MODUL PRAKTIKUM

2018

Pemrograman C++
ALGORITMA & STRUKTUR DATA

Rizki Muliono, S.Kom, M.Kom

TEKNIK INFORMATIKA
UNIVERSITAS MEDAN AREA

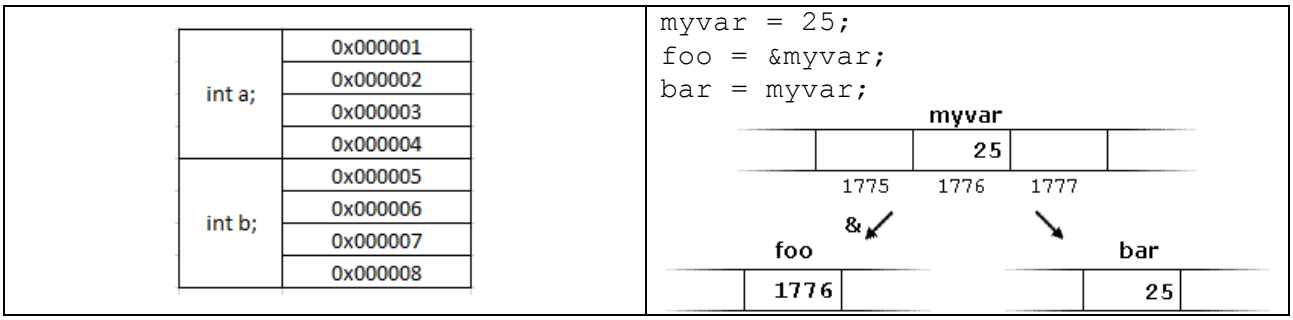
DAFTAR ISI

BAB. 1 POINTER 1
1.1. Defenisi Pointer 1
1.2. Operator Pointer 1
1.3. Mendeklarasikan Variabel Pointer 2
1.4. Pointer pada Pointer 3
1.5. Pointer pada Array 3
1.6. Pointer pada String 4
BAB. 2 ARRAY 5
2.1. Array Satu Dimensi 5
2.2. Array Dua Dimensi 7
BAB. 3 STRUCTURE 10
BAB. 4 LINKED LIST 12
4.1. Single Linked List 12
4.2. Duble Linked List 17
BAB. 5 STACK 20
5.1. Definisi Stack 20
5.2. Stack dengan Array 20
5.3. Stack dengan Single Linked List 22
BAB. 6 QUEUE 24
6.1. Definisi Queue 24
6.2. Implementasi Queue dengan Linear Array 24
6.3. Implementasi Queue dengan Circular Array 25
6.4. Implementasi Queue dengan Double Linked List 27
BAB. 7 TREE 29
8.1. Definisi Tree 29
8.2. Jenis-Jenis Tree 29
BAB. 8 GRAPH 32
8.1. Defenisi Grap 32
8.2. Graph Pencarian Jalur Terpendek 32
DAFTAR PUSTAKA 35

BAB. 1 POINTER

1.1. Defenisi Pointer

Pointer adalah suatu variabel penunjuk, berisi nilai yang menunjuk alamat suatu lokasi memori tertentu. Jadi pointer tidak berisi nilai data, melainkan berisi suatu alamat memori. Lokasi memori tersebut bisa diwakili sebuah variabel atau juga berupa alamat memori secara langsung.
Misalkan variabel x dan terletak di memori 0x000001. Jika ingin memasukkan nilai 100 kedalam variabel x, maka processor harus membawa nilai 100 tersebut kedalam variabel x yang terletak di alamat memori 0x000001. Hal yang perlu kita ketahui adalah, setiap variabel ternyata memiliki ukuran byte yang berbeda-beda dalam memori. Sebagai contoh suatu variabel bertipe int memiliki ukuran 4 byte dalam memori. Maka variabel tersebut akan menempati 4 kapling lokasi dalam memori, misalkan 0x000001, 0x000002, 0x000003, dan 0x000004. Jika terdapat dua buah variabel bertipe int yang bersebelahan, maka alamat variabel pertama terletak di 0x000001 dan variabel kedua terletak dialamat 0x000005. Memori menggunakan bilangan heksadesimal yang ditandai dengan awalan '0x', sehingga jika suatu variabel menempati blok kesepuluh dalam memori, maka alamatnya adalah 0x00000a.



1.2. Operator Pointer

Ada dua operator yang digunakan pada tipe data pointer yaitu :

a. Operator Deference (&)
Deference (&) merupakan suatu operator yang berfungsi untuk menanyakan alamat dari suatu variabel. Apabila kamu memberikan simbol & pada awal variabel dan mencetak hasilnya pada jendela CLI, maka yang akan tercetak adalah alamat dari variabel tersebut bukan nilai yang ditampung oleh variabel tersebut.

Contoh Program 1

```
1. #include <iostream>
2. using namespace std;
3.
4. int main(){
5. int a = 5;
6. cout<<"Alamat Variabel a adalah :"<<&a<<endl;
7. cout<<"Nilai Variabel a adalah :"<<a<<endl;
8. return 0;
9. }
```

Hasil output programnya adalah :

```
Alamat Variabel a adalah :0x7ffee1b20698
Nilai Variabel a adalah :5
```

Alamat variabel 'a' pada setiap komputer akan berbeda-beda tergantung kompiler dalam mengalokasikan memori untuk suatu variabel.

b. Operator Reference (*)

Reference (*) merupakan suatu operator yang berfungsi menyatakan suatu variabel adalah variabel pointer. Sama halnya dengan operator deference, peletakan simbol operator reference diletakan diawal variabel. Operator reference ini akan membuat suatu variabel pointer untuk menampung alamat.

Contoh Program 2

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main(){
5.  int a=5; //Memberikan nilai 5 pada variabel a
6.  int *b; //Mendeklarasikan variabel b sebagai pointer
7.  b = &a; //Mengkopikan alamat variabel a kedalam variabel pointer b
8.  cout<<"Nilai variabel a adalah "<<a<<endl;
9.  cout<<"Alamat variabel a adalah "<<&a<<endl;
10. cout<<"Isi dari variabel b adalah "<<b<<endl;
11. cout<<"Nilai yang tertampung dalam variabel b adalah "<<*b<<endl;
12.
13. return 0;
14. }
```

Hasil Output programnya adalah :

```
Nilai variabel a adalah 5
Alamat variabel a adalah 0x7fe1fe33086c
Isi dari variabel b adalah 0x7fe1fe33086c
Nilai yang tertampung dalam variabel b adalah 5
```

1.3. Mendeklarasikan Variabel Pointer

Suatu variabel pointer didefinisikan dengan bentuk sebagai berikut :

`tipe_data *nama_variabel`

- **`tipe_data`** dapat berupa sembarang tipe seperti halnya pada pendefinisian variabel bukan pointer.
- **`nama_variabel`** adalah nama variabel pointer.

Contoh Program 3

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.  int x, y;
6.  int *px;
7.
8.  x = 89;
9.  y = x;
10. px = &x;
11.
12. cout << "Nilai x = " << x << endl;
13. cout << "Nilai y = " << y << endl;
14. cout << "Alamat x = " << &x << endl;
15. cout << "Alamat px = " << px << endl;
16. cout << "Nilai px = " << *px << endl;
17.
18. x = 108;
19. cout << "\nNilai x = " << x << endl;
20. cout << "Nilai y = " << y << endl;
21. cout << "Alamat x = " << &x << endl;
22. cout << "Alamat px = " << px << endl;
23. cout << "Nilai px = " << *px << endl;
24.
25. *px = 123;
26. cout << "\nNilai x = " << x << endl;
27. cout << "Nilai y = " << y << endl;
28. cout << "Alamat x = " << &x << endl;
29. cout << "Alamat px = " << px << endl;
30. cout << "Nilai px = " << *px << endl;
31. }
32.
```

Hasil output programnya :

```
Nilai x = 89
Nilai y = 89
Alamat x = 0x77a2b621317c
Alamat px = 0x77a2b621317c
Nilai px = 89
```

```
Nilai x = 108
Nilai y = 89
Alamat x = 0x77a2b621317c
Alamat px = 0x77a2b621317c
Nilai px = 108

Nilai x = 123
Nilai y = 89
Alamat x = 0x77a2b621317c
Alamat px = 0x77a2b621317c
Nilai px = 123
```

1.4. Pointer pada Pointer

Tidak terbatas menunjuk alamat dari suatu variabel, pointer dapat pula menunjuk ke pointer lainnya. Dalam pendeklarasiannya, kita tambahkan pointer reference (*) pada variabel yang akan ditunjuk.

Contoh Program 4

```
1.    #include <iostream>
2.    using namespace std;
3.    int main() {
4.    int x;
5.    int *px; //pointer ke variabel
6.    int **ppx; //pointer ke pointer
7.
8.    x = 175;
9.    px = &x;
10.   ppx = &px;
11.   cout << "Nilai x = " << x << endl;
12.   cout << "Nilai px = " <<*px<<endl;
13.   cout << "Nilai ppx = "<<**ppx<<endl;
14.   return 0;
15.   }
16.
```

Hasil output programnya adalah :

```
Nilai x = 175
Nilai px = 175
Nilai ppx = 175
```

1.5. Pointer pada Array

Pada Array/Larik, pointer hanya perlu menunjukan alamat elemen pertama saja karena alamat array dalam memori sudah disusun secara berurutan.

```
int a[] = {76, 67, 88, 98};
int *pa;
pa = a;
```

Pernyataan pa=a artinya pointer pa menyimpan alamat array a, yang alamatnya diwakili alamat elemen pertama, yaitu a[0]. Kita juga bisa mengganti perintah pa=a dengan pa=&a[0].

Contoh Program 5

```
1.   #include <iostream>
2.   #define MAX 5
3.   using namespace std;
4.
5.   int main() {
6.
7.   int a[MAX];
8.   int *pa; pa = a; //atau pa = &a[0]
9.
10.  for (int i = 0; i < MAX; i++) {
11.   cout << "Masukkan Nilai "<<i+1<<":";
12.   cin >> a[i];
13.  }
14.   cout << endl;
15.   for (int i = 0; i < MAX; i++) {
16.   cout << "Nilai a["<<i<<"]="<<*pa<<endl;
17.   pa++;
18.   }
19.  }
20.
```

Hasil Output programnya adalah :

```
Masukkan Nilai 1 : 23
Masukkan Nilai 2 : 5
Masukkan Nilai 3 : 3
Masukkan Nilai 4 : 56
Masukkan Nilai 5 : 12
Nilai a[0] = 23
Nilai a[1] = 5
Nilai a[2] = 3
Nilai a[3] = 56
Nilai a[4] = 12
```

1.6. Pointer pada String

Contoh kode program pointer pada string dapat dilihat pada contoh berikut :

Contoh Program 6

```
1.  #include <iostream>
2.  #define MAX 5
3.  using namespace std;
4.
5.  int main() {
6.  char nama[] = "Albert Einstein";
7.  char *pNama = nama;
8.
9.  cout << "Nama = " << nama << endl;
10. cout << "pNama = " << pNama << endl;
11. pNama += 7; cout <<"\nSetelah pNama += 7"<<
    endl;
12. cout << "Nama = " << nama << endl;
13. cout << "pNama = " << pNama << endl;
14. }
```

Hasil output programnya adalah :

```
Nama = Albert Einstein
pNama = Albert Einstein

Setelah pNama += 7
Nama = Albert Einstein
pNama = Einstein
```

Contoh Program 7

```
1.  #include <iostream>
2.  #define MAX 5
3.  using namespace std;
4.
5.  int main() {
6.   int x[10] = {0,1,2,3,4,5,6,7,8,9};
7.   int *px;
8.   int i;
9.
16.
10.  for(i=0; i<10; i++)
11.  {
12.     px = &x[i]; //membaca alamat dari x

13.     cout<<x[i]<<" "<<*px<<" "<<px<<endl;

14.  }
15. }
```

Hasil output programnya adalah :

```
0 0 0x7ffee5b5cb50
1 1 0x7ffee5b5cb54
2 2 0x7ffee5b5cb58
3 3 0x7ffee5b5cb5c
4 4 0x7ffee5b5cb60
5 5 0x7ffee5b5cb64
6 6 0x7ffee5b5cb68
7 7 0x7ffee5b5cb6c
8 8 0x7ffee5b5cb70
9 9 0x7ffee5b5cb74
```

BAB. 2 ARRAY

Array adalah suatu tipe data terstruktur yang dapat menyimpan banyak data dengan suatu nama yang sama dan menempati tempat di memori yang berurutan (kontigu) serta bertipe data sama pula. Larik dapat diakses berdasarkan indeksnya. Indeks larik umumnya dimulai dari 0 dan ada pula yang dimulai dari angka bukan 0. Pengaksesan larik biasanya dibuat dengan menggunakan perulangan (*looping*).

