04 Introduction to Algorithm

LOOP

sebagai pembentuk

kerangka dasar

Algoritma

69

apakah LOOP itu ?

LOOP adalah sekelompok **instruksi** yang dikerjakan secara berulang-ulang

LOOP adalah suatu **proses** yang terjadi secara berulang-ulang

70

control statement
for dan while
sebagai pembentuk
LOOP

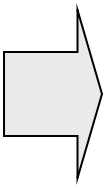
4.1 contoh Penggunaan for( ) dan while( )

Contoh-1 :

```
#include<iostream.h>
void main()
{int I;
 for(I=1; I<=5; I=I+1)
   {
     cout << "\nJakarta";
   }
 cout << "\nSelesai" ;
}
```

```
#include<iostream.h>
void main()
{int I;
 I=1;
 while( I<=5 )
   {
     cout << "\nJakarta";
     I = I + 1;
   }
 cout << "\nSelesai" ;
}
```

Bila kedua program diatas dijalankan, (diRUN) maka keduanya akan tecetak:



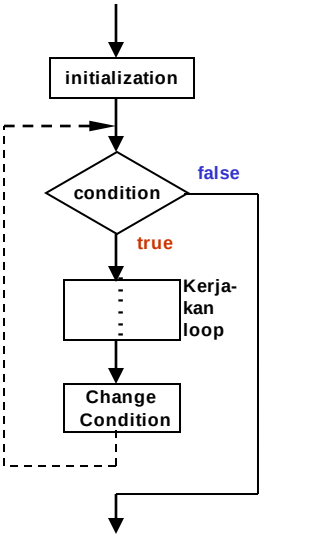
Jakarta
Jakarta
Jakarta
Jakarta
Jakarta
Selesai

71

Bentuk Umum

```
for ( init; cond; chng of cond )
  {
    .
    .  loop
    .
  }
```

```
init;
while ( cond )
  {
    .
    .  loop
    .
    chng of cond
  }
```



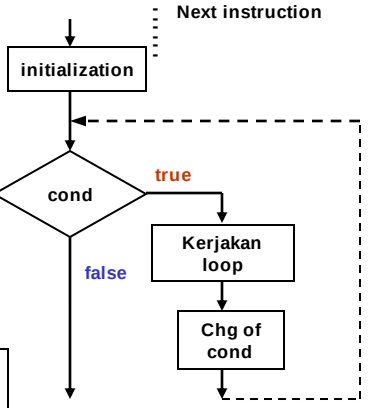
Loop adalah sekumpulan instruksi yang rencananya akan dikerjakan secara berulang-ulang

cond = condition

Suatu pernyataan yang mengandung nilai BENAR (true) atau SALAH (False)

init = inisialisai

Instruksi pemberian suatu nilai yang mempengaruhi nilai condition. Pada proses yang normal, pemberian nilai awal ini akan menyebabkan condition bernilai true.
Instruksi ini hanya pernah satu kali dilaksanakan, yaitu hanya pada saat awal



Chng of cond =

Change of condition

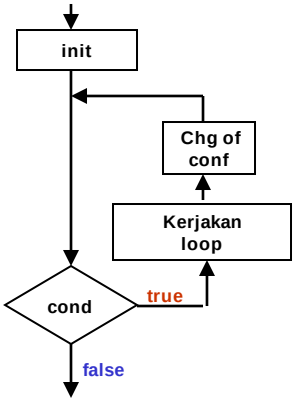
Suatu instruksi yang dapat mempengaruhi nilai condition. Pada proses yang normal, perubahan nilai disini suatu saat akan membuat nilai condition = false

for()

```
#include<iostream.h>
void main()
{ int I;
  for(I=1; I<=5; I=I+1)
     {
        loop
     }
}
```

while()

```
#include<iostream.h>
void main()
{ int I;
  I = 1;
  while(I<=5)
     {
        loop
        I=I+1;
     }
}
```



Berapa Kali Loop Dikerjakan ?

Jawab : 5 kali

72

Contoh bentuk loop yang 'normal'

Contoh-2.

```
#include<iostream.h>
void main()
{ int I;
  for(I=1; I<=5; I=I+1)
     { cout << "\n" << I ;
     }
}
```

Tercetak :
1
2
3
4
5

Bila manggunakan: `while()`

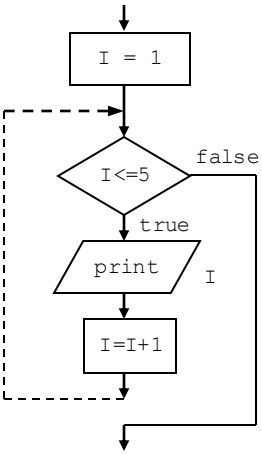
```
#include<iostream.h>
void main()
{ int I;
  I = 1;
  while(I<=5)
    {cout << "\n" << I ;
     I=I+1;
    }
}
```

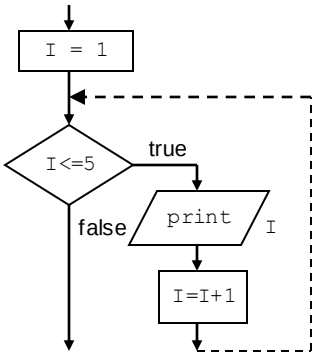
Tercetak :
1
2
3
4
5

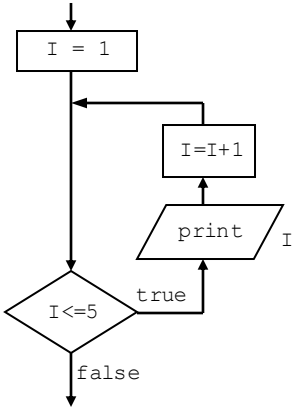
Perkembangan nilai `I`

nilai I	Kondisi I<=5	Tercetak oleh **cout** I	Oleh **I=I+1** nilai **I** menjadi:
1	True	**1**	2
2	True	**2**	3
3	True	**3**	4
4	True	**4**	5
5	True	**5**	6
6	**False**	Keluar dari loop	

For dan **while** diatas cara kerjanya sama. Ada banyak cara menggambarkan loop, baik for maupun while diatas, antara lain ditunjukkan oleh 3 gambar berikut ini :







Perhatikan kembali **Contoh-2** sdiatas sebagai berikut :

`for()`

```
#include<iostream.h>
void main()
{ int I;
  for ( I=1 ; I<=5 ; I=I+1 )
      {
        cout << "\n" << I ;
       }
}
```

I=I+1, dapat ditulis menjadi :
I++ atau ++I (khusus untuk loop)

sehingga menjadi: for (I=1; I<=5; I++)

atau for(I=1; I<=5; ++I)

Buku literatur selalu menggunakan **I++**
untuk I = I+1

73

Beberapa Cara penulisan loop dengan for( ) untuk Contoh-2, yang menghasilkan efek yang sama :

```
{ int I;
  for ( I=1 ; I<=5 ; I++ )
      cout << "\n" << I ;
}
```

Bila instruksi dalam loop hanya ada **1 instruksi**, maka boleh tidak diapit oleh tanda kurung.

```
{ int I;
  for(I=1; I<=5; I++) cout << "\n" << I ;
}
```

Loop boleh langsung ditulis pada baris yang sama dengan for().

```
{ int I;
  I=1;
  for ( ; I<=5; I++ )
 cout << "\n" << I ;
}
```

Inisialisasi dapat ditulis sebelum for( )

