untuk menampilkan daftar menu pilihan.

Contoh program queue

```
#include <conio.h>
#include <iostream.h>
#include <string.h>
#define max 20

typedef struct queue //  Mendefinisikan queue dengan menggunakan
struct
{  int head;
     int tail;
     char data [15][20];        // menampung 15 data dengan
jumlah string max 20 huruf
}antrian;
queue antri;
void inisialisasi()
{ antri.head = antri.tail = -1;}

int isFull()
{    if (antri.tail==max-1)
          return 1;
     else
          return 0; }
int isEmpty(){
     if (antri.tail==-1)
     {  antri.head=-1;
        return 1;}
     else
        return 0; }

void enqueue(char d[20])
{    antri.head=0;
     antri.tail++;
     strcpy(antri.data[antri.tail],d);
     cout<<"\tdata berhasil dimasukkan,tekan sembarang tombol
untuk lanjut !\n\n";}

void dequeue()
{  cout<<"data terambil"<<"   "<<antri.data[antri.head]<<endl;
```

```
     for (int i=antri.head;i<=antri.tail;i++)
          strcpy (antri.data[i],antri.data[i+1]);
          antri.tail--; }
void clear()
{  antri.head=antri.tail=-1;
     cout<<"semua data terhapus.\n";}
void print()
{    for (int i=0;i<=antri.tail;i++)
     cout<<"\ntampil data "<<antri.data[i]<<endl<<endl;}
int main()
{    int pil;
     inisialisasi();
     char dt[20];
     do{
          cout<<"1. input\n";
          cout<<"2. delete\n";
          cout<<"3. print\n";
          cout<<"4. clear\n";
          cout<<"5. exit\n";
          cout<<"Pilihan : ";cin>>pil;
          switch(pil){
          case 1: if(isFull() != 1){
                    cout<<"Data = ";cin>>dt;
                    enqueue(dt);}
                  else
                    cout<<"\nSudah penuh!\n";
                  break;
          case 2: if(isEmpty() != 1)
                    dequeue();
                  else
                    cout<<"\nMasih kosong!\n";
                  break;
          case 3: if(isEmpty() != 1)
                     print();
                  else
                     cout<<"\nMasih kosong!\n";
                  break;
          case 4: clear();
                  cout<<"\nSudah kosong!\n";
                  break; }
          getch();}

          while(pil != 5);
          getch();}
```

3. LATIHAN

Ketik ulang semua contoh kode pengurutan lalu analisa hasilnya.

MODUL VIII
LINK LIST

1. TUJUAN

- Mengenal single linked list dan operasinya
- Mengenal variasi operasi linked list
- Dapat membuat simulasi sederhana operasi linked list

2. TEORI DASAR

Bab ini akan membahas tentang linked list beserta operasinya. Selain itu juga akan membahas variasi operasi linked list yang meliputi penambahan dan pembacaan. Langkah pertama adalah membuat deklarasi node menggunakan struct. Di dalam node ada sebuah pointer yang menunjuk ke node tersebut.

Contoh deklarasi node

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

typedef struct Node {
  int elm;
  struct Node *next;
} node;

node *start;  // kepala linked list
int count;    // jumlah node
```

Kemudian membuat fungsi init(), yaitu membuat kepala linked list. Di sini terjadi pengalokasian memory secara dinamis untuk start. Kepala linked list ini tidak masuk dalam hitungan rangkaian.

Contoh fungsi init

```
void init(void) {
  start = malloc(sizeof(node));   // alokasi memori dinamis
if(!start) {        // bila alokasi gagal
    printf("alokasi gagal..\n");
    return;  }
start->next = NULL;
count = 0; }
```

Alokasi memori dinamis dapat dilakukan dengan fungsi malloc(). Dalam potongan kode di atas, alokasi memori untuk start adalah sebesar ukuran node.

Berikutnya adalah membuat fungsi insert() yang digunakan untuk menambahkan node baru dalam rangkaian linked list. Node yang ditambahkan akan selalu diletakkan di depan node dengan isi lebih besar (urut naik).

Contoh Fungsi insert

```
void insert(int e) {
  node *curr;
  node *tmp;

  curr = malloc(sizeof(node));
if(!curr) {
    printf("alokasi gagal..\n");
    return;  }
curr->elm = e;

// bila list masih kosong
// node diletakkan di awal

  if(!start->next) {
    start->next = curr;
    curr->next = NULL;
  } else {
    tmp = malloc(sizeof(node));

  if(!tmp) {
      printf("alokasi gagal..\n");
      return;}

    // bila list tidak kosong
    // lakukan perulangan selama
    // isi node berikutnya lebih besar dari
    // isi node yang akan dimasukkan

  tmp = start;
    while(tmp->next && tmp->next->elm <= e) {
      tmp = tmp->next;  }

    curr->next = tmp->next;
    tmp->next = curr;}
count++; }
```

Berukutnya adalah membuat fungsi delete() yang digunakan untuk menghapus node berdasar isinya.

Contoh fungsi delete

```
void delete(int e) {
  node *tmp, *curr;

  // jika list kosong
```

```
  if(!start->next) {
    printf("List kosong..\n");
    return;   }

tmp = malloc(sizeof(node));
  curr = malloc(sizeof(node));

if(!(tmp && curr)) {
    printf("alokasi gagal..\n");
    return;  }

tmp = start;  // posisikan tmp di start
       // lakukan perulangan selama isi node tidak ketemu

while(tmp->next && tmp->next->elm != e) {
    tmp = tmp->next;   }

  // bila node berikutnya adalah NULL
  // tidak tidak ditemukan

if(!tmp->next) {
    printf("tidak ada node berisi elemen '%d'..\n", e);
    return;   }
  curr = tmp->next;
  tmp->next = curr->next;

  free(curr);
  count--;  }
```

Kemudian membuat fungsi show() yang digunakan untuk membaca isi linked list. Pembacaan berdasarkan arah rangkaian.

Contoh fungsi show

```
void show(void) {
  node *tmp;

if(!start->next) {
    printf("List kosong..\n");
    return;  }
  tmp = malloc(sizeof(node));

if(!tmp) {
    printf("alokasi gagal..\n");
    return;   }

  printf("Jumlah: %d\n", count);
  printf("Isi   : ");

tmp = start;
  while(tmp = tmp->next) {
    printf("%d ", tmp->elm);
  }
printf("\n");}
```

Terakhir adalah membuat fungsi main() untuk mendemokan operasi – operasi yang sudah dibuat sebelumnya.

Contoh fungsi main

```
int main(void) {
  char s[80];
  char e[80];

init();
  for(;;) {
    system("clear");
    printf("Linked List. Jumlah: %d\n", count);
    printf("1. Insert\n");
    printf("2. Show\n");
    printf("3. Delete\n");
    printf("4. Exit\n");
    printf("   : ");
    gets(s);

    if(strcmp(s, "") == 0)
      break;

    switch(*s) {
      case '1' :
        printf(" Insert: ");
        gets(e);
        insert(atoi(e));
        break;
      case '2' :
        show();
        system("pause");
        break;
      case '3' :
        printf(" Delete: ");
        gets(s);
        delete(atoi(s));
        break;
      case '4' :
        exit(0);
        break;
    }   }
return 0; }
```

3. LATIHAN

Gabungkanlah semua potongan kode di atas sehingga menjadi kode program lengkap.