2.1. Array Satu Dimensi

Array Satu dimensi tidak lain adalah kumpulan elemen-elemen identik yang tersusun dalam satu baris. Elemen-elemen tersebut memiliki tipe data yang sama, tetapi isi dari elemen tersebut boleh berbeda.

Syntax array : `Type_data name[jumlah_elemen]`
Contoh penggunaan : `int usia[5];`

usia	0	1	2	3	4
	0	0	0	0	0

`int usia[5];` potongan kode ini akan membuat array sebanya 5 elemen dengan nilai default 0. Kita bisa membuat array dengan memberikan nilai langsung pada saat dideklarasi.
Contoh deklarasi nilai array : `int usia[5] = {10, 3, 8, 5, 6};`
Maka elemen array akan berisi :

usia	usia[0]	usia[1]	usia[2]	usia[3]	usia[4]
	10	3	8	5	6

Cara pengaksesan array dengan memanggil nilai index nya : **usia[4]** maka akan menampilkan nilai elemen ke 5 yaitu **6.** Untuk memberi nilai langsung pada nilai index array dengan cara : **usia[4] = 29**; maka isi elemen array ke 5 akan berisi 29 atau bisa juga di simpan ke dalam variabel **a = usia[4];** variabel **a** berisi nilai array index ke 4.

Contoh Program 8

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.  int usia[5] = {12,45,30,40,23};
6.      for(int i=0;i<5; i++)
7.      {
8.      cout<<"Suhu [" <<i<< "] = "<<usia[i]<<endl;
9.      }
10. }
```

Hasil output programnya adalah :

```
Suhu [0] = 12
Suhu [1] = 45
Suhu [2] = 30
Suhu [3] = 40
Suhu [4] = 23
```

Berikut contoh program mencari bilangan terbesar dan terkecil dalam array :

Contoh Program 9

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.  int bil[10] = {12,45,40,23,44,45,78,90,103,2
    9};
6.  int i;
7.  int min = 1000; //asumsi paling minimum
```

```
8.  int maks = -1000; //asumsi paling maksimum
9.      for(i=0;i<10; i++)
10.     {
11.         if(bil[i] > maks)
12.         {
13.             maks = bil[i];
14.         }
15.         if(bil[i] < min)
24.
16.         {
17.             min = bil[i];
18.         }
19.     }
20.
21.     cout<<"Nilai maksismum : "<<maks<<endl;
22.     cout<<"Nilai minimum : "<<min<<endl;
23. }
```

Hasil output programnya adalah :

```
Nilai maksismum : 103
Nilai minimum : 12
```

Contoh program mencari bilangan tertentu dalam array, dan menampilkan seluruh bilangan yang ketemu pada console :

Contoh Program 10

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int bil[10] = {12,45,40,23,44,45,78,90,1
    03,29};
6.      int i,bilcari,jlh;
7.      bool ketemu = false;
8.
9.      jlh = 0;
10.     cout<<"Bilangan yang akan dicari : ";
11.     cin>>bilcari;
12.
13.     for(i=0;i<10; i++)
14.     {
15.       if(bil[i] == bilcari)
16.       {
32.
17. ketemu = true;
18. cout<<"Bilangan ditemukan di elemen : "<<i<<
    endl;
19. jlh++;
20. }
21. }
22.
23. if(ketemu)
24. {
25.  cout<<"Jumlah data : "<< jlh <<endl;
26. }
27. else
28. {
29.   cout<<"Bilangan tersebut tidak ditemukan!"
    <<endl;
30. }
31. }
```

Hasil output programnya adalah :

```
Bilangan yang akan dicari : 45
Bilangan ditemukan di elemen : 1
Bilangan ditemukan di elemen : 5
Jumlah data : 2
```

Atau

```
Bilangan yang akan dicari : 7
Bilangan tersebut tidak ditemukan !
```

Contoh Program menghitung jumlah dan rata-rata bilangan array yang di inputkan :

Contoh Program 11

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.    int nilai[5];
7.    int i,jum;
8.    float rata;
9.    for (i=1; i<=5; i++)
10.   {
11.    cout<<"Masukkan nilai ke-"<<i<< " : ";
12.     cin>>nilai[i];
13.   }
14. jum = 0 ;
15. for(i=1;i<=5;i++)
16. jum = jum + nilai[i];
17. rata = jum / 5;
18. cout<<"Jumlah : "<<jum<<endl;
19. cout<<"Rata-rata : "<<rata<<endl;
20. return 0;
21. }
```

Hasil output programnya adalah :

```
Masukkan nilai tes ke-1 : 2
Masukkan nilai tes ke-2 : 4
Masukkan nilai tes ke-3 : 6
Masukkan nilai tes ke-4 : 7
Masukkan nilai tes ke-5 : 5
Jumlah : 24
Rata-rata : 4
```

Contoh program menampilkan data mahasiswa dalam bentuk array :

Contoh Program 12

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  void data_mahasiswa()
5.  {
6.   string nama [5];
7.   int indeks;
8.   for (indeks=1; indeks <=5; indeks++)
9.   {
10.   cout<<"Masukkan nama mahasiswa ke-
    "<<indeks<<":";
11.   cin >> nama[indeks];
12.  }
13. system("clear");
14. cout <<" Daftar Nama Mahasiswa " << endl;
15. cout <<"---------------------- " << endl;
16. cout <<"No | Nama Mahasiswa " << endl;
17. cout <<"---------------------- " << endl;
18. for (indeks=1; indeks <=5; indeks++)
19.  {
20.   cout <<indeks<<'\t'<<nama[indeks]<<endl;
21.  }
22. }
23.
24. int main()
25. {
26.   data_mahasiswa();
27.   return 0;
28. }
```

Hasil output programnya adalah :

```
Masukkan nama mahasiswa ke-1 : Danu
Masukkan nama mahasiswa ke-2 : Rina
Masukkan nama mahasiswa ke-3 : Jojon
Masukkan nama mahasiswa ke-4 : Yogi
Masukkan nama mahasiswa ke-5 : Dayah
```

```
Daftar Nama Mahasiswa
----------------------
No | Nama Mahasiswa
----------------------
1       Danu
2       Rina
3       Jojon
4       Yogi
5       Dayah
```

2.2. Array Dua Dimensi

Array dua dimensi merupakan kumpulan dari array satu dimensi terdiri dari baris dan kolom. Misal a[2][3] maka terbentuk array dengan 2 x 3 = 6 elemen array, 2 baris dan dan 3 kolom.

Syntax array dua dimensi :

```
Type_data name[jumlah_elemen_baris][jumlah_elemen_kolom]
```

Contoh penggunaan : `int usia[2][3];`

		0	1	2
	0	0	0	0
usia	1	0	0	0

`int usia[2][3];` potongan kode ini akan membuat array sebanya 6 elemen dengan nilai default 0. Kita bisa membuat array dengan memberikan nilai langsung pada saat dideklarasi.

Contoh deklarasi nilai array :

```
int usia[2][3] = {{1,2,3},{4,5,6}};
Atau dengan mengisi langsung sesuai titik index nya :
```

```
usia[0][0] = 1; // posisi (1, 1)
usia[0][1] = 2; // posisi (1, 2)
usia[0][2] = 3; // posisi (1, 3)
usia[1][0] = 4; // posisi (2, 1)
usia[1][1] = 5; // posisi (2, 2)
usia[1][2] = 6; // posisi (2, 3)
```

Maka elemen array akan berisi :

usia[][]

1 [0][0]	2 [0][1]	3 [0][3]
4 [1][0]	5 [1][1]	6 [1][2]

Mengakses array dengan memanggil nilai index nya : **usia[0][3]** maka akan menampilkan nilai elemen ke 3 yaitu **3.**

Contoh program array dua dimensi :

Contoh Program 13

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.      int x[2][3] = {{1, 2, 3}, {4, 5, 6}}; // insialisasi data
6.      int i, j;
7.
8.      for (i=0; i<2; i++) { // for yang pertama
9.          for (j=0; j<3; j++) { // for yang kedua
10.             printf("%d ", x[i][j]); // mencetak isi array
11.         }
12.         printf("\n"); // mencetak enter pada baris terakhir.
13.     }
14.     return 0;
15. }
```

Hasil output programnya adalah :

```
1 2 3
4 5 6
```

Contoh Program penjumlahan dua buah matriks 2x2 :

Contoh Program 14

```
1.  #include <iostream>
2.  #define Nmaks 10
3.  using namespace std;
4.  typedef int matrik[Nmaks][Nmaks];
5.
6.  void gotoxy(int x, int y) {
7.      printf("\033[%d;%dH", y, x);
8.  }
9.  int main(void) {
10.     int n,i,j;
11.     matrik A,B,C;
12.     cout<<"Program Penjumlahan Matrik A 2x2
    dan B 2x2"<<endl;
13.     cout<<endl;
14.     n=2;
15.     cout<<"Masukkan NIlai-
    nilai Matrik A"<<endl;
16.     for(i=1; i<=n; i++)
17.     {
18.         for(j=1; j<=n; j++)
19.         {
20.             cout<<"A["<<i<<","<<j<<"] = ";
21.             cin>>A[i][j];
22.         }
23.     }
24.     system("clear");
25.     cout<<"Masukkan NIlai-
    nilai Matrik B"<<endl;
26.     for(i=1; i<=n; i++)
27.     {
28.         for(j=1; j<=n; j++)
29.         {
30.             cout<<"B["<<i<<","<<j<<"] = ";
31.             cin>>B[i][j];
32.         }
33.     }
```

```
34.     system("clear");
35.     cout<<endl;
36.     //proses penjumlahan Matrik C = A + B
37.     for(i=1; i<=n; i++)
38.     {
39.         for(j=1; j<=n; j++)
40.         {
41.             C[i][j] = A[i][j] + B[i][j];
42.         }
43.     }
44.     system("clear");
45.     cout<<"Nilai-
    nilai Matrik A,B,C"<<endl;
46.     //Menampilkan isi Matrik A
47.     gotoxy(1,5);
48.     cout<<"A = ";
49.     for(i=1; i<=n; i++)
50.     {
51.         for(j=1; j<=n; j++)
52.         {
53.             gotoxy(2+4*j,2+2*i);
54.             cout<<A[i][j];
55.         }
56.     }
57.     //Menampilkan isi Matrik B
58.     gotoxy(1,10);
59.     cout<<"B = ";
60.     for(i=1; i<=n; i++)
61.     {
62.         for(j=1; j<=n; j++)
63.         {
64.             gotoxy(2+4*j,7+2*i);
65.             cout<<B[i][j];
66.         }
67.     }
```

```
68.     //Menampilkan isi Matrik C
69.     gotoxy(1,15);
70.     cout<<"C = ";
71.     for(i=1; i<=n; i++)
72.     {
73.         for(j=1; j<=n; j++)
74.         {
75.             gotoxy(2+4*j,12+2*i);
76.             cout<<A[i][j];
77.         }
78.     }
79.
80.     gotoxy(12,15);
81.     cout<<" + ";
82.     for(i=1; i<=n; i++)
83.     {
84.         for(j=1; j<=n; j++)
85.         {
86.             gotoxy(13+4*j,12+2*i);
87.             cout<<B[i][j];
88.         }
89.     }
90.
91.     gotoxy(23,15);
92.     cout<<" = ";
93.     for(i=1; i<=n; i++)
94.     {
95.         for(j=1; j<=n; j++)
96.         {
97.             gotoxy(24+4*j,12+2*i);
98.             cout<<C[i][j];
99.         }
100.    }
101.   cout<<endl;
102. }
```

Hasil output programnya adalah :

```
Program Penjumlahan Matrik A 2x2 dan B 2x2

Masukkan NIlai-nilai Matrik A
A[1,1] = 2
A[1,2] = 4
A[2,1] = 1
A[2,2] = 5
```

```
Masukkan NIlai-nilai Matrik B
B[1,1] = 4
B[1,2] = 5
B[2,1] = 2
B[2,2] = 9
```

```
Nilai-nilai Matrik A,B,C

     2   4
A =
     1   5


     4   5
B =
     2   9


     2   4       4   5       6   9
C =          +           =
     1   5       2   9       3   14
```

BAB. 3 STRUCTURE

Structure (struktur) adalah kumpulan atau kelompok elemen-elemen data yang digabungkan menjadi satu kesatuan. Masing-masing elemen data tersebut dikenal dengan sebutan field. Field data tersebut dapat memiliki tipe data yang sama ataupun berbeda. Walaupun field-field tersebut berada dalam satu kesatuan, masing-masing field tersebut tetap dapat diakses secara individual. Dalam bahasa pemrograman lain sebuah structure di sebut juga sebagai record dan setiap header kolom disebut field.