Dua instruksi :

```
cout << "\n" << I ;
I++;
```

dapat ditulis jadi satu instruksi :

```
cout << "\n" << I++;
```

yang artinya cetak dulu nilai I, Kemudian I ditambah 1.

```
{ int I;
  I=1;
  for ( ; I<=5 ; )
     { cout << "\n" << I ;
       I++;
     }
}
```

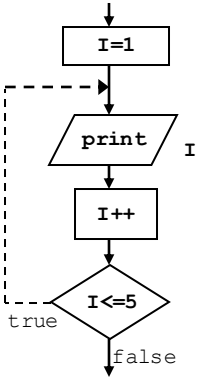
mirip while( )

Perubahan nilai condition, pada instruksi for() dapat ditulis sesudah loop dalam blok yang sama dengan loop

```
{ int I;
  I=1;
  for ( ; I<=5 ; )
      cout << "\n" << I++;
}
```

do while()

```
#include<iostream.h>
void main()
{ int I;
  I=1;
  do
    { cout << "\n" << I;
      I++;
    }
   while( I<=5 );
}
```



Perkembangan nilai I

I	cou %i", I	I++	Kondisi I <= 5
1	**1**	2	True
2	**2**	3	True
3	**3**	4	True
4	**4**	5	True
5	**5**	**6**	False

Tercetak : 1 2 3 4 5

dua instruksi :

```
cout << "\n" << I;
I++;
```

dapat ditulis menjadi :

```
cout << "\n" << I++ ;
```

I dicetak dulu, Kemudian ditambah 1

True, lanjutkan proses
False, keluar dari loop

74

Contoh-3. Didalam loop, nilai condition diubah lebih dulu kemudian mengerjakan proses

for()

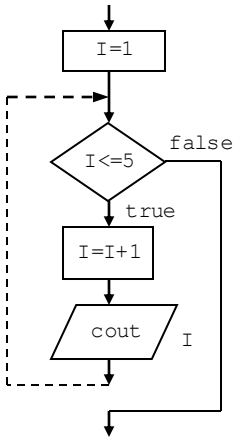
```
#include<iostream.h>
void main()
{ int I;
  for(I=1; I<=5 ; )
     { I++;
       cout << "\n" << I;
     }
}
```

Tercetak : 2 3 4 5 6

while()

```
#include<iostream.h>
void main()
{ int I;
  I = 1;
  while(I<=5)
     { I++;
       cout << "\n" << I;
     }
}
```

Tercetak : 2 3 4 5 6



I ditambah dulu baru dicetak

Sehingga awal cetakan adalah: **2**

dua instruksi ini : I++;
cout << "n" << I ;

dapat ditulis menjadi
Satu instruksi sebagai berikut : cout "\n", ++I ;

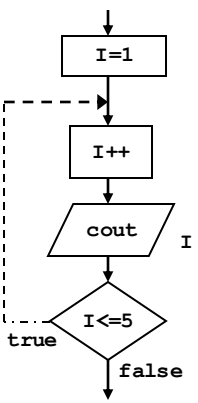
do while()

```
#include<iostream.h>
void main()
{ int I;
  I=1;
  do
     { I++;
       cout << "\n" << I;
       }
  while ( I<=5 );

}
```

Tercetak : 2 3 4 5 6

I ditambah 1 lebih dulu, kemudian baru dicetak



Perkembangan nilai I

I	I++	printf %i", I	Kondisi I <= 5
1	2	**2**	True
2	3	**3**	True
3	4	**4**	True
4	5	**5**	True
5	**6**	**6**	False

True, lanjutkan proses
False, keluar dari loop

75

Contoh-4

Nilai awal condition tidak mesti selalu = 1. Perubahan nilai condition juga tidak mesti selalu +1

```
{ int I;
  for ( I=3 ; I<=10 ; I=I+2 )
      { cout << "\n", I;
      }
}
```

Tercetak:
3
5
7
9

```
{ int I;
  I = 3;
  while( I<=10 )
    { cout "\n", I;
      I=I+2;
    }
}
```

Tercetak:
3
5
7
9

Perkembangan nilai I

nilai I	Kondisi I <= 10	Tercetak oleh cout << I	I=I+2 nilai I menjadi:
3	True	3	5
5	True	5	7
7	True	7	9
9	True	9	11
11	False	Keluar dari loop	

Contoh-5 :

for()

```
#include<iostream.h>
void main()
{ int I;
  for(I=1; I<=5; I=I+1)
   {
     cout <<"\n " << I*2;
   }
}
```

Tercetak :
2
4
6
8
10

Perkembangan nilai I

nilai I	Kondisi I <= 5	Tercetak oleh cout << I*2	I=I+1 nilai I menjadi:
1	True	2	2
2	True	4	3
3	True	6	4
4	True	8	5
5	True	10	6
6	False	Keluar dari loop	

Bila menggunakan: while()

```
#include<iostream.h>
void main()
{ int I;
  I = 1;
  while(I<=5)
   {
    cout <<"\n " << I*2;
    I=I+1;
   }
}
```

Tercetak :
2
4
6
8
10

76

Contoh-6

```
#include<iostream.h>
void main()
{ int I, N;
  N = 8;
  for(I=1; I<=5; I=I+1)
   {
     cout << "\n" << N;
     N = N + 2;
   }
}
```

Tercetak :

8
10
12
14
16

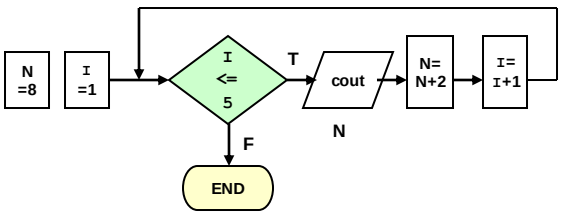
N	I	I<=5	printf N	N= N+2	I= I+1
8	1	True	8	10	2
10	2	True	10	12	3
12	3	True	12	14	4
14	4	True	14	16	5
16	5	True	16	18	6
18	6	False	Keluar dari loop		

Keluar dari loop nilai :
I = 6
N = 18

```
#include<iostream.h>
void main()
{ int I, N;
  N = 8;
  I = 1;
  while(I<=5)
   {
     cout << "\n" << N;
     N = N + 2;
     I=I+1;
   }
}
```

Tercetak :

8
10
12
14
16



Contoh-7

```
#include<iostream.h>
void main()
{ int I, N;
  N = 8;
  for(I=1; I<=5; I=I+1)
   {
     N = N + 2;
     cout << "\n" << N;
   }
}
```

Tercetak :

10
12
14
16
18

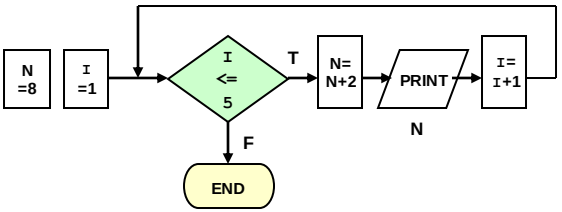
N	I	I<=5	N= N+2	printf N	I= I+1
8	1	True	10	10	2
10	2	True	12	12	3
12	3	True	14	14	4
14	4	True	16	16	5
16	5	True	18	18	6
18	6	False	Keluar dari loop		

Keluar dari loop nilai :
I = 6
N = 18

```
int I, N;
N = 8;
I = 1;
while( I<=5 )
   { N = N + 2;
     cout << "\n" << N;
     I = I + 1;
   }
```

Tercetak :

10
12
14
16
18



77

78

4.2 contoh Aplikasi dasar menggunakan Loop

4.2.1 Mencetak nilai deret

4.2.2 Menghitung dan mencetak total data yang diinput

4.2.3 Mencetak nilai Maksimum / Minimum data yang diinput

4.2.4 Menghitung bunga uang

4.2.5 Menghitung jarak dan waktu perjalanan

4.2.6 Aplikasi matematik

4.2.7 Mengurutkan nilai dalam array satu dimensi

79

4.2.1 Mencetak nilai Deret

Soal-24 : Susun Algoritma yang ditulis dalam Bahasa C++ untuk mencetak 10 suku pertama deret berikut ini :

1, 2, 3, 4, 5, ...

Bila algoritma benar, maka tercetak :

1
2
3
4
5
6
7
8
9
10

Cara-1

```
#include<iostream.h>
void main( )
{ int I;
  for( I=1; I<=10; I=I+1 )
     { cout << "\n" << I ;
     }
}
```

Disini masih dapat diketahui dengan segera bahwa :
Loop dikerjakan 10 kali

I	Tercetak :
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	

I terakhir nilainya = 11, Tapi tidak ikut dicetak Karena sudah keluar dari loop

Cara-2

```
#include<iostream.h>
void main( )
{ int I, N;
  for( I=1; I<=10; I=I+1 )
     { N = I;
       cout << "\n" << N ;
     }
}
```

Dapat diketahui dengan segera bahwa :
Loop dikerjakan 10 kali

I	N
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	

Disini nilai terakhir N = 10

Disini nilai terakhir N = 11, tapi tidak ikut dicetak

Cara-3

```
#include<iostream.h>
void main()
{ int I, N;
  N = 1;
  for( I=1; I<=10; I=I+1 )
     { cout << "\n" << N ;
       N = N + 1;
     }
}
```

Dapat diketahui dengan segera bahwa :
Loop dikerjakan 10 kali

N awalnya =1 dan selalu ditambah 1

Loop dikerjakan 10 kali

I	N
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11

Nilai N pernah = 11 tapi tidak ikut dicetak karena sudah keluar dari loop

80

Soal-25 :

Susun Algoritma yang ditulis dalam Bahasa C++ untuk mencetak 10 suku pertama deret berikut ini :

1, 3, 5, 7, 9, ...

Bila algoritma benar, maka tercetak :

1
3
5
7
9
11
13
15
17
19

Cara-1

```
#include<iostream.h>
void main()
{ int I;
  for( I=1; I<=19; I=I+2 )
     { cout << "\n" << I ;
     }
}
```

Mencari nilai teakhir (contoh disini = 19) tidak selalu mudah

I	Tercetak :
1	1
3	3
5	5
7	7
9	9
11	11
13	13
15	15
17	17
19	19
21	

I terakhir nilainya = 21, Tapi tidak ikut dicetak Karena sudah keluar dari loop

Tidak dapat diketahui dengan segera : Berapa kali loop dikerjakan

Cara-2

```
#include<iostream.h>
void main()
{ int I, N;
  for( I=1; I<=10; I=I+1 )
     { N = I*2 - 1;
       cout << "\n" << N ;
     }
}
```

I	N
1	1
2	3
3	5
4	7
5	9
6	11
7	13
8	15
9	17
10	19
11	

Disini nilai terakhir N = 19

Dapat diketahui dengan segera bahwa : Loop dikerjakan 10 kali

Cara-3

N awalnya =1 dan selalu ditambah 2

Loop tetap dikerjakan 10 kali

```
#include<iostream.h>
void main()
{ int I, N;
  N = 1;
  for( I=1; I<=10; I=I+1 )
     {  cout << "\n" << N ;
        N = N + 2;
     }
}
```

Dapat diketahui dengan segera bahwa : Loop dikerjakan 10 kali

I	N
1	1
2	3
3	5
4	7
5	9
6	11
7	13
8	15
9	17
10	19
11	21

Disini nilai terakhir N = 11, tapi tidak ikut dicetak

Nilai N pernah = 21 tapi tidak ikut dicetak karena sudah keluar dari loop

81

Soal-26 :

Susun Algoritma yang ditulis dalam Bahasa C++ untuk mencetak 10 suku pertama deret berikut ini :

1, 2, 4, 8, 16, ...

Bila algoritma benar, maka tercetak :

```
1
2
4
8
16
32
64
128
256
512
```

Cara-1

```
#include<iostream.h>
void main()
{ int I;
  for( I=1; I<=512; I=I+2 )
     { cout << "\n" << I ;
     }
}
```

Mencari nilai teakhir (contoh disini = 512) tidak selalu mudah

I	Tercetak :
1	1
2	2
4	4
8	8
16	16
32	32
64	64
128	128
256	256
512	512
1024	

I terakhir nilainya = 1024, Tapi tidak ikut dicetak Karena sudah keluar dari loop

Tidak dapat diketahui dengan segera : Berapa kali loop dikerjakan

Cara-2

```
#include<iostream.h>
#include<math.h>
void main()
{ int I, N;
  for( I=1; I<=10; I=I+1 )
     { N = pow(2, I-1 );
       cout << "\n" << N ;
     }
}
```

I	N
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	256
10	512
11	

Disini nilai terakhir N = 512

Dapat diketahui dengan segera bahwa : Loop dikerjakan 10 kali

Cara-3

N awalnya =1 dan selalu dikali 2

Loop tetap dikerjakan 10 kali

```
#include<iostream.h>
void main()
{ int I, N;
  N = 1;
  for( I=1; I<=10; I=I+1 )
     {  cout << "\n" << N ;
        N = N * 2;
     }
}
```

Dapat diketahui dengan segera bahwa : Loop dikerjakan 10 kali

I	N
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	256
10	512
11	1024

Disini nilai terakhir N = 1024, tapi tidak ikut dicetak

Nilai N pernah = 1024 tapi tidak ikut dicetak karena sudah keluar dari loop

82

Soal-26 :

Susun Algoritma yang ditulis dalam Bahasa C++ untuk mencetak 10 suku pertama deret berikut ini :

5, 8, 11, 14, 17, ...

Bila algoritma benar, maka tercetak :

5
8
11
14
17
20
23
26
29
32

Cara-1

```
#include<iostream.h>
void main()
{ int I;
  for( I=1; I<=32; I=I+2 )
     { cout << "\n" << I ;
     }
}
```

Mencari nilai teakhir (contoh disini = 32) tidak selalu mudah

I	Tercetak :
5	5
8	8
11	11
14	14
17	17
20	20
23	23
26	26
29	29
32	32
35	

I terakhir nilainya = 35, Tapi tidak ikut dicetak Karena sudah keluar dari loop

Tidak dapat diketahui dengan segera : Berapa kali loop dikerjakan

Cara-2

```
#include<iostream.h>
#include<math.h>
void main()
{ int I, N;
  for( I=1; I<=10; I=I+1 )
     { N = I * 3 + 2;
       cout << "\n" << N ;
     }
}
```

I	N
1	5
2	8
3	11
4	14
5	17
6	20
7	23
8	26
9	29
10	32
11	

Disini nilai terakhir N = 32

Catatan : `I * 3 + 2`

Berasal dari : `(I-1)*5 + 3`

Selalu naik 3

Awalnya = 5

Dapat diketahui dengan segera bahwa : Loop dikerjakan 10 kali

Cara-3

N awalnya =5 dan selalu ditambah 3

Loop tetap dikerjakan 10 kali

```
#include<iostream.h>
void main()
{ int I, N;
  N = 5;
  for( I=1; I<=10; I=I+1 )
     {  cout << "\n" << N ;
        N = N + 3;
     }
}
```

Dapat diketahui dengan segera bahwa : Loop dikerjakan 10 kali

Disini nilai terakhir N = 35, tapi tidak ikut dicetak

I	N
1	5
2	8
3	11
4	14
5	17
6	20
7	23
8	26
9	29
10	32
11	35

Nilai N pernah = 35 tapi tidak ikut dicetak karena sudah keluar dari loop

83

Soal-27 :

Susun Algoritma yang ditulis dalam Bahasa C++ untuk mencetak 10 suku pertama deret berikut ini :

5, 8, 13, 20, 29, ...