Bentuk umum :

```
struct namastruct
{
    <tipe data> field1;
    <tipe data> field2;
    <tipe data> field3;
};
```

Contoh deklarasi penggunaan :

```
struct mahasiswa
{
    char nim[10];
    char nama[50];
    char alamat[100];
    float ipk;
};
```

untuk pemanggilan field pada struktur dengan menambahkan simbol titik (.) misal ingin menampilkan nim mahasiswa di layar :

```
cout<<mahasiswa.nim;
```

Contoh program mengisi data mahasiswa dan ipk dengan struktur :

Contoh Program 15

```
1.  #include <iostream>
2.  #include <string>
3.  using namespace std;
4.  struct mahasiswa
5.  {
6.      char nim[10];
7.      char nama[50];
8.      char alamat[100];
9.      float ipk;
10. };
11. int main()
12. {
13.  mahasiswa mhs;
14.  cout<<"NIM \t\t: ";
15. cin.getline(mhs.nim,15);
16. cout<<"Nama \t\t: ";
17. cin.getline(mhs.nama,50);
18. cout<<"Alamat \t\t: ";
19. cin.getline(mhs.alamat,100);
20. cout<<"Nilai IPK \t\t: ";cin>>mhs.ipk;
21. cout<<endl;
22. cout<<"NIM Mhs \t\t: "<<mhs.nim<<endl;
23. cout<<"Nama Mhs \t\t: "<<mhs.nama<<endl;
24. cout<<"Alamat Mhs \t\t: "<<mhs.alamat<<endl;
25. cout<<"Nilai IPK Mhs \t\t: "<<mhs.ipk<<end;
26. cout<<endl;
27. }
```

Hasil output programnya adalah :

```
NIM             : 163180001
Nama            : Dimas
Alamat          : Jl. Kolam No 1 Medan
Nilai IPK       : 3.80

NIM Mhs                 : 163180001
Nama Mhs                : Dimas
Alamat Mhs              : Jl. Kolam No 1 Medan
Nilai IPK Mhs           : 3.8
```

Latihan :

1. Buat program menghitung durasi rental warnet, dengan ketentuan perhitungannya:
   30 detik = Rp. 130,-
   Satuan waktu : jam : menit : detik

2. Buat program menghitung jumlah nilai akhir mahasiswa 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

BAB. 4 LINKED LIST

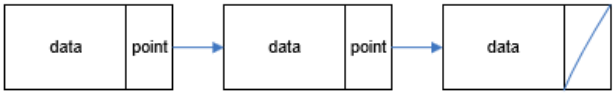
Linked List (Senarai Berantai) adalah jenis struktur data yang berisi kumpulan data yang disusun secara linear dengan setiap data disimpan dalam sebuah simpul dan antara satu simpul dengan simpul lain dihubungkan melalui pointer. Pada linked list ipe data pointer bersifat dinamis, variabel akan dialokasikan hanya pada saat dibutuhkan dan sesudah tidak dibutuhkan dapat direlokasikan kembali. Setiap ingin menambahkan data, Anda selalu menggunakan variabel pointer yang baru, akibatnya Anda akan membutuhkan banyak sekali pointer. Oleh karena itu, ada baiknya jika Anda hanya menggunakan satu variabel pointer saja untuk menyimpan banyak data dengan metode yang kita sebut Linked List. Linked list adalah sekumpulan elemen bertipe sama, yang mempunyai keterurutan tertentu, yang setiap elemennya
terdiri dari dua bagian.

Deklarasi node :

```
struct node
{
     char nama[20];
     int umur;
     float tinggi;
     node *next; // Pointer menyambung ke node selanjutnya
};
```

4.1. Single Linked List

Suatu linked list dikatakan single linked list apabila hanya ada satu pointer yang menghubungkan setiap node (satu arah “next”).



Gambar dari sebuah node :
Bagian **data**, disebut medan informasi, berisi informasi yang akan disimpan dan diolah.
Bagian **pointer**, disebut medan penyambung (link field), berisi alamat simpul berikutnya

Pada gambar di atas, pointer awal menunjuk ke simpul pertama dari senerai tersebut. Medan penyambung (pointer) dari suatu simpul yang tidak menunjuk simpul lain disebut **pointer kosong**, yang nilainya dinyatakan sebagai **null** (**null** adalah kata baku yang berarti bahwa pointer 0 atau bilangan negatif). Jadi kita bisa melihat bahwa dengan hanya sebuah pointer Awal saja maka kita bisa membaca semua informasi yang tersimpan dalam senerai.

Program manipulasi data dengan linked list :

Contoh Program 16

```
1.  #include <stdio.h>
2.  #include <iostream>
3.  using namespace std;
4.  struct TNode{
5.       int data;
6.       TNode *next;
7.  };
8.  TNode *head, *tail;
9.
10. void init(){
11.      head = NULL;
12.      tail = NULL;
13. }
14. int isEmpty(){
15.   if(tail == NULL) return 1;
16.   else return 0;
17. }
18. void insertDepan(int databaru){
19.    TNode *baru;
20.    baru = new TNode;
21.    baru->data = databaru;
22.    baru->next = NULL;
23.    if(isEmpty()==1){
24.            head=tail=baru;
25.            tail->next=NULL;
26.       }
27.    else {
28.           baru->next = head;
29.           head = baru;
30.    }
31.    cout<<"Data masuk\n";
32. }
33. void insertBelakang(int databaru){
34.   TNode *baru,*bantu;
35.   baru = new TNode;
36.   baru->data = databaru;
37.   baru->next = NULL;
38.   if(isEmpty()==1){
39.   head=baru;
40.   tail=baru;
```

```
41.  tail->next = NULL;
42.  }
43.  else {
44.   tail->next = baru;
45.   tail=baru;
46.  }
47.  cout<<"Data masuk\n";
48. }
49.
50. void tampil(){
51.  TNode *bantu;
52.  bantu = head;
53.      if(isEmpty()==0){
54.           while(bantu!=NULL){
55.            cout<<bantu->data<<"->";
56.            bantu=bantu->next;
57.           }
58.           cout<<"NULL";
59.      } else cout<<"Masih kosong\n";
60.   }
61.
62. void hapusDepan(){
63.      TNode *hapus;
64.      int d;
65.      if (isEmpty()==0){
66.           if(head!=tail){
67.            hapus = head;
68.            d = hapus->data;
69.            head = head->next;
70.            delete hapus;
71.           } else {
72.            d = tail->data;
73.            head=tail=NULL;
74.           }
75.      cout<<d<<" terhapus";
76.      } else cout<<"Masih kosong\n";
77. }
78. void hapusBelakang(){
79.      TNode *bantu,*hapus;
80.      int d;
81.      if (isEmpty()==0){
82.       bantu = head;
83.           if(head!=tail){
84.                while(bantu->next!=tail){
85.                 bantu = bantu->next;
86.                }
87.                hapus = tail;
88.                tail=bantu;
89.                d = hapus->data;
90.                delete hapus;
91.                tail->next = NULL;
92.             }else {
93.             d = tail->data;
94.              head=tail=NULL;
95.             }
96.       cout<<d<<" terhapus\n";
97.      } else cout<<"Masih kosong\n";
98. }
99. void clear()
100.   {
101.           TNode *bantu,*hapus;
102.           bantu = head;
103.           while(bantu!=NULL)
104.           {
105.               hapus = bantu;
106.               bantu = bantu->next;
107.               delete hapus;
108.          }
109.          head = NULL;
110.        printf("CLEAR");
111.  }
112.
113.  int main()
114.  {
115.   int pil,databaru;
116.   do
117.   {
118.    system("clear");
119.  cout<<endl<<endl;
120.  cout<<" ========================="<<endl;
121.  cout<<" =  PROGRAM LINKED LIST  ="<<endl;
122.  cout<<" ========================="<<endl;
123.  cout<<" = 1. Insert Depan       ="<<endl;
124.  cout<<" = 2. Insert Belakang    ="<<endl;
125.  cout<<" = 3. Delete Depan       ="<<endl;
126.  cout<<" = 4. Delete Belakang    ="<<endl;
127.  cout<<" = 5. Tampil Data        ="<<endl;
128.  cout<<" = 6. Clear             ="<<endl;
129.  cout<<" = 7. Exit              ="<<endl;
130.  cout<<" ========================="<<endl;
131.  cout<<"   Masukan Pilihan : ";cin>>pil;
132.    switch (pil)
133.       {
134.           case 1: system("clear");{
135.              cout<<"Masukkan Data = ";
136.              cin>>databaru;
137.              insertDepan(databaru);
138.              break;
139.           }
140.           case 2: system("clear");{
141.              cout<<"Masukkan Data = ";
142.              cin>>databaru;
143.              insertBelakang(databaru);
144.              break;
145.           }
146.           case 3: system("clear");{
147.              hapusDepan();
148.              break;
149.           }
150.           case 4: system("clear");{
151.              hapusBelakang();
152.              break;
153.           }
154.           case 5: system("clear");{
155.              tampil();
156.              break;
157.           }
158.           case 6: system("clear");{
159.              clear();
160.              break;
161.          }
162.          case 7: {
163.          return 0;
164.          break;
165.         }
166.    default : system("clear");{
167.    cout<<"\n Maaf, Pilihan yang anda pilih
    tidak tersedia!";
168.         }
169.       }
```

```
170.       }
171.     while(pil!=7);
172.   }
```

Hasil output programnya adalah :

```
   ============================
   =  PROGRAM LINKED LIST     =
   ============================
   = 1. Insert Depan          =
   = 2. Insert Belakang       =
   = 3. Delete Depan          =
   = 4. Delete Belakang       =
   = 5. Tampil Data           =
   = 6. Clear                 =
   = 7. Exit                  =
   ============================
   Masukan Pilihan : 1

   Masukkan Data = 45
   Data masuk
   Masukkan Data = 12
   Data masuk

   Masukkan Data = 78
   Data masuk

   78->12->45->NULL
```

Contoh program data record dengan linked list :