Bila algoritma benar, maka tercetak :

5
8
13
20
29
40
53
68
85
104

Cara-1

```
#include<iostream.h>
#include<math.h>
void main()
{ int I;
  for( I=5; I<=104; I=I+2*sqrt(I-4)+1 )
     { cout << "\n" << I ;
     }
}
```

Mencari nilai teakhir (contoh disini = 104) tidak selalu mudah

Alangkah sulitnya membuat rumus

Tidak dapat diketahui dengan segera :
Berapa kali loop dikerjakan

I	Tercetak :
5	5
8	8
13	13
20	20
29	29
40	40
53	53
68	68
85	85
104	104
125	

Cara-2

```
#include<iostream.h>
#include<math.h>
void main()
{ int I, N;
  for( I=1; I<=10; I=I+1 )
     { N = I * I + 4;
       cout << "\n" << N ;
     }
}
```

Dapat diketahui dengan segera bahwa :
Loop dikerjakan 10 kali

I	N
1	5
2	8
3	13
4	20
5	29
6	40
7	53
8	68
9	85
10	104
11	

Disini nilai terakhir N = 104

X : 3 5 7 9 11 13

N : 5 8 13 20 29 40 53 ..

Cara-3

N awalnya =5 dan selalu ditambah X,
Dan
X awalnya = 3 dan selalu ditambah 2

Loop tetap dikerjakan 10 kali

```
#include<iostream.h>
void main()
{ int I, N, X;
  N = 5;  X = 3;
  for( I=1; I<=10; I=I+1 )
     {  cout << "\n" << N ;
        N = N + X;
        X = X + 2;
     }
}
```

Dapat diketahui dengan segera bahwa :
Loop dikerjakan 10 kali

I	N
1	5
2	8
3	13
4	20
5	29
6	40
7	53
8	68
9	85
10	104
11	125

84

4.2.2 Menghitung dan mencetak total data yang diinput

Soal-28 : Susun program untuk menginput 100 buah bilangan dan mencetak totalnya

```
#include <iostream.h>                          input
void main()                                & cetak total
{ int I,N,T;
  T = 0;
  for(I=1; I <=100; I++)
    { cout << "Nilai ke-" << I << " : ";
      cin >> N;
      T = T + N;
      }
  cout << "\nTotal = : " << T;
}
```

Yang terlihat di layar monitor

```
Nilai ke-1 : 12
Nilai ke-2 : 7
Nilai ke-3 : 10
--
--
--
--
Nilai ke-100 : 15

Total  = : xxxx
```

Nilai T

Soal-29 : Dalam lembar dokumen tersedia banyak sekali data berupa bilangan-bilangan integer. Susun program untuk menginput hanya sebagian dari data tersebut, dan mencetak totalnya. Berapa buah bilangan yang diinput, atau berapa kali kita menginput tergantung total data yang telah diinput. Bila totalnya sudah melebihi 1000, maka berhenti menginput dan langsung mencetak total kemudian proses selesai.

Input s.d Total > 1000

Cara-1

```
int N , T = 0;
while( T <= 1000)
 { cout <<"\nNilai: ";
   cin >> N ;
   T = T + N;
 }
 cout << "Total: " << T;
```

Cara-2

```
int N,T,Flag;
T = 0; Flag = 0;
while( Flag == 0)
  { cout << Nilai: ";
    cin >> N;
    T = T + N;
    if(T > 1000)
       { Flag = 1; }
  }
cout << "Total : " << T);
```

Cara-3

```
int N, T = 0;
do
  { cout << "\nNilai: ";
    cin >> N ;
    T = T + N;
  }
while( T <= 1000);
cout << "Total: " << T;
```

Cara-4

```
int N, T = 0;
while( 1 )
  { cout << "Nilai: ";
    cin >> N ;
    T = T + N;
    if(T > 1000) break;
  }
cout << "Total : " << T);
```

While(1)
agar selalu TRUE

Keluar dari loop bsa dipaksa, misal dengan **break**

85

Soal-30 :

Dalam lembar dokumen ada sejumlah bilangan integer nilai ujian mahasiswa. Ada berapa buah bilangan tersebut tidak diketahui. Yang diketahui adalah akhir dari bilangan adalah sebuah bilangan yang nilainya = 999 suatu bilangan yang dipastikan bukan merupakan nilai ujian.
Susun program untuk menginput bilangan-bilangan tersebut dan menghitung serta mencetak totalnya

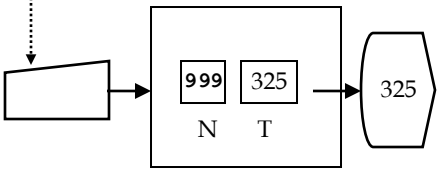
Misal data yang ada
60, 50, 70, 65, 75, 999
Maka ilustrasi input dapat digambarkan sebagai berikut:

60
50
70
65
75
999

Ada 5 buah data, yang Totalnya = 60 + 50 + 70 + 65 + 75 = 325

Hanya sebagai batas (end of data) tidak ikut ditambah

Ada 6 kali input : 5 kali menginput data, dan 1 kali menginput end of data



Cara-1

```
int N, T;
T= 0;
cin >> N;
while (N != 999)
  { T = T + N;
    cin >> N;
  }
cout << "\n", T;
```

Input yang pertama dilakukan disini

Input selanjutnya berulang-ulang dilakukan disini sampai yang diinput = 999 baru keluar dari loop

Cara ini yang biasa dicontohkan dalam buku literatur.

Cara-2.

```
int N,T;
T = 0;
N = 0;
while (N != 999)
  { T = T + N;
    cin >> N;
  }
cout "\n" << T ;
```

Nilai awal N sengaja diberi nilai awal = 0, agar pertama kali (N != 999) akan bernilai TRUE, sehinga paling tidak loop dikerjakan satu kali.

Pertama kali T ditambahkan dengan 0 (nol) sehingga tidak mempengaruhi nilai awal T.

Apa yang terjadi bila bagian program ini ditulis menjadi :

```
while (N != 999)
  { cin >> N ;
    T = T + N;
  }
```

Cara-3

```
int N,T;
int Flag;
T = 0;
Flag = 0;
while (Flag = 0)
 { cin >> N ;
   if (N != 999)
      { T = T + N;
      }
   else
      Flag = 1;

cout << "\n" <<  T ;
```

Bila yang diinput = 999, maka Flag dibuat = 1 agar *condition* pada WHILE bernilai False sehingga keluar dari loop.

Cara-4

```
{int N,T;
 T = 0;
 while ( 1 )
  { cin >> N ;
    if (N == 999)
        break;
    else
        { T = T + N;
        }
   }
 cout << "\n" << T;
}
```

Perhatikan : **while(1)**

Dengan (1) sebagai condition, maka condition while() akan selalu bernilai true, sehingga selalu mengerjakan loop.
Keluar dari loop hanya oleh intruksi **break;**
Instruksi break dilakukan hanya bila nilai yang diinput = 999.