Contoh Program 17

```
1. #include <iostream>
2. using namespace std;
3. struct node
4. {
5.   char nama[20];
6.   int umur;
7.   float tinggi;
8.   node *next;
9. };
10.
11.
12.   node *awal_ptr = NULL;
13.     node *posisi; //digunakan untuk memba
   ca sepanjang list
14.     int option = 0;
15.
16.   void tambah_awal_list()
17.   {
18.     node *baru;
19.     baru = new node;
20.     cout << "Masukkan Nama      : ";
21.     cin >> baru->nama;
22.     cout << "Masukkan Umur      : ";
23.     cin >> baru->umur;
24.     cout << "Masukkan tingggi   : ";
25.     cin >> baru->tinggi;
26.     baru->next = NULL;
27.     if(awal_ptr == NULL)
28.     {
29.       awal_ptr=baru;
30.       awal_ptr->next = NULL;
31.     }
32.     else
33.     {
34.       baru->next = awal_ptr;
35.       awal_ptr = baru;
36.     }
37.   }
38.
39.   void menambah_node_di_akhir()
40.     {
41.     node *temp, *temp2;
42.     // Temporary pointers
43.     // menciptakan node baru
44.     temp = new node;
45.     cout << "Masukkan Nama      : ";
46.     cin >> temp-
   >nama;  cout << "Masukkan Umur      : ";
47.   cin>>temp->umur;
   cout << "Masukkan tingggi   : ";
48.    cin >> temp->tinggi;temp->next=NULL;
49.
50.   // Set up link pada node
51.     if (awal_ptr == NULL)
52.     {
53.       awal_ptr = temp;
54.       posisi = awal_ptr;
55.     }
56.     else
57.     {
58.       temp2 = awal_ptr;
59.       // node tidak NULL list tidak koson
   g
60.       while (temp2->next != NULL)
61.       {
62.         temp2 = temp2->next;
63.         // Memindahkan pada next link dal
   am rantai
```

```
64.       }
65.     temp2->next = temp;
66.     }
67.   }
68.
69.   void display_list()
70.   {
71.     node *temp;
72.     temp = awal_ptr;
73.     cout << endl;
74.     if (temp == NULL)
75.       cout << "List kosong!" << endl;
76.     else
77.     {
78.     cout<<"Nama\t| Umur | tinggi"<<endl;
79.     cout<<"------------------------
  "<<endl;
80.     while (temp != NULL)
81.     {  // Menampilkan detail data
82.     cout << ""<<temp->nama << "";
83.     cout << "\t|  " << temp->umur<<"";
84.     cout << "\t| " <<temp->tinggi;
85.     if (temp == posisi)
86.     cout << " <<<< posisi node";
87.        cout << endl;
88.        temp = temp->next;
89.       }
90.   cout << "->NULL (Akhir list!)"<<endl;
91.     }
92.   }
93.
94.   void hapus_awal_node()
95.   {
96.     node *temp;
97.     temp = awal_ptr;
98.     awal_ptr = awal_ptr->next;
99.     delete temp;
100.  }
101.
102.  void hapus_akhir_node()
103.  {
104.    node *temp1, *temp2;
105.    if (awal_ptr == NULL)
106.      cout << "List kosong!" << endl;
107.    else
108.    {
109.      temp1 = awal_ptr;
110.      if (temp1->next == NULL)
111.      {
112.        delete temp1;
113.        awal_ptr = NULL;
114.      }
115.      else
116.      {
117.        while (temp1->next != NULL)
118.        {
119.          temp2 = temp1;
120.          temp1 = temp1->next;
121.        }
122.        delete temp1;
123.        temp2->next = NULL;
124.      }
125.     }
126.  }
127.
128.  void pindah_posisi_sebelumnya()
129.  {
130.    if (posisi->next == NULL)
131.    cout << "Kamu berada pada akhir list.
   " << endl;
132.    else
133.    posisi = posisi->next;
134.  }
135.
136.  void pindah_posisi_berikutnya()
137.  {
138.    if (posisi == awal_ptr)
139.  cout<<"Berada pada awal list"<<endl;
140.    else
141.    {
142.      node *previous;// deklarasi pointer

143.      previous = awal_ptr;
144.      while (previous->next != posisi)
145.      {
146.        previous = previous->next;
147.      }
148.      posisi = previous;
149.    }
150.  }
151.
152.  void tambah_tengah_list()
153.  {
154.    node *baru, *bantu;
155.    int posisi_sisip;
156.    if(awal_ptr != NULL)
157.    {
158.  cout<<"Akan disisip setelah Data Ke?:";
159.      cin>>posisi_sisip;
160.      bantu=awal_ptr;
161.      baru =new node;
162.    for(int i=1;i<posisi_sisip-1;i++) {
163.       if(bantu->next != NULL)
164.          bantu=bantu->next;
165.       else
166.          break;
167.      }
168.    cout << "Masukkan Nama      : ";
169.    cin >> baru->nama;
170.    cout << "Masukkan Umur      : ";
171.    cin >> baru->umur;
172.    cout << "Masukkan tingggi   : ";
173.    cin >> baru->tinggi;
174.    baru->next=bantu->next;
175.    bantu->next=baru;
176.    }
177.    else
178.    {
179.      cout<<"Belum ada data !! silahkan i
   si data dulu....";
180.      getc;
181.    }
182.  }
183.  void Hapus_tengah_list()
184.  {
185.    int banyakdata,posisi_hapus,poshapus;

186.    node *hapus, *bantu;
187.    if(awal_ptr != NULL)
188.    {
```

```
189.   cout<<" Akan dihapus pada data ke : ";
190.      cin>>posisi_hapus;
191.      banyakdata=1;
192.      bantu=awal_ptr;
193.      while(bantu->next != NULL)
194.      {
195.        bantu=bantu->next;
196.        banyakdata++;
197.      }
198.      if((posisi_hapus<1)||(posisi_hapus>
   banyakdata))
199.      {
200.        cout<<"Belum ada data !! masukkan
    Data dula aja...\n";
201.      }
202.      else
203.      {
204.        bantu=awal_ptr;
205.        poshapus=1;
206.       while(poshapus<(posisi_hapus-1))
207.        {
208.          bantu=bantu->next;
209.          poshapus++;
210.        }
211.        hapus=bantu->next;
212.        bantu->next=hapus->next;
213.        delete hapus;
214.      }
215.    }
216.    else
217.      cout<<"Data Masih kosong, tidak bisa
    hapus data dari tengah! ";
218.    getc;
219.   }
220.
221.
222.   int main()
223.   {
224.    awal_ptr = NULL;
225.    do
226.    {
264.
```

```
227.   system("clear");
228.   display_list();
229.   cout << endl;
230.   cout << "MENU PILIHAN : " <<endl;
231.   cout << "0. Keluar program." <<endl;
232.   cout << "1. Tambah awal list." <<endl;
233.   cout << "2. Tambah akhir list." <<endl;
234.   cout << "3. Tambah tengah list."<<endl;
235.   cout << "4. Hapus awal list."<<endl;
236.   cout << "5. Hapus akhir list."<<endl;
237.   cout << "6. Hapus tengah list."<<endl;
238.   cout << "7. Pindah posisi pointer ke be
   rikutnya." << endl;
239.   cout << "8. Pindah posisi pointer ke se
   belumnya." << endl;
240.   cout << endl << " Pilihan >> ";
241.   cin >> option;
242.
243.   switch (option)
244.     {
245.     case 1 : tambah_awal_list();
246.       break;
247.     case 2 : menambah_node_di_akhir();
248.       break;
249.     case 3 : tambah_tengah_list();
250.       break;
251.     case 4 : hapus_awal_node();
252.       break;
253.     case 5 : hapus_akhir_node();
254.       break;
255.     case 6 : Hapus_tengah_list();
256.       break;
257.     case 7 : pindah_posisi_sebelumnya();

258.       break;
259.     case 8 : pindah_posisi_berikutnya();

260.     }
261.    }
262.   while (option != 0);
263.   }
```

Hasil output programnya adalah :

```
List kosong!

MENU PILIHAN :
0. Keluar program.
1. Tambah awal list.
2. Tambah akhir list.
3. Tambah tengah list.
4. Hapus awal list.
5. Hapus akhir list.
6. Hapus tengah list.
7. Pindah posisi pointer ke berikutnya.
8. Pindah posisi pointer ke sebelumnya.

 Pilihan >> 1
```

```
Masukkan Nama     : sasa
```

```
Masukkan Umur     : 23
Masukkan tingggi  : 100

Nama    | Umur  | tinggi
------------------------
sasa      | 23     | 100
->NULL (Akhir list!)

MENU PILIHAN :
0. Keluar program.
1. Tambah awal list.
2. Tambah akhir list.
3. Tambah tengah list.
4. Hapus awal list.
5. Hapus akhir list.
6. Hapus tengah list.
```

```
7. Pindah posisi pointer ke berikutnya.
8. Pindah posisi pointer ke sebelumnya.

 Pilihan >> 2
```

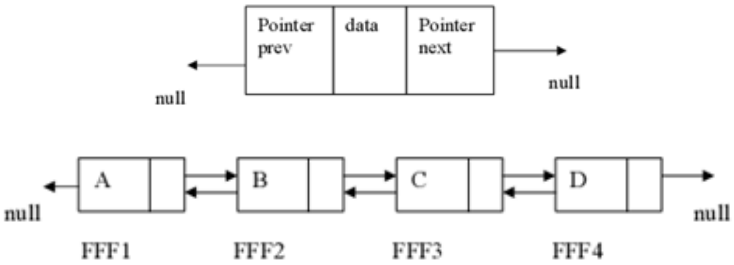
```
Masukkan Nama     : Susi
Masukkan Umur     : 20
Masukkan tingggi  : 140

Nama    | Umur  | tinggi
------------------------
sasa      |  23     | 100
Susi      |  20     | 140
->NULL (Akhir list!)

MENU PILIHAN :
0. Keluar program.
1. Tambah awal list.
2. Tambah akhir list.
3. Tambah tengah list.
4. Hapus awal list.
5. Hapus akhir list.
6. Hapus tengah list.
7. Pindah posisi pointer ke berikutnya.
8. Pindah posisi pointer ke sebelumnya.
```

```
 Pilihan >> 1
```

```
Masukkan Nama     : Jono
Masukkan Umur     : 35
Masukkan tingggi  : 165

Nama    | Umur  | tinggi
------------------------
Jono      |  35     | 165
sasa      |  23     | 100
Susi      |  20     | 140
->NULL (Akhir list!)

MENU PILIHAN :
0. Keluar program.
1. Tambah awal list.
2. Tambah akhir list.
3. Tambah tengah list.
4. Hapus awal list.
5. Hapus akhir list.
6. Hapus tengah list.
7. Pindah posisi pointer ke berikutnya.
8. Pindah posisi pointer ke sebelumnya.

 Pilihan >>
```

4.2. Duble Linked List

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 membutuhkan memori tambahan dengan adanya pointer tambahan tersebut.

Program double linked list

Contoh Program 18

```
1.  #include <iostream>
2.  #include <stdio.h>
3.  #include <iomanip>
4.  using namespace std;
5.  typedef struct node
6.  {
7.      long data;
8.      node* next; //tipe data bertipe sa
    ma untuk menunjuk ke node lain
9.      node* prev;
```

```
};

//buat variabel node
node* head;
node* tail;
node* print;
node* del;
node* entry;

void inisialisasi()
{
      head = NULL;
      tail = NULL;
}

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

void input(int *dta)
{
      entry = new node;
      entry->data = *dta;
      entry->next = NULL;
      entry->prev = NULL;
      if(isEmpty()==1)
      {
            head = entry;
            head->next = NULL;
            head->prev = NULL;
            tail=head;
      }
      else
      {
            tail->next = entry;
            entry->prev = tail;
            tail = entry;
      }
}

//penghapusan data di belakang melalui h
ead
void hapus()
{
   int simpan;
   if(head!=NULL) //jika hanya kondisi i
ni saja maka akan terjadi error karena d
isana ada syntax head->prev = NULL
  {
   if(head->next != NULL)
   {
    del=head;
    simpan = head->data;
    cout<<"\n"<<simpan<<" telah dihapus"
<<endl;
    head = head->next;
    head->prev = NULL;
    delete del;
    }
    else
    {
    simpan = head->data;
    cout<<"\n"<<simpan<<" telah dihapus
"<<endl;
    head = NULL;
    }
   }
  else
 cout<<"\nLinked List kosong penghapusan
 tidak dapat dilakukan"<<endl;
}

void cetak()
{
      print = head;
      if(head!=NULL)
      {
        while(print!=NULL)
         {
          cout<<"\n\t"<<print->data;
          print = print->next;
         }
      }
      else
          cout<<"\nTidak ada data dalam
linked list"<<endl;

}
void menu()
{
      char pilih, ulang;
      int data;
      do
         {
           system("clear");
  menu :
  cout<<"DOUBLE LINKED LIST NONCIRCULAR
"<<endl;
  cout<<"----------------------------
"<<endl;
  cout<<"Menu: ";
  cout<<"\n1. Entry Data";
  cout<<"\n2. Hapus Data";
  cout<<"\n3. Cetak Data";
  cout<<"\n4. Keluar";
  cout<<"\nMasukkan pilihan Anda : ";
  cin>>pilih;
     switch(pilih)
     {
     case '1' :
     cout<<"\nMasukkan Data  : ";
     cin>>data;

     input(&data);
           cout<<"\n"<<data<<" telah di
tambahkan"<<endl;
     break;
     case '2' :
              hapus();
              break;
        case '3'  :
              cetak();
    break;
    case '4' :
    cout<<"\nTerima kasih telah menggun
akan program ini"<<endl;
```

```
129.     exit(EXIT_SUCCESS);
130.     break;
131.    default :
132.    cout<<"\nPilih ulang"<<endl;
133.    goto menu;
134.   }
135.   cout<<"\nKembali ke menu? (y/n)";
136.   cin>>ulang;
137.   }while(ulang=='y' || ulang =='Y');
```

```
138.   }
139.
140.   int main()
141.   {
142.         inisialisasi();
143.         menu();
144.         return EXIT_SUCCESS;
145.   }
```