86

Soal-31 :

Dalam lembar dokumen ada sejumlah bilangan integer nilai ujian mahasiswa. Ada berapa buah bilangan tersebut tidak diketahui. Yang diketahui adalah akhir dari bilangan adalah sebuah bilangan yang nilainya = 999 suatu bilangan yang dipastikan bukan merupakan nilai ujian.
Susun program untuk menginput bilangan-bilangan tersebut dan menghitung serta mencetak nilai rata-rata dari semua nilai ujian yang ada.

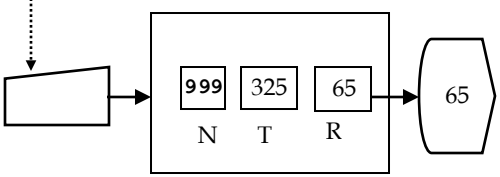
Misal data yang ada
60, 50, 70, 65, 75, 999
Maka ilustrasi input dapat digambarkan sebagai berikut:

60
50
70
65
75
999

Ada 5 buah data, yang Totalnya = 60 + 50 + 70 + 65 + 75 = 325
Nilai rata-rata = 325 / 5 = 65

Hanya sebagai batas (end of data) tidak ikut ditambah

Ada 6 kali input : 5 kali menginput data, dan 1 kali menginput end of data



Cara-1

```
int N, T, Jum, R;
Jum=0;
T= 0;
cin >> N;
while (N != 999)
  { T = T + N;
    Jum++;
    cin >> N;
  }
 R = T / Jum;
 cout << "\n", R;
```

Input yang pertama dilakukan disini

Input selanjutnya berulang-ulang dilakukan disini sampai yang diinput = 999 baru keluar dari loop

Cara ini yang biasa dicontohkan dalam buku literatur

Cara-2.

```
int N,T, Jum, R;
Jum = 0;
T = 0;
N = 0;
while (N != 999)
  { T = T + N;
    Jum++;
    cin >> N;
  }
R = T / Jum;
cout "\n" << R ;
```

Nilai awal N sengaja diberi nilai awal = 0, agar pertama kali (N != 999) akan bernilai TRUE, sehinga paling tidak loop dikerjakan satu kali.

Pertama kali T ditambahkan dengan 0 (nol) sehingga tidak mempengaruhi nilai awal T.

Apa yang terjadi bila bagian program ini ditulis menjadi :

```
while (N != 999)
  { cin >> N ;
    T = T + N;
    Jum++;
  }
```

Cara-3

```
int N,T,Jum;
int R, Flag;
T = 0; Jum = 0;
Flag = 0;
while (Flag = 0)
 { cin >> N ;
   if (N != 999)
      { T = T + N;
        Jum++;
      }
   else
      Flag = 1;
 }
R = T / Jum;
cout << "\n" <<  R ;
```

Bila yang diinput = 999, maka Flag dibuat = 1 agar *condition* pada WHILE bernilai False sehingga keluar dari loop.

Cara-4

```
{int N,T,Jum,R;
 T = 0; Jum =0;
 while ( 1 )
  { cin >> N ;
    if (N == 999)
        break;
    else
       { T = T + N;
         Jum++;
       }
   }
 R = T / Jum;
 cout << "\n" << R;
}
```

Perhatikan : **while(1)**

Dengan (1) sebagai condition, maka condition while() akan selalu bernilai true, sehingga selalu mengerjakan loop.
Keluar dari loop hanya oleh intruksi **break;**
Instruksi break dilakukan hanya bila nilai yang diinput = 999.

87

4.2.3 Mencetak nilai Maksimum / Minimum data yang diinput

Soal-32 :

Susun program untuk menginput 100 buah bilangan yang merupakan nilai ujian mahasiswa, serta mencetak nilai tertinggi yang didapat mahasiswa .

```
int N, I, MAX;
cin >> N;
MAX = N;
for(I=2; I <= 100; I++)
   {  cin >> N;
      if(N > MAX)
            MAX = N ;
   }
 cout << "\n" << Max;
```

Untuk pertama kali, nilai MAX dibuat sama dengan data pertama yang dibaca.

Disini dilakukan 99 kali input. (dari 2 sampai dengan 100).

Bila nilai yang baru dibaca lebih besar dari MAX, maka nilai MAX diganti dengan nilai yang baru.

Ilustrasi Proses :

Bilangan yang diinput (N) :	75	72	75	82	64	50	82	74	82	66	87	72	87	68 . .
Nilai MAX :	75	--	--	82	--	--	--	--	--	--	87	--	--	-- . .

Awalnya MAX dibuat = nilai yang diinput pertama, dalam hal ini = 75

Tanda -- maksudnya nilainya sama dengan nilai Max sebelumnya

Disini Max berubah menjadi 82, atau Diganti menjadi 82

Disini nilai MAX diubah lagi menjadi 87

Sampai disini nilai MAX masih = 87

Soal-33 :

Susun program untuk menginput 100 buah bilangan yang merupakan nilai ujian mahasiswa, serta mencetak nilai tertinggi yang didapat mahasiswa dan juga mencetak ada berapa orang mahasiswa yang mendapat nilai tertinggi tersebut.

Jawab :

```
int N,I, MAX, Jum;
cin >> N;
MAX = N;
Jum = 1;
for(I=2; I < = 100; I++)
  { cin >> N;
    if(N > MAX)
       { MAX = N ;
         Jum = 1;
       }
    else if(N == MAX)
            Jum++;
    }
cout << "\n Terbesar: " << MAX;
cout << "\n Jum Mhs: " << Jum;
```

Untuk pertama kali, nilai MAX dibuat sama dengan data pertama yang dibaca, dan JUM dibuat = 1 (satu)

Disini dilakukan 99 kali input. (dari 2 sampai dengan 100).

Bila nilai yang baru dibaca lebih besar dari MAX, maka nilai MAX diganti dengan nilai yang baru.

Perhatikan :
Setiap kali nilai MAX diganti, nilai JUM kembali menjadi 1 (satu) lagi.

Kalau N sama dengan MAX, maka Jum ditambah 1

Ilustrasi Proses :

Ilustrasi :

Bilangan yang diinput (N) :	75	72	75	82	64	50	82	74	82	66	87	72	87	68 . .
Nilai MAX :	75	--	--	82	--	--	--	--	--	--	87	--	--	-- . .
Nilai JUM :	1	--	2	1	--	--	2	--	3	--	1	--	2	-- . .

Awalnya MAX dibuat = nilai yang diinput pertama, dan JUM = 1

Nilai JUM kembali menjadi 1 begitu nilai MAX diganti

Nilai JUM ditamah 1 bila N == MAX

Sampai disini nilai MAX = 87 nilai JUM = 2

88

Soal-34 :

Susun Algoritma untuk menginput 100 buah bilangan yang merupakan nilai ujian mahasiswa, dan mencetak nilai tertinggi (terbesar) dan yang terendah (terkecil) yang didapat mahasiswa.

Jawab :

```
int N,I, MAX, MIN;
Cin >>N;
MAX = N;
MIN = N;
for( I=2; I <= 100; I++)
   {  cin >> N;
      if(N > MAX)
         MAX = N;
      else
         if(N < MIN)
             MIN = N;
   }
cout << "\nTerbesar : " << MAX
cout << "\nTerkecil : " << MIN;
```

Untuk pertama kali, nilai MAX, dan MIN dibuat sama dengan data pertama yang diinput.

Kalau N lebih besar dari MAX, maka nilai MAX diganti

Tapi kalau N tidak lebih besar dari MAX, ada kemungkinan N lebih kecil dari MIN. Bila N lebih kecil dari MIN, maka nilai MIN yang diganti

Ilustrasi Proses :

Bilangan yang diinput (N) :	75	72	75	82	64	50	82	74	82	66	87	72	87	68..
Nilai MAX :	75	--	--	82	--	--	--	--	--	--	87	--	--	--..
Nilai MIN :	75	72	--	--	64	50	--	--	--	--	--	--	--	--..

Awalnya, MAX dan MIN nilainya dibuat sama dengan data yang pertama dibaca (N)

Nilai MAX tetap 75 tapi nilai MIN diganti menjadi 72

Sampai disini nilai MAX = 87 nilai MIN = 50

89

4.2.4. Menghitung Bunga Uang

Soal-35 :

Seseorang menyimpan uang Rp. 1.000.000 di bank dengan bunga ber-bunga 2% perbulan. Jadi setelah satu bulan uangnya menjadi Rp. 1.020.000. Satu bulan berikutnya uang Rp. 1.020.000 ini mendapat bunga lagi 2%, sehingga setelah 2 bulan uangnya menjadi Rp. 1.020.000 + Rp. 20.400 = Rp. 1.040.400. Demikian seterusnya (bunga bulan ini ditambahkan ke saldo uangnya dan mendapatkan bunga lagi pada bulan berikutnya) . Susun program untuk menghitung dan mencetak jumlah uangnya setelah 10 bulan.

```
#include <iostream.h>
void main()
{float U, B;
 int I;
 U = 1000000.0;
 for(I=1; I<=12;I++)
   {
     B = U * 0.02;
     U = U + B;
   }
 cout << U ;
}
```

Dua instruksi ini dapat diganti menjadi :

```
U = U * 1.02
```

Tanpa menggunakan variabel B

Bu-lan ke-	Jumlah uang		
	Pada awal bulan ke-I	Bunga 2%	Pada akhir bulan ke-I
I	U	B=0.02*U	U=U+B
1	1.000.000	20.000	1.020.000
2	1.020.000	20.400	1.040.400
3	1.040.400	20.808	1.061.208
4	1.061.208	xxxxxx	xxxxxxxxx
--	---------	------	---------
--	---------	------	---------
--	---------	------	---------
10	xxxxxxxxx	xxxxxx	xxxxxxxxx

U = Uang,
B = Bunga

Karena ada perkalian dengan bilangan pecahan, maka nilai terpaksa harus dismpan dalam tipe float. Dicetak dengan format 10.0f agar sen-nya tidak ikut dicetak.
Bila sen ingin dicetak, maka dipakai 10.2.
Bila dicetak dengan **cout**, maka perlu setw(). Lihat Bab 4.

Tercetak :
1.218.994 ← ini yang dicetak

Lihat tabel dibawah

I	U	U + B
1	1.000.000	1.020.000
2	1.020.000	1.040.400
3	1.040.400	1.061.208
4	1.061.208	1.082.432
5	1.082.432	1.104.080
6	1.104.081	1.126.162
7	1.126.162	1.148.685
8	1.148.686	1.171.659
9	1.171.659	1.195.092
10	1.195.093	**1.218.994**
11	1.218.994	1.243.374
12	1.243.374	1.268.241
13	1.268.242	1.293.606
14	1.293.607	1.319.478
15	1.319.479	1.345.868
16	1.345.868	1.372.785
17	1.372.786	1.400.241
18	1.400.241	1.428.246
19	1.428.246	1.456.811
20	1.456.811	1.485.947
21	1.485.947	**1.515.666**

Setelah 1 bulan uangnya menjadi 1.020.000 rupiah

Setelah 21 bulan uangnya sudah mencapai (bahkan melebihi) 1.500.000 rupiah

Soal-36 :

Seseorang menyimpan uang Rp. 1.000.000 di bank dengan bunga ber bunga 2% perbulan, seperti contoh-01 diatas. Susun program untuk menhitung dan mencetak pada bulan ke berapa uangnya mencapai atau sedikit melebihi. 1.500.000 rupiah

Bandingkan dengan rumus yang sering digunakan dalam matematik ekonomi

1000000 * (1.02) ^ 10
= 1218994

pangkat

```
float U, B;
int I;
U = 1000000.0;
I = 0;
while( U <= 1500000.0 )
  {
    B = U * 0.02;
    U = U + B;
    I++;
  }
cout << I ;
```

Tercetak : **21**

Ini yang dicetak

90

Soal-37 :

Seseorang meminjam uang Rp. 1.000.000 dengan bunga 2% perbulan.
Jadi setelah satu bulan pertama utangnya menjadi Rp. 1.020.000.
Pada akhir bulan pertama ia harus mencicil 10% dari utangnya yaitu = Rp. 102.000, sehingga pada awal bulan berikutnya sisa utangnya =
Rp. 1.020.000 - Rp. 102.000 = Rp. 918.000.
Pada akhir bulan kedua utangnya ditambah dengan bunga menjadi :
Rp. 918.000 + Rp. 18.360 = Rp. 936.360.
Pada akhir bulan ke-dua ia harus mencicil 10% dari utang yaitu : Rp. 93.636
Demikian seterusnya sampai 10 bulan.

Susun Algoritma untuk mencetak daftar cicilan yang harus dibayarnya pada setiap akhir bulan selama jangka waktu 10 bulan. Dan juga cetak sisa utangnya setelah 10 kali pembayaran cicilan yang akan dibayar lunas pada awal bulan ke-11.

Uraian Jumlah Utang (U)

Bulan ke-	Utang Pada awal bulan ke-I	Bunga Utang 2%	Utang Pada akhir bulan ke-I	Cicilan Pada akhir bulan ke-I	SisaUtang Pada akhir bulan ke-I
I	U	B=2%*U	U=U+B	C=10%*U	U=U-C
1	1.000.000	20.000	1.020.000	102.000	918.000
2	918.000	18.360	936.360	93.636	842.724
3	842.724	16.854	859.578	85.958	773.621
4	773.621	15.472	789.093	78.909	710.184
5	710.184	14.204	724.387	72.439	651.949
6	651.949	13.039	664.988	66.499	598.489
7	598.489	11.970	610.459	61.046	549.413
8	549.413	10.988	560.401	56.040	504.361
9	504.361	10.087	514.448	51.445	463.003
10	463.003	9.260	472.263	47.226	425.037

Ini sisa utangnya setelah mencicil 10 bulan

10 baris isi kolom ini yang dicetak

```
#include <iostream.h>
void main()
{
 float U, B, C;
 int I;
 U = 1000000.0;
 for( I=1; I <= 10; I++)
   {
     B = U * 0.02;
     U = U + B;
     C = 0.1 * U;
     U = U - C;
     cout << "\nCicilan bulan " << I << C;
   }
 cout << "\n\nSisa Utang : "" << U;
}
```

Tercetak :

```
Cicilan bulan  1 :   102000
Cicilan bulan  2 :    93636
Cicilan bulan  3 :    85958
Cicilan bulan  4 :    78909
Cicilan bulan  5 :    72439
Cicilan bulan  6 :    66499
Cicilan bulan  7 :    61046
Cicilan bulan  8 :    56040
Cicilan bulan  9 :    51445
Cicilan bulan 10 :    47226

Sisa utang : 425037
```


91

4.2.5. Menghitung Waktu dan Jarak Perjalanan

Soal-38 : Seseorang mengendarai sepeda dengan kecepatan tetap 2 m/det. Susun program untuk mencetak berapa m yang dia tempuh setelah berjalan selama 100 detik.