Hasil output programnya adalah :

```
DOUBLE LINKED LIST NON CIRCULAR
--------------------------------
Menu:
1. Entry Data
2. Hapus Data
3. Cetak Data
4. Keluar
Masukkan pilihan Anda : 1

Masukkan Data  : 23
23 telah ditambahkan

Kembali ke menu? (y/n)y
```

```
DOUBLE LINKED LIST NON CIRCULAR
--------------------------------
Menu:
1. Entry Data
2. Hapus Data
3. Cetak Data
4. Keluar
Masukkan pilihan Anda : 1

Masukkan Data  : 78
78 telah ditambahkan

Kembali ke menu? (y/n)y
```

```
DOUBLE LINKED LIST NON CIRCULAR
--------------------------------
Menu:
1. Entry Data
2. Hapus Data
3. Cetak Data
4. Keluar
Masukkan pilihan Anda : 1

Masukkan Data  : 5
5 telah ditambahkan

Kembali ke menu? (y/n)y
```

```
DOUBLE LINKED LIST NON CIRCULAR
--------------------------------
Menu:
1. Entry Data
2. Hapus Data
3. Cetak Data
4. Keluar
Masukkan pilihan Anda : 1

Masukkan Data  : 3
3 telah ditambahkan

Kembali ke menu? (y/n)y
```

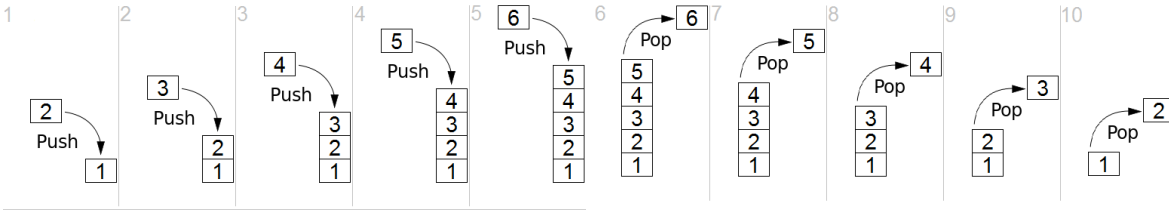
```
DOUBLE LINKED LIST NON CIRCULAR
--------------------------------
Menu:
1. Entry Data
2. Hapus Data
3. Cetak Data
4. Keluar
Masukkan pilihan Anda : 3

       23
       5
Kembali ke menu? (y/n)y
```

```
DOUBLE LINKED LIST NON CIRCULAR
--------------------------------
Menu:
1. Entry Data
2. Hapus Data
3. Cetak Data
4. Keluar
Masukkan pilihan Anda : 3

       23
       5
       78
       3
Kembali ke menu? (y/n)
```

BAB. 5 STACK

5.1. Definisi Stack

Stack adalah suatu tumpukan dari benda. Konsep utamanya adalah LIFO (Last In First Out), benda yang terakhir masuk dalam stack akan menjadi benda pertama yang dikeluarkan dari stack.



Ada dua cara penerapan prinsip stack, yakni dengan array dan linked list. Setidaknya stack haruslah memiliki operasi-operasi sebagai berikut.

- **Push** Untuk menambahkan item pada tumpukan paling atas
- **Pop** Untuk mengambil item teratas
- **Clear** Untuk mengosongkan stack
- **IsEmpty** Untuk memeriksa apakah stack kosong
- **IsFull** Untuk memeriksa apakah stack sudah penuh
- **Retreive** Untuk mendapatkan nilai dari item teratas

5.2. Stack dengan Array

Sesuai dengan sifat stack, pengambilan / penghapusan di elemen dalam stack harus dimulai dari elemen teratas. Operasi-operasi pada Stack dengan Array.

- **IsFull** Fungsi ini memeriksa apakah stack yang ada sudah penuh. Stack penuh jika puncak stack terdapat tepat di bawah jumlah maksimum yang dapat ditampung stack atau dengan kata lain Top = MAX_STACK -1.
- **Push** Fungsi ini menambahkan sebuah elemen ke dalam stack dan tidak bisa dilakukan lagi jika stack sudah penuh.
- **IsEmpty** Fungsi menentukan apakah stack kosong atau tidak. Tanda bahwa stack kosong adalah Top bernilai kurang dari nol.
- **Pop** Fungsi ini mengambil elemen teratas dari stack dengan syarat stack tidak boleh kosong.
- **Clear** Fungsi ini mengosongkan stack dengan cara mengeset Top dengan -1. Jika Top bernilai kurang dari nol maka stack dianggap kosong.
- **Retreive** Fungsi ini untuk melihat nilai yang berada pada posisi tumpukan teratas

Contoh Program :

Program untuk Insert (Push) Nilai dan Delete (Pop) Nilai dalam Stack

Contoh Program 19

```
1.  #include <iostream>
2.  #include <stdio.h>
3.  #include <stdlib.h>
4.  #define MAX 3
5.  using namespace std;
6.
7.  void push(int stack[],int *top,int value);
8.  void pop(int stack[],int *top,int *value);
9.  int main()
10. {
11. int stack[MAX];
12. int top = -1;
13. int n, value;
14. do
15. {
16. do
17. {
18. cout<<"Masukkan Nilai yang akan di Push :";
19. cin>>value;
20. push(stack,&top,value);
21. cout<<"Tekan 1 untuk melanjutkan Push, 2 unt
    uk Pop"<<endl;
22. cin>>n;
23. } while (n == 1);
24. cout<<"Tekan 2 untuk Melakukan Pop"<<endl;
25. cin>>n;
26. while (n == 2)
27. {
28. pop(stack,&top,&value);
29. cout<<"Nilai yang di Pop :"<<value<<endl;
```

```
30. cout<<"Tekan 2 untuk Pop sebuah Elemen, 1 un
    tuk push"<<endl;
31. cin>>n;
32. }
33. cout<<endl;
34. cout<<"Tekan 1 untuk Melanjutkan"<<endl;
35. cin>>n;
36. }while (n == 1);
37. }
38. void push(int stack[], int *top, int value)/
    /fungsi untuk insert nilai
39. {
40. if(*top < MAX)
41. {
42.  *top = *top + 1;
43.  stack[*top] = value;
44. }
45. else
46. {
47. cout<<"Stack Penuh, Push nilai tidak dapat d
    ilakukan"<<endl;
48. exit(0);
49. }
50. }
51.
52. void pop(int stack[],int *top,int *value)//f
    ungsi untuk insert nilai
53. {
54. if(*top >= 0)
55. {
56.   *value = stack[*top];
57.   *top = *top - 1;
58. }
59. else
60. {
61. cout<<"Stack Kosong, Pop Tidak dapat dilakuk
    an!"<<endl;
62.   exit(0);
63.  }
64. }
65.
```

Hasil output programnya adalah :

```
Masukkan Nilai yang akan di Push :23
Tekan 1 untuk melanjutkan Push, 2 untuk Pop
1
Masukkan Nilai yang akan di Push :45
Tekan 1 untuk melanjutkan Push, 2 untuk Pop
1
Masukkan Nilai yang akan di Push :67
Tekan 1 untuk melanjutkan Push, 2 untuk Pop
2
Tekan 2 untuk Melakukan Pop
2
Nilai yang di Pop :67
```

```
Tekan 2 untuk Pop sebuah Elemen, 1 untuk
push
2
Nilai yang di Pop :45
Tekan 2 untuk Pop sebuah Elemen, 1 untuk
push
2
Nilai yang di Pop :23
Tekan 2 untuk Pop sebuah Elemen, 1 untuk
push
2
Stack Kosong, Pop Tidak dapat dilakuk
```

Contoh program Stack dengan array :

Contoh Program 20

```
1.  #include<iostream>
2.  #include<stdio.h>
3.  using namespace std;
4.  #define size 50
5.
6.  struct stack {
7.    int elemen[size];
8.    int top;
9.  };
10. typedef struct stack STACK;
11.
12. // operasi push
13. void push(STACK *p,int value){
14.   if(p->top==size-1)
15.    cout<<"STACK penuh ";
16.   else
17.    p->elemen[++p->top]=value;
18. }
19. //operasi pop
20. int pop(STACK *p)  {
21. if (p->top==-1)
22. {
23.    cout<<"STACK kosong";
24.    return -1;
25.   }
26.   else
27.    return p->elemen[p->top--];
28. }
29. //menampilkan stack
30. void display (STACK *p) {
31.   int i;
32.   if(p->top==-1)
33.    cout<<"\n STACK kosong\n";
34.   else
35.    cout<<"\nIsi STACK adalah : \n";
36.    for (i=p->top;i>=0; --i)
37.     cout<<p->elemen[i]<<"\n";
38. }
39.
40. int main() {
41.   STACK s ;
42.   int x,c,i;
43.   s.top=-1;
44.   do
45.   {
46.    cout<<"MENU PILIHAN";
47.    cout<<"\n1: Operasi PUSH\n";
48.    cout<<"2: Operasi POP\n";
```

```
49.    cout<<"3: Tampilkan Stack\n";
50.    cout<<"4: Hapus Stasck\n";
51.    cout<<"5: Keluar\n";
52.    cout<<"\n\n Pilihan anda : ";cin>>c;
53.    switch(c)   {
54.     case 1:  cout<<"\nMasukkan Elemen Stack:
     ";cin>>x;
55.        push (&s,x);
56.        display(&s);
57.      break;
58.     case 2:  x=pop(&s);
59.      if(x!=-1)
60.       cout<<"\nMenghapus Element = "<<x;
61.      break;
62.     case 3: display(&s);
63.      break;
64.     case 4:
65.     if(s.top==-1)
66.       cout<<endl<<"STACK kosong";
67.     else
68.       cout<<endl<<"STACK dihapus"<<endl;
69.       //Menghapus STACK
70.     for (i=s.top;i>=0; --i)
71.      cout<<"Elemen yang dihapus adalah : "<<
    pop(&s)<<endl;
72.      s.top=-1;
73.    }
74.    getc;
75. system("clear");
76.   }
77. while(c!=5);
78. }
79.
```

Hasil output programnya adalah :

```
MENU PILIHAN
1: Operasi PUSH
2: Operasi POP
3: Tampilkan Stack
4: Hapus Stasck
5: Keluar
 Pilihan anda : 1

Masukkan Elemen Stack: 23

Isi STACK adalah :
23
MENU PILIHAN
1: Operasi PUSH
2: Operasi POP
3: Tampilkan Stack
4: Hapus Stasck
5: Keluar
 Pilihan anda : 1

Masukkan Elemen Stack: 45

Isi STACK adalah :
45
23
MENU PILIHAN
1: Operasi PUSH
2: Operasi POP
3: Tampilkan Stack
4: Hapus Stasck
5: Keluar
 Pilihan anda : 3

Isi STACK adalah :
45
23
MENU PILIHAN
1: Operasi PUSH
2: Operasi POP
3: Tampilkan Stack
4: Hapus Stasck
5: Keluar
 Pilihan anda  :
```

5.3. Stack dengan Single Linked List

Operasi-operasi untuk Stack dengan Linked List :

IsEmpty Fungsi memeriksa apakah stack yang adamasih kosong.

Push Fungsi memasukkan elemen baru ke dalam stack. Push di sini mirip dengan insert dalam single linked list biasa.