Cara-1

```
#include <iostream.h>
void main()
{
  cout <<  100 * 2;
}
```

Tercetak : 200

Cara-2

```
{int t,v;
 t = 100;
 v = 2;
 cout << t * v ;
}
```

Tercetak : 200

Cara-3

```
{int t,v,s;
 t = 100;
 v = 2;
 s = t * v;
 cout <<  s ;
}
```

Tercetak : 200

Cara-1, 2, 3 : memanfaatkan pengetahuan aritmatika : perkalian

Cara-4

```
#include <iostream.h>
void main()
{int t,v,s;
 s = 0;
 v = 2;
 for(t=1; t<=100; t=t+1 )
   { s = s + v;
   }
 cout <<  s ;
}
```

Tercetak : 200

t = waktu,
v = kecepatan/satuan waktu ,
s = jarak yang ditempuh

t :	1	2	3	4	5	6	100
v :	2	2	2	2	2	2	2
s :	2	4	6	8	12	12	200

Ini yang dicetak

Cara ini yang menjadi pokok bahasan

Cara ini :
memanfaatkan pengetahuan algorima menggunakan loop

92

Soal-39 :

Seseorang mengendarai sepeda dengan kecepatan tetap 2 m/det. Susun program untuk mencetak berapa detik yang dia perlukan untuk menempuh jarak sepanjang 100 m.

Cara-1

```
#include <iostream.h>
void main()
{
  printf("%i ",100/2);
}
```

Tercetak : **50**

Cara-2

```
{int s, v;
 s = 100;
 v = 2;
 printf("%i", s/v);
}
```

Tercetak : **50**

Cara-3

```
{int t,v,s;
 s = 100;
 v = 2;
 t = s / v;
 printf("%i", t );
}
```

Tercetak : **50**

Cara-1, 2, 3 : memanfaatkan pengetahuan matematika : pembagian

Cara-4

```
#include <stdio.h>
void main()
{int t,v,s;
 s = 0;
 v = 2;
 t = 0;
 while( s < 100 )
      { s = s + v;
        t = t + 1;
      }
 printf("%i", t );
}
```

Keluar dari loop , bila S >= 100

Tercetak : **50**

t = waktu,
v = kecepatan/satuan waktu ,
s = jarak yang ditempuh

v :	2	2	2	2	2	2	..	..	..	2
s :	2	4	6	8	12	12	..	..	..	100
t :	1	2	3	4	5	6	..	..	..	50

Ini yang dicetak

93

Soal-40:

Seseorang mengendarai sepeda motor dengan kecepatan yang selalu berubah. Pada detik pertama kecepatannya 2 m/det. Pada detik ke-2 kecepatannya bertambah menjadi 2.1 m/det. Pada detik ke-3 kecepatannya naik lagi menjadi 2.2 m/det. Demikian seterusnya setiap detik kecepatannya selalu bertambah sebesar 0.1 m/det. Susun program untuk mencetak berapa m yang dia tempuh setelah berjalan selama 100 detik.

Lihat! Karena kecepatannya tidak konstant (tapi selalu berubah) maka tidak dapat lagi sekedar memahami aritmatika perkalian.

Terpaksa menggunakan Loop

```
#include <iostream.h>
void main()
{double t,v,s;
 s = 0.0;
 v = 2.0;
 for(t=1; t<=100; t=t+1)
   { s = s + v;
     v = v + 0.1;
   }
 cout <<  s ;
}
```

Tercetak : **695**

t = waktu,
v = kecepatan/satuan waktu ,
s = jarak yang ditempuh

t :	1	2	3	4	5	6	100
v :	2	2.1	2.2	2.3	2.4	2.5	.. .. .. xx.xx
s :	2	4.1	6.3	8.6	11.0	13.5	xxx.xx

Ini yang dicetak

Coba test dengan :

```
for(t=1; t <= 5; t=t+1 )
```

Maka akan tercetak : **11**

Soal-41:

Seseorang mengendarai sepeda motor dengan kecepatan yang selalu berubah. Pada detik pertama kecepatannya 2 m/det. Pada detik ke-2 kecepatannya bertambah menjadi 2.1 m/det. Pada detik ke-3 kecepatannya naik lagi menjadi 2.2 m/det. Demikian seterusnya setiap detik kecepatannya selalu bertambah sebesar 0.1 m/det. Susun program untuk mencetak berapa detik yang dia perlukan untuk menempuh jarak sepanjang 100 m.

```
{double t,v,s;
 s = 0.0;
 v = 2.0;
 t = 0.0;
 while ( s < 100 )
   { s = s + v;
     t = t + 1;
     v = v + 0.1;
   }
 cout << t ;
}
```

Tercetak : **30**

Artinya untuk mencapai jarak 100 m diperlukan waktu 30 detik

t = waktu,
v = kecepatan/satuan waktu ,
s = jarak yang ditempuh

v :	2	2.1	2.2	2.3	2.4	2.5	.. .. .. xx.xx
s :	2	4.1	6.3	8.6	11.0	13.5	100.0
t :	1	2	3	4	5	6	xxx

Coba test dengan :

```
for(t=1; t < 11; t=t+1 )
```

Maka akan tercetak : **5**

94

4.2.6 Aplikasi Matematik

Soal-42:

Bilangan Genap dan Ganjil

Susun program untuk menginput sebuah nilai integer, bilangan **bulat** positip lebih besar dari nol, kemudian cetak perkataan **"EVEN",** bila bilangan tersebut merupakan bilangan **genap**, sebaliknya cetak perkataan **"ODD"** bila bilangan tersebut merupakan bilangan **ganjil**.

Catatan :
Bilangan bulat postip lebih besar dari nol **genap** : 2, 4, 6, 8, dan seterusnya
Bilangan bulat positip lebih besar dari nol **ganjil** : 1, 3, 5,7, dan seterusnya

while()

```
#include <iostream.h>
void main()
{int N;
  cout << "Inputkan sebuah nilai: ";
  cin >> N;
  while (N > 0)
      { N = N - 2 }
  if (N == 0)
      cout << "EVEN";
  else
      cout << "ODD";
}
```

Kurangi **N** dengan 2 terus menerus selama nilainya masih > 0. Loop berakhir bila **N** = 0 atau = -1.

Bila nilai akhirnya = 0, maka **N** adalah bilangan genap, dan bila berakhir dengan -1 berarti **N** = bilangan ganjil

Bila N=8	Bila N=7
8	7
6	5
4	3
2	1
0	-1

N terakhir = -1
Berarti N awal adalahGANJIL

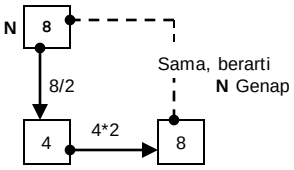
N terakhir = 0
Berarti N awal adalahGENAP

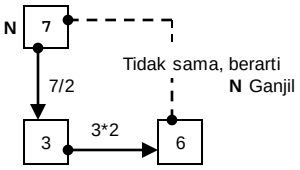
LEBIH PRAKTIS BILA MENGGUNAKAN :

```
cin >> N;
if (N % 2 == 0)
     cout << "EVEN";
 else
     cout << "ODD";
}
```

Bilangan yang habis dibagi 2, Berarti bilangan GENAP

```
cin >> N;
if( N == (N/2) * 2 )
     cout << "EVEN";
else
     cout << "ODD";
```