Pop Fungsi ini mengeluarkan elemen teratas dari stack

Program Stack dengan link list :

Contoh Program 21

```
1.  #include <iostream>
2.  using namespace std;
3.  //   Creating a NODE Structure
4.  struct node
5.  {
6.  int data;
7.  struct node *next;
8.  };
9.  // Creating a class STACK
10. class stack
11. {
12. struct node *top;
13. public:
14. stack() // constructor
15. {
16. top=NULL;
17. }
18. void push(); // to insert an element
```

```
19. void pop();  // to delete an element
20. void show(); // to show the stack
21. };
22. // PUSH Operation
23. void stack::push()
24. {
25. int value;
26. struct node *ptr;
27. cout<<"\nPUSH Operationn";
28. cout<<"Enter a number to insert: ";
29. cin>>value;
30. ptr=new node;
31. ptr->data=value;
32. ptr->next=NULL;
33. if(top!=NULL)
34. ptr->next=top;
35. top=ptr;
36. cout<<"\nNew item is inserted to the stack!!
    !";
37. }
38. // POP Operation
39. void stack::pop()
40. {
41. struct node *temp;
42. if(top==NULL)
43. {
44. cout<<"\nThe stack is empty!!!";
45. }
46. temp=top;
47. top=top->next;
48. cout<<"\nPOP Operation.\nPoped value is "<<t
    emp->data;
49. delete temp;
50. }
51. // Show stack
52. void stack::show()
53. {
54.  struct node *ptr1=top;
55.  cout<<"\nThe stack is\n";
56. while(ptr1!=NULL)
57. {
58.  cout<<ptr1->data<<" ->";
59.  ptr1=ptr1->next;
60. }
61.  cout<<"NULL\n";
62. }
63. // Main function
64. int main()
65. {
66.  stack s;
67.  int choice;
68.  while(1)
69. {
70. cout<<"\n-----------------------------------
    ";
71. cout<<"\n\t\tSTACK USING LINKED LIST\n\n";
72. cout<<"1:PUSH\n2:POP\n3:DISPLAY STACK\n4:EXI
    T";
73. cout<<"\nEnter your choice(1-4): ";
74. cin>>choice;
75. switch(choice)
76. {
77. case 1:
78.   s.push();
79.   break;
80. case 2:
81.   s.pop();
82.   break;
83. case 3:
84.   s.show();
85.   break;
86. case 4:
87.   return 0;
88.   break;
89. default:
90.   cout<<"\nPlease enter correct choice(1-
    4)!!";
91.   break;
92.  }
93. }
94. return 0;
95. }
```

Hasil output programnya adalah :

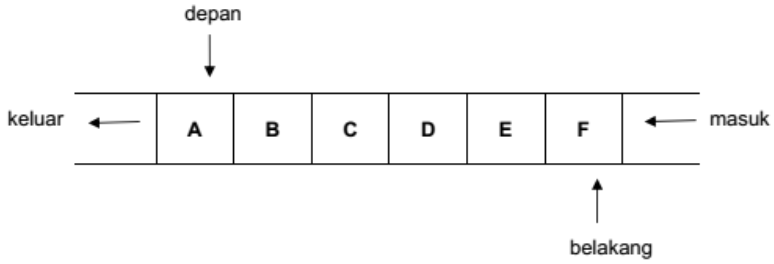
```
-----------------------------------------
              STACK USING LINKED LIST
1:PUSH
2:POP
3:DISPLAY STACK
4:EXIT
Enter your choice(1-4): 1
PUSH OperationnEnter a number to insert: 23
New item is inserted to the stack!!!
-----------------------------------------
              STACK USING LINKED LIST
1:PUSH
2:POP
3:DISPLAY STACK
4:EXIT
Enter your choice(1-4): 1
PUSH OperationnEnter a number to insert: 45
New item is inserted to the stack!!!
-----------------------------------------
              STACK USING LINKED LIST
1:PUSH
2:POP
```

```
3:DISPLAY STACK
4:EXIT
Enter your choice(1-4): 1
PUSH OperationnEnter a number to insert: 67
New item is inserted to the stack!!!
-----------------------------------------
              STACK USING LINKED LIST
1:PUSH
2:POP
3:DISPLAY STACK
4:EXIT
Enter your choice(1-4): 3
The stack is
67 ->45 ->23 ->NULL
-----------------------------------------
              STACK USING LINKED LIST
1:PUSH
2:POP
3:DISPLAY STACK
4:EXIT
Enter your choice(1-4):
```

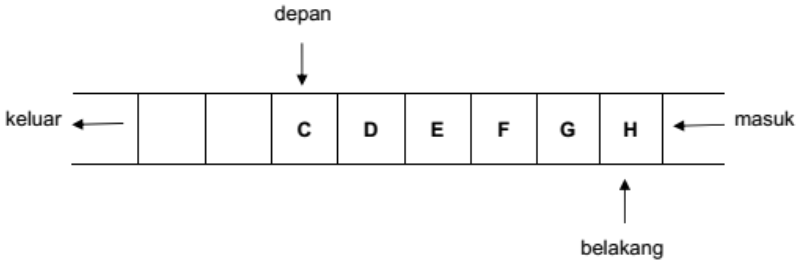
BAB. 6 QUEUE

6.1. Definisi Queue

Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain. Queue bersifat FIFO (First In First Out). Antrian (Queue) merupakan suatu kumpulan data yang penambahan elemennya (masuk antrian) hanya bisa dilakukan pada suatu ujung (disebut dengan sisi belakang/rear) atau disebut juga enqueue yaitu apabila seseorang masuk ke dalam sebuah antrian dan keluar dari antrian adalah dequeue.



Gambar antrian penuh dengan 6 elemen



Gambar antrian dengan 2 data sudah keluar (dequeue).

Struktur data queue setidaknya harus memiliki operasi-operasi sebagai berikut :

EnQueue : Memasukkan data ke dalam antrian
DeQueue : Mengeluarkan data terdepan dari antrian
Clear : Menghapus seluruh antrian
IsEmpty : Memeriksa apakah antrian kosong
IsFull : Memeriksa apakah antrian penuh

6.2. Implementasi Queue dengan Linear Array

Contoh Program 22

```
1.  #include <iostream>
2.  #define MAX 20 //maksimum data queue
3.  using namespace std;
4.
5.  //Deklarasi struct antrian
6.  struct Queue {
7.  int front, rear, data[MAX];
8.  }Q;
9.
10. //cek apakah antrian penuh
11. bool isFull() {
12. return Q.rear == MAX;
13. }
14.
15. //cek apakah antrian kosong
16. bool isEmpty() {
17. return Q.rear == 0;
18. }
19.
20. //Menampilkan Queue
21. void printQueue() {
22. if (isEmpty()) {
23. cout << "Antrian kosong"<<endl;
24. }
25. else {
26. cout << "QUEUE : ";
27. for (int i = Q.front; i < Q.rear; i++)
28.     cout << Q.data[i] << ((Q.rear-
    1 == i) ? "" : ",");
29. cout << endl;
30. }
31. }
32.
33. //manambahkan data ke antrian
34. void enqueue() {
35. if (isFull())
36. {
37. cout << "Antrian penuh!"<<endl;
38. }
39. else {
```

```
40. int data;
41. //menambahkan data ke antrian
42. cout << "Masukkan Data : ";cin >> data;
43. Q.data[Q.rear] = data;
44. //menempatkan tail pada elemen data terakhir
     yang ditambahkan
45. Q.rear++;
46. cout << "Data ditambahkan\n";
47. printQueue();
48. }
49. }
50.
51. // mengambil data dari antrian
52. void dequeue() {
53. if (isEmpty())
54. {
55. cout << "Antrian masih kosong"<<endl;
56. }
57. else{
58. cout << "Mengambil data \"" << Q.data[Q.fron
    t] << "\"..." << endl;
59. //menggeser antrian data ke head
60. for (int i = Q.front; i < Q.rear; i++)
61.     Q.data[i] = Q.data[i + 1];
62. //menempatkan tail pada data terakhir yang d
    igeser
63. Q.rear--;
64. printQueue();
65. }
66. }
```

```
67.
68. int main() {
69. int choose;
70. do
71. {
72. //Tampilan menu
73. cout << "--------------------\n"
74. << "   Menu Pilihan\n"
75. << "--------------------\n"
76. << " [1] Enqueue \n"
77. << " [2] Dequeue\n"
78. << " [3] Keluar \n\n"
79. << "--------------------\n"
80. << "Masukkan pilihan : "; cin >> choose;
81. switch (choose)
82. {
83.     case 1:
84.     enqueue();
85.     break;
86.     case 2:
87.     dequeue();
88.     break;
89.     default:
90.     cout << "Pilihan tidak tersedia";
91.     break;
92. }
93. } while (choose !=3);
94. return 0;
95. }
```

Hasil output programnya adalah :

```
--------------------
   Menu Pilihan
--------------------
 [1] Enqueue
 [2] Dequeue
 [3] Keluar

--------------------
Masukkan pilihan : 1
```

```
Masukkan Data : 23
Data ditambahkan
QUEUE : 23
--------------------
   Menu Pilihan
--------------------
 [1] Enqueue
 [2] Dequeue
 [3] Keluar

--------------------
Masukkan pilihan : 1
```

```
Masukkan Data : 33
Data ditambahkan
QUEUE : 23,33
--------------------
   Menu Pilihan
--------------------
 [1] Enqueue
 [2] Dequeue
 [3] Keluar

--------------------
Masukkan pilihan : 1
```

```
Masukkan Data : 45
Data ditambahkan
QUEUE : 23,33,45
--------------------
   Menu Pilihan
--------------------
 [1] Enqueue
 [2] Dequeue
 [3] Keluar

--------------------
Masukkan pilihan : 2
```

```
Mengambil data "23"...
QUEUE : 33,45
--------------------
   Menu Pilihan
--------------------
 [1] Enqueue
 [2] Dequeue
 [3] Keluar

--------------------
Masukkan pilihan :
```

6.3. Implementasi Queue dengan Circular Array

Contoh Program 23

```
1.  #include <iostream>
2.  #define MAX 5
3.  using namespace std;
4.
5.  class Circular_Queue
6.  {
7.  private:
8.  int *cqueue_arr;
9.  int front, rear;
10. public:
11. Circular_Queue()
12. {
13. cqueue_arr = new int [MAX];
14. rear = front = -1;
15. }
16.
17. void insert(int item)
18. {
19. if ((front == 0 && rear == MAX-
    1) || (front == rear+1))
20. {
21.     cout<<"Queue Penuh \n";
22.     return;
23. }
24. if (front == -1)
25. {
26.     front = 0;
27.     rear = 0;
28. }
29. else
30. {
31.     if (rear == MAX - 1)
32.         rear = 0;
33.     else
34.         rear = rear + 1;
35. }
36. cqueue_arr[rear] = item ;
37. display();
38. }
39.
40. void del()
41. {
42. if (front == -1)
43. {
44.     cout<<"Queue elements : Kosong..!\n";
45.     return ;
46. }
47. cout<<"Element deleted from queue is : "<<c
    queue_arr[front]<<endl;
48. if (front == rear)
49. {
50.     front = -1;
51.     rear = -1;
52. }
53. else
54. {
55.     if (front == MAX - 1)
56.         front = 0;
57.     else
58.         front = front + 1;
59. }
60. display();
61. }
62.
63. void display()
64. {
65. int front_pos = front, rear_pos = rear;
66. if (front == -1)
67. {
68.     cout<<"Queue elements : Kosong..!\n";
69.     return;
70. }
71. cout<<"Queue elements :\n";
72. if (front_pos <= rear_pos)
73. {
74.     while (front_pos <= rear_pos)
75.     {
76.         cout<<cqueue_arr[front_pos]<<"  ";

77.         front_pos++;
78.     }
79. }
80. else
81. {
82.     while (front_pos <= MAX - 1)
83.     {
84.         cout<<cqueue_arr[front_pos]<<"  ";
85.         front_pos++;
86.     }
87.     front_pos = 0;
88.     while (front_pos <= rear_pos)
89.     {
90.         cout<<cqueue_arr[front_pos]<<"  ";
91.         front_pos++;
92.     }
93. }
94. cout<<endl;
95. }
96. };
97.
98. int main()
99. {
100.   int choice, item;
101.   Circular_Queue cq;
102.   do
103.   {
104.   cout<<"1.Insert\n";
105.   cout<<"2.Delete\n";
106.   cout<<"3.Quit\n";
107.   cout<<"Input pilihan : ";
108.   cin>>choice;
109.   switch(choice)
110.   {
111.       case 1:
112.       cout<<"Ketik nilai yang akan di queu
    e : ";
113.       cin>>item;
114.       cq.insert(item);
115.       break;
116.       case 2:
117.       cq.del();
118.       break;
119.       case 3:
120.       break;
121.       default:
122.       cout<<"Pilihan salah.!\n";
```

```
123.    }
124.    }
125.    while(choice != 3);
```

```
126.    return 0;
127.    }
```

Hasil output programnya adalah :

```
1.Insert
2.Delete
3.Quit
Input pilihan : 1
Ketik nilai yang akan di queue : 4
Queue elements :
4
```

```
1.Insert
2.Delete
3.Quit
Input pilihan : 2
Element deleted from queue is : 4
Queue elements :
7  6
```

```
1.Insert
2.Delete
3.Quit
Input pilihan : 1
Ketik nilai yang akan di queue : 7
Queue elements :
4  7
1.Insert
2.Delete
3.Quit
Input pilihan : 1
Ketik nilai yang akan di queue : 6
Queue elements :
4  7  6
```

```
1.Insert
2.Delete
3.Quit
Input pilihan : 2
Element deleted from queue is : 7
Queue elements :
6
```

```
1.Insert
2.Delete
3.Quit
Input pilihan : 2
Element deleted from queue is : 6
Queue elements : Kosong..
```

6.4. Implementasi Queue dengan Double Linked List

Contoh Program 24

```
1.  #include <iostream>
2.  using namespace std;
3.  struct Node {
4.  int data;
5.  Node* next;
6.  };
7.  class Queue {
8.  struct Node* head,* tail;
9.  public:
10. Queue() {
11. head = tail = NULL;
12. }
13. void enQueue();
14. void deQueue();
15. void displayQueue();
16. void menu();
17. int elem;
18. int choice;
19. };
20. void Queue::enQueue() {
21. cout << "Enter your element to be inserted t
    he queue: ";
22. cin >> elem;
23. Node* pointer = new Node;
24. pointer -> data = elem;
25. pointer -> next = NULL;
26. if(head == NULL) {
27. head = pointer;
28. }
29. else
30. tail -> next = pointer;
```

```
31. tail = pointer;
32. cout << "Element has been inserted in the qu
    eue!" << endl;
33. }
34. void Queue::deQueue() {
35. if(head == NULL){
36. cout << "Queue is empty!" << endl;
37. }
38. Node* temp = head;
39. head = head -> next;
40. delete temp;
41. }
42. void Queue::displayQueue() {
43. Node* pointer1 = head;
44. if(head == NULL) {
45. cout << "Queue is empty!" << endl;
46. }
47. else
48. cout << "Elements of your QUEUE!" << endl;
49. while (pointer1 != NULL) {
50. cout << pointer1 -> data << endl;
51. pointer1 = pointer1 -> next;
52. }
53. cout << "End" << endl;
54. }
55. void Queue::menu() {
56. while(1)
57. {
58. cout<<"==================="<<"\n";
59. cout<<"  1. Queue"<<"\n";
60. cout<<"  2. Dequeue"<<"\n";
```

```
61. cout<<"  3. Display Queue"<<"\n";
62. cout<<"  4. Exit"<<"\n";
63. cout<<"==================="<<"\n";
64. cout<<"\nEnter your choice: ";
65. cin>>choice;
66. switch(choice)
67. {
68.     case 1:
69.     enQueue();
70.     break;
71.     case 2:
72.     deQueue();
73.     break;
74.     case 3:
75.     displayQueue();
76.     break;
77.     case 4:
78.     break;
79.     default:
80.     cout<<"Enter choice(1-4)";
81.     break;
82. }
83. }
84. }
85. int main () {
86. Queue frank;
87. frank.menu();
88. }
89.
```

Hasil output programnya adalah :

```
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
====================

Enter your choice: 1
Enter your element to be inserted the
queue: 8
Element has been inserted in the queue!
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
====================

Enter your choice: 1
Enter your element to be inserted the
queue: 6
Element has been inserted in the queue!
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
====================

Enter your choice: 1
Enter your element to be inserted the
queue: 9
Element has been inserted in the queue!
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
```

```
====================

Enter your choice: 3
Elements of your QUEUE!
8
6
9
End
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
====================

Enter your choice: 2
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
====================

Enter your choice: 3
Elements of your QUEUE!
6
9
End
====================
     1. Queue
     2. Dequeue
     3. Display Queue
     4. Exit
====================

Enter your choice:
```

BAB. 7 TREE

8.1. Definisi Tree

Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan hubungan yang bersifat hierarkis (hubungan one to many) antara elemen-elemen. Tree bias didefinisikan sebagai kumpulan simpul/node dengan elemen khusus yang disebut Root. Notde lainnya terbagi menjadi himpunan-himpunan yang saling tak berhubungan satu sama lain (disebut Subtree). Untuk lebih jelasnya, di bawah akan diuraikan istilahistilah umum dalam tree.

Predecessor Node yang berada di atas node tertentu
Successor Node yang berada dibawah node tertentu
Ancestor Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Descendant Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Parent Predecessor satu level di atas suatu node
Child Successor satu level di bawah suatu node
Sibling Node-node yang memiliki parent yang sama dengan suatu node
Subtree Bagian dari tree yang berupa suatu node beserta descendantnya dan memiliki semua karakteristik dari tree tersebut.
Size Banyaknya node dalam suatu tree
Height Banyaknya tingkatan / level dalam suatu tree
Root Satu-satunya node khusus dalam tree yang tak punyak predecessor
Leaf Node-node dalam tree yang tak memiliki successor
Degree Banyaknya child yang dimiliki suatu node

8.2. Jenis-Jenis Tree

Binary Tree
Binary Tree adalah tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal dua subtree dan kedua subtree tersebut harus terpisah. Sesuai dengan definisi tersebut tiap node dalam binary tree hanya boleh memiliki paling banyak dua child.
Jenis- Jenis Binary Tree :
Full Binary Tree
Jenis binary tree ini tiap nodenya (kecuali leaf) memiliki dua child dan tiap subtree harus mempunyai panjang path yang sama.
Complete Binary Tree
Jenis ini mirip dengan Full Binary Tree, namun tiap subtree boleh memiliki panjang path yang berbeda dan setiap node kecuali leaf hanya boleh memiliki 2 child.
Skewed Binary Tree
Skewed Binary Tree adalah Binary Tree yang semua nodenya (kecuali leaf) hanya memiliki satu child.
Implementasi Binary Tree
Binary tree dapat diimplementasikan dalam C++ dengan menggunakan double linkedlist

Contoh Program travesal binary tree, Preorder, Inorder, Postorder :

Contoh Program 25

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  struct node
4.  {
5.      int data;
6.      struct node* left;
7.      struct node* right;
8.  };
9.
10. struct node* newNode(int data)
11. {
12.     struct node* node = (struct node*)
13.     malloc(sizeof(struct node));
14.     node->data = data;
15.     node->left = NULL;
16.     node->right = NULL;
```

```
17.
18.     return(node);
19. }
20.
21. void printPostorder(struct node* node)
22. {
23.     if (node == NULL)
24.     return;
25.     printPostorder(node->left);
26.     printPostorder(node->right);
27.     printf("%d ", node->data);
28. }
29.
30. void printInorder(struct node* node)
31. {
32.     if (node == NULL)
33.     return;
34.     printInorder(node->left);
35.     printf("%d ", node->data);
36.     printInorder(node->right);
37. }
38.
39. void printPreorder(struct node* node)
40. {
41.     if (node == NULL)
42.     return;
43.     printf("%d ", node->data);
44.     printPreorder(node->left);
45.     printPreorder(node->right);
46. }
47.
48. int main()
49. {
50.     struct node *root = newNode(1);
51.     root->left        = newNode(2);
52.     root->right       = newNode(3);
53.     root->left->left  = newNode(4);
54.     root->left->right = newNode(5);
55.
56.     printf("\nPreorder traversal of bina
    ry tree adalah : \n");
57.     printPreorder(root);
58.
59.     printf("\nInorder traversal of binar
    y tree adalah : \n");
60.     printInorder(root);
61.
62.     printf("\nPostorder traversal of bin
    ary tree adalah : \n");
63.     printPostorder(root);
64.
65.     getchar();
66.     return 0;
67. }
```

Hasil output programnya adalah :

```
Preorder traversal of binary tree adalah :
1 2 4 5 3
Inorder traversal of binary tree adalah :
4 2 5 1 3
Postorder traversal of binary tree adalah :
4 5 2 3 1
```

Contoh program menampilkan gambar binary tree :

Contoh Program 26

```
1.  #include <stdio.h>
2.  #include <iostream>
3.  #include <string.h>
4.  #include <stdlib.h>
5.  using namespace std;
6.
7.  typedef struct simpulku *ptrsimpul;
8.  typedef struct simpulku {int data ;
9.  ptrsimpul kanan;
10. ptrsimpul kiri;
11. }simpul;
12. typedef struct {
13. simpul *akar;
14. }pohon;
15. void buatpohonbaru (int dt, pohon *v) {

16. simpul *simpulbaru;
17. simpulbaru=new simpul;
18. simpulbaru->data=dt;
19. simpulbaru->kanan=NULL;
20. simpulbaru->kiri=NULL;
21. (*v).akar=simpulbaru;
22. }
23. void tambahKanan (int dt,simpul **akar){

24. if((*akar)->kanan==NULL){
25. simpul *simpulbaru;
26. simpulbaru=new simpul;
27. simpulbaru->data=dt;
28. simpulbaru->kanan=NULL;
29. simpulbaru->kiri=NULL;
30. (*akar)->kanan=simpulbaru;
31. }else{
32. cout<<"subpohon kanan telah terisi\n";
33. }
34. }
35. void tambahKiri (int dt,simpul **akar){

36. if((*akar)->kiri==NULL) {
37. simpul *simpulbaru;
38. simpulbaru=new simpul;
39. simpulbaru->data=dt;
40. simpulbaru->kanan=NULL;
41. simpulbaru->kiri=NULL;
42. (*akar)->kiri=simpulbaru;
43. }
44. else {
```

```
45. cout<<"Subpohon kiri telah terisi\n";
46. }
47. }
48. void kunjunganPreorder(simpul *akar){
49. if(akar!=NULL){
50. cout<<akar->data<<" ";
51. kunjunganPreorder (akar->kiri);
52. kunjunganPreorder (akar->kanan);
53. }
54. }
55. void kunjunganinorder(simpul *akar){
56. if(akar!=NULL){
57. kunjunganinorder (akar->kiri);
58. cout<<akar->data<<" ";
59. kunjunganinorder (akar->kanan);
60. }
61. }
62. void kunjunganpostorder(simpul *akar){
63. if(akar!=NULL){
64. kunjunganPreorder (akar->kiri);
65. kunjunganPreorder (akar->kanan);
66. cout<<akar->data<<" ";
67. }
68. }
69. int main ()
70. {
95.
```

```
71. pohon v;
72. printf("\n\n\tStruktur Pohon: \n\n");
73. printf("\t        60\n\t       / \\\n\t
       50    20 \n\t    / \\    / \\\n\t  4
   0   55 65 80\n\n ");
74. buatpohonbaru (60,&v);
75. tambahKiri (50,&v.akar);
76. tambahKiri (40,&(v.akar->kiri));
77. tambahKanan (55,& (v.akar->kiri));
78. tambahKanan (20,&v.akar);
79. tambahKanan(65,&(v.akar->kanan));
80. tambahKiri(80,&(v.akar->kanan));
81. cout<<"Kunjungan Preorder"<<endl;
82. cout<<"==================="<<endl;
83. kunjunganPreorder(v.akar);
84. cout<<endl;
85. cout<<"Kunjungan Inorder"<<endl;
86. cout<<"==================="<<endl;
87. kunjunganinorder (v.akar);
88. cout<<endl;
89. cout<<"Kunjungan postorder"<<endl;
90. cout<<"==================="<<endl;
91. kunjunganpostorder (v.akar);
92. cout<<endl;
93. getc;
94. }
```

Hasil output programnya adalah :

```
       Struktur Pohon:

              60
            /  \
           50    20
          / \    / \
        40  55 65  80

 Kunjungan Preorder
===================
60 50 40 55 20 80 65
Kunjungan Inorder
===================
40 50 55 60 80 20 65
Kunjungan postorder
===================
50 40 55 20 80 65 60
```

BAB. 8 GRAPH

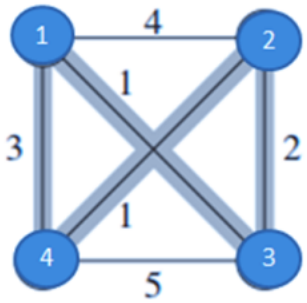
8.1. Defenisi Grap

Teori graph atau teori grafik dalam matematika dan ilmu komputer adalah cabang kajian yang mempelajari sifat-sifat "graf" atau "grafik". Ini tidak sama dengan "Grafika". Secara informal, suatu graf adalah himpunan benda-benda yang disebut "simpul" (vertex atau node) yang terhubung oleh "sisi" (edge) atau "busur" (arc). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan "simpul") yang dihubungkan oleh garis-garis (melambangkan "sisi") atau garis berpanah (melambangkan "busur"). Suatu sisi dapat menghubungkan suatu simpul dengan simpul yang sama. Sisi yang demikian dinamakan "gelang" (loop).

Banyak sekali struktur yang bisa direpresentasikan dengan graf, dan banyak masalah yang bisa diselesaikan dengan bantuan graf. Jaringan persahabatan pada Facebook bisa direpresentasikan dengan graf, yakni simpul-simpulnya adalah para pengguna Facebook dan ada sisi antar pengguna jika dan hanya jika mereka berteman. Perkembangan algoritme untuk menangani graf akan berdampak besar bagi ilmu komputer.
Sebuah struktur graf bisa dikembangkan dengan memberi bobot pada tiap sisi. Graf berbobot dapat digunakan untuk melambangkan banyak konsep berbeda. Sebagai contoh jika suatu graf melambangkan jaringan jalan maka bobotnya bisa berarti panjang jalan maupun batas kecepatan tertinggi pada jalan tertentu. Ekstensi lain pada graf adalah dengan membuat sisinya berarah, yang secara teknis disebut graf berarah atau digraf (directed graph). Digraf dengan sisi berbobot disebut jaringan.

Jaringan banyak digunakan pada cabang praktis teori graf yaitu analisis jaringan. Perlu dicatat bahwa pada analisis jaringan, definisi kata "jaringan" bisa berbeda, dan sering berarti graf sederhana (tanpa bobot dan arah).

8.2. Graph Pencarian Jalur Terpendek



	1	2	3	4
1	0	4	1	3
2	4	0	2	1
3	1	2	0	5
4	3	1	5	0

Di atas kita dapat melihat grafik dan matriks biaya yang lengkap yang mencakup jarak antara masing-masing desa. Kita dapat mengamati bahwa matriks biaya adalah simetris yang berarti jarak antara desa 2 hingga 3 sama dengan jarak antara desa 3 hingga 2.

Masalah di sini adalah penjual keliling ingin mengetahui turnya dengan biaya minimum.
Katakanlah T (1, {2,3,4}), artinya, awalnya dia di desa 1 dan kemudian dia bisa pergi ke salah satu dari {2,3,4}. Dari sana untuk mencapai simpul yang tidak dikunjungi (desa) menjadi masalah baru. Di sini kita dapat mengamati bahwa masalah utama meluas menjadi sub-masalah, ini adalah milik pemrograman dinamis.

Catatan: Saat menghitung di bawah nilai sisi kanan dihitung dengan cara bottom-up. Nilai warna merah diambil dari perhitungan di bawah.

T (1, {2,3,4}) = minimum
= {(1,2) + T (2, {3,4}) 4 + **6** = 10
= {(1,3) + T (3, {2,4}) 1 + **3** = 4

= {(1,4) + T (4, {2,3}) 3 + **3** = 6

Di sini minimal di atas 3 jalur adalah jawaban tetapi kita hanya tahu nilai (1,2), (1,3), (1,4) hal yang tersisa yaitu T (2, {3,4})… adalah masalah baru sekarang . Pertama kita harus menyelesaikannya dan menggantikannya di sini.

T (2, {3,4}) = minimum
= {(2,3) + T (3, {4}) 2 + **5** = 7
= {(2,4) + T {4, {3}) 1 + **5** = 6

T (4, {2,3}) = minimum
= {(4,2) + T (2, {3}) 1 + **2** = 3
= {(4,3) + T {3, {2}) 5 + **2** = 7

T (3, {2,4}) = minimum
= {(3,2) + T (2, {4}) 2 + **1** = 3
= {(3,4) + T {4, {2}) 5 + **1** = 6

T (3, {4}) = (3,4) + T (4, {}) 5 + 0 = 5
T (4, {3}) = (4,3) + T (3, {}) 5 + 0 = 5
T (2, {4}) = (2,4) + T (4, {}) 1 + 0 = 1
T (4, {2}) = (4,2) + T (2, {}) 1 + 0 = 1
T (2, {3}) = (2,3) + T (3, {}) 2 + 0 = 2
T (3, {2}) = (3,2) + T (2, {}) 2 + 0 = 2

Di sini T (4, {}) mencapai kondisi dasar dalam rekursi, yang mengembalikan 0 (nol) jarak.
Di sinilah kita dapat menemukan jawaban akhir, T (1, {2,3,4}) = minimum
= {(1,2) + T (2, {3,4}) 4 + **6** = 10 di jalur ini kita harus menambahkan +1 karena jalur ini berakhir dengan 3.
Dari sana kita harus mencapai 1 jadi 3-> 1 jarak 1 akan ditambahkan jarak total adalah 10 + 1 = 11
= {(1,3) + T (3, {2,4}) 1 + **3** = 4 di jalur ini kita harus menambahkan +3 karena jalur ini berakhir dengan 3.
Dari sana kita harus mencapai 1 jadi 4-> 1 jarak 3 akan ditambahkan jarak total 4 + 3 = 7
= {(1,4) + T (4, {2,3}) 3 + **3** = 6 di jalur ini kita harus menambahkan +1 karena jalur ini berakhir dengan 3.
Dari sana kita harus mencapai 1 jadi 3-> 1 jarak 1 akan ditambahkan jarak total 6 + 1 = 7
Jarak minimum adalah 7 yang mencakup jalur 1-> 3-> 2-> 4-> 1.

Tapi itu tidak menjamin bahwa setiap titik terhubung ke titik lain maka kita mengambil biaya itu sebagai tak terhingga. Setelah itu, kami mengambil minimal di antara semua sehingga jalur yang tidak terhubung mendapatkan infinity dalam perhitungan dan tidak akan dipertimbangkan.

Jika S kosong itu berarti kami mengunjungi semua node, kami mengambil jarak dari node yang terakhir dikunjungi ke node 1 (node pertama). Karena setelah mengunjungi semua dia harus kembali ke simpul awal.

Program mencari jalur terpendek pada Graph :

Contoh Program 27

```
1.  #include<iostream>
2.  using namespace std;
3.  int ary[10][10],completed[10],n,cost=0;

4.  void takeInput()
5.  {
6.      int i,j;
7.      cout<<"Masukkan Jumlah Kota : ";
8.      cin>>n;
9.      cout<<"\nNilai Cost Matrix\n";
10.     for(i=0;i < n;i++)
11.     {
12.         cout<<"\nCost Element Baris ke-
    : "<<i+1<<"\n";
13.         for( j=0;j < n;j++)
14.         cin>>ary[i][j];
15.         completed[i]=0;
16.     }
17.     cout<<"\n\nCost List :";
18.     for( i=0;i < n;i++)
19.     {
20.         cout<<"\n";
21.         for(j=0;j < n;j++)
22.         cout<<"\t"<<ary[i][j];
23.     }
24. }
25.
26. int least(int c)
27. {
28.     int i,nc=999;
29.     int min=999,kmin;
30.     for(i=0;i < n;i++)
31.     {
32.     if((ary[c][i]!=0)&&(completed[i]==0)
    )
33.         if(ary[c][i]+ary[i][c] < min)
```

```
34.         {
35.             min=ary[i][0]+ary[c][i];
36.             kmin=ary[c][i];
37.             nc=i;
38.         }
39.     }
40.
41.     if(min!=999)
42.     cost+=kmin;
43.     return nc;
44. }
45.
46. void mincost(int city)
47. {
48.     int i,ncity;
49.     completed[city]=1;
50.     cout<<city+1<<"--->";
51.     ncity=least(city);
52.     if(ncity==999)
53.     {
54.         ncity=0;
55.         cout<<ncity+1;
56.         cost+=ary[city][ncity];
57.         return;
58.     }
59.     mincost(ncity);
60. }
61. int main()
62. {
63.     takeInput();
64.     cout<<"\n\nJalur Terpendek :\n";
65.     mincost(0); //passing 0 because star
    ting vertex
66.     cout<<"\n\nMinimum Cost : "<<cost<<e
    ndl;
67.     return 0;
68. }
69.
```

Hasil output programnya adalah :

```
Masukkan Jumlah Kota : 4
Nilai Cost Matrix
Cost Element Baris ke-: 1
0
4
1
3

Cost Element Baris ke-: 2
4
0
2
1

Cost Element Baris ke-: 3
1
2
0
5

Cost Element Baris ke-: 4
3
1
5
0

Cost List :
        0       4       1       3
        4       0       2       1
        1       2       0       5
        3       1       5       0

Jalur Terpendek :
1--->3--->2--->4--->1

Minimum Cost : 7
```

DAFTAR PUSTAKA

1. Bruce Eckel.Thinking in C++, Volume 1, 2nd Edition. 2000. Prentice Hall.
2. Desphande P.S., O.G. Kakde (2004). C dan Data Structures. Charles River Media, Inc. Massachusetts
3. Heriyanto, Imam, Budi Raharjo (2003). Pemrograman Borland C++ Builder. Informatika Bandung.
4. Indrajit, Richardus Eko. Manajemen Sistem Informasi dan Teknologi Informasi.
5. Indrajit, Richardus Eko. Kajian Strategis Analisa Cost-Benefit Investasi Teknologi Informasi.
6. Lidya, Leoni, rinaldi Munir (2002). Algoritama dan Pemrograman dalam Bahas Pascal dan C. Informatika Bandung.
7. Sanjaya, Dwi (2005). Asyiknya Belajar Struktur Data di Planet C++. Elex Media Komputindo.
8. Solichin, Achmad (2003). Pemrograman Bahasa C dengan Turbo C. IlmuKomputer.Com
9. https://www.thecrazyprogrammer.com/2017/05/travelling-salesman-problem.html
10. https://www.geeksforgeeks.org/graph-and-its-representations/

DAFTAR ISI

BAB. 1 POINTER 1
1.1. Defenisi Pointer 1
1.2. Operator Pointer 1
1.3. Mendeklarasikan Variabel Pointer 2
1.4. Pointer pada Pointer 3
1.5. Pointer pada Array 3
1.6. Pointer pada String 4

BAB. 2 ARRAY 5
2.1. Array Satu Dimensi 5
2.2. Array Dua Dimensi 7

BAB. 3 STRUCTURE 10

BAB. 4 LINKED LIST 12
4.1. Single Linked List 12
4.2. Duble Linked List 17

BAB. 5 STACK 20
5.1. Definisi Stack 20
5.2. Stack dengan Array 20
5.3. Stack dengan Single Linked List 22

BAB. 6 QUEUE 24
6.1. Definisi Queue 24
6.2. Implementasi Queue dengan Linear Array 24
6.3. Implementasi Queue dengan Circular Array 25
6.4. Implementasi Queue dengan Double Linked List 27

BAB. 7 TREE 29
8.1. Definisi Tree 29
8.2. Jenis-Jenis Tree 29

BAB. 8 GRAPH 32
8.1. Defenisi Grap 32
8.2. Graph Pencarian Jalur Terpendek 32

DAFTAR PUSTAKA 35