95

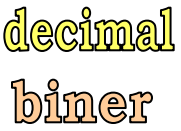
Soal-43:

Susun program untuk menginput sebuah bilangan bulat positip maksimum 255 yang menyatakan suatu bilangan desimal. Kemudian konversi dan cetak nilai desimal tersebut kedalam bentuk bilangan binary

Konversi Decimal ke Binary

Contoh:

```
Bila diinput     maka akan tercetak :
      0          0  0  0  0  0  0  0  0
      7          0  0  0  0  0  1  1  1
      8          0  0  0  0  1  0  0  0
     65          0  1  0  0  0  0  0  1
    127          0  1  1  1  1  1  1  1
    175          1  0  1  0  1  1  1  1
    255          1  1  1  1  1  1  1  1
```



Jawab : Cara-1

```
#include<stdio.h>
#include<conio.h>
void main( )
{ int N, I, X, A;
  clrscr( );
  scanf("%i", &N);
  X = 128;
  for(I=1; I<=8; I++)
    { if( N >= X)
        { N = N - X;
          A = 1;
        }
      else
        {  A = 0;  }
      printf("%3i", A );
      X = X/2;
    }
}
```

nilai **255** = 8 bit : 1 1 1 1 1 1 1 1 dan bit yang paling kiri nilainya = **128**

nilai maksimum = **255** berarti maksimum 8 bit setiap bit satu kali loop

128/2 = 64
64/2 = 32
dan seterusnya menyatakan nilai bit-bit selanjutnya

Jawab : Cara-2

```
#include<stdio.h>
#include<conio.h>
void main( )
{ int N, I, X, A;
  clrscr( );
  scanf("%i", &N);
  X = 128;
  for(I=1; I<=8; I++)
    { A = N/ X;
      printf("%i  ", A);
      N = N % X;
      X = X / 2;
    }
}
```

Catatan : **N % X**
Maksudnya N Mod X.

Contoh : Bila diinput ke N nilai 175
maka tercetak : 1 0 1 0 1 1 1 1
yaitu nilai A

I	N	X	A =N/X	N= N%X	X= X/2
1	175	128	**1**	47	64
2	47	64	**0**	47	32
3	47	32	**1**	15	16
4	15	16	**0**	15	8
5	15	8	**1**	7	4
6	7	4	**1**	3	2
7	3	2	**1**	1	1
8	2	1	**1**	0	0

Soal : Ubah program diatas sehingga

Bila diinput	maka tercetak :
0	0
7	1 1 1
8	1 0 0 0
65	1 0 0 0 0 0 1
127	1 1 1 1 1 1 1
255	1 1 1 1 1 1 1 1

96

Soal-43:

Susun program untuk menginput sebuah bilangan bulat lebih besar dari 1. Kemudian periksa apakah bilangan tersebut adalah bilangan PRIMA atau bukan. Bila bilangan tersebut bilangan prima, maka cetak perkataan "PRIMA", sedangkan bila bilangan tersebut bukan bilangan prima maka cetak perkataan "BUKAN PRIMA".

Bilangan prima

Catatan : Bilangan prima adalah bilangan bulat yang habis dibagi HANYA bila dibagi dengan bilangan itu sendiri, kecuali 1 (satu). Jadi satu (1) bukan bilangan prima

bilangan

Contoh bilangan prima :

2 3 5 7 11 13 17 19 23 29 …. dan seterusnya

4, 6, 8 bukan bilangan prima karena habis dibagi 2
9, 15, 21, juga bukan bilangan prima karena habis dibagi 3

Konsep pemikiran pertama (Cara-1) :

Misal N adalah bilangan yang diinput, maka periksa : apakah N habis dibagi dengan salah satu nilai dibawahnya :

- Bila habis, maka N bukan bilangan prima.
- Bila tidak habis, maka N bilangan prima.

Contoh :

Misal N = 9
Bilangan dibaw ahnya adalah : 2, 3, 4, 5, 6, 7, 8
Catatan : Nilai terkecil adalah 2
bukan 1.
Ternyata N habis dibagi 3, jadi N bukan prima.

Misal N = 10
Bilangan dibaw ahnya adalah : 2, 3, 4, 5, 6, 7, 8, 9
Ternyata N habis dibagi 2, jadi N bukan prima.

Misal N = 11
Bilangan dibaw ahnya adalah : 2, 3, 4, 5, 6, 7, 8, 9, 10
Ternyata N tidak habis dibagi dengan semua bilangan yang ada dibaw ahnya, jadi N adalah **prima**

Misal N = 13
Bilangan dibaw ahnya adalah :
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Ternyata N tidak habis dibagi dengan semua bilangan yang ada dibaw ahnya, jadi N adalah **prima**

Cara-1

```
#include<iostream.h>
void main()
{ int I, N, Flag = 0;
  cin >> N;
  I=2;
  while( I <= N-1)
     { if(N % I == 0)
          Flag = 1;
       I++;
     }
  if(Flag == 1)
     cout << "Bukan Prima";
  else
     cout << "Prima";
}
```

Program ini walaupun benar, tapi tidak efisien. Lihat untuk N = 9; Seharusnya sewaktu N % 3 == 0 true, langsung keluar dan mencetak "Bukan Prima".

```
#include<iostream.h>
void main()
{ int I, N, Flag = 0;
  cin >> N;
  I =2;
  while(I<= N-1 && Flag == 0)
     { if (N % I == 0)
          Flag = 1;
       I++;
     }
  if(Flag == 1)
     cout << "Bukan Prima";
  else
     cout << "Prima";
}
```

Program inipun masih belum efisien, karena selain 2, bilangan prima adalah bilangan ganjil

97

Konsep pemikiran kedua (Cara-2):

Contoh bilangan Prima : 2 3 5 7 11 13 17 19 23 29 31 37 41 .....

Perhatikan: Selain 2 maka semua bilangan PRIMA adalah bilangan GANJIL

Sehingga pemeriksaan apakah N bilangan prima atau bukan, dilakukan sebagai berikut :

1. Bila N = 2 maka cetak "PRIMA", dan proses selesai.

2. Bila N bukan 2 maka periksa :
   - Bila N bilangan GENAP (habis dibagi 2)
     maka cetak "BUKAN PRIMA" dan proses selesai.

   - Bila N bilangan GANJIL, maka periksa apakah bilangan tersebut Prima atau bukan dengan cara sebagai berikut:

Prima

Cara-2

```
#include <iostream.h>
#include <math.h>
main()
{   int N, X, Batas, Flag;
    cin >> N;
    if(N == 2)
         cout << "PRIMA";
    else
        { if ( N%2 == 0 )
              cout << "BUKAN PRIMA";
          else
                {  X = 3;
                   Batas = N-1;
                   Flag = 0;
                   while( Flag == 0 && X <= Batas)
                          {  if ( N % X == 0 )  Flag = 1;
                             X = X + 2;   }
                   if(Flag == 0)
                          cout << "PRIMA";
                   else
                          cout << "BUKAN PRIMA";
                      }
            }
}
```

Batas = N-1

Program inpun masih kurang efisien.
Walaupun sudah dibatasi pada bilangan ganjil,
tapi batasnya masih = N-1,
padahal batas dapat dibuat = $\sqrt{N}$

Contoh bila N = 1000, tidak peru batas = 999
Tapi cukup batas = $\sqrt{N}$ = 31
(diambil angka bulat karena integer)

Jadi cukup diperisa dengan :
3 5 7 9 11 15 17 19 21 23 25 27 29 31 (cukup sampai 31)

Terlihat nanti, inipun tidak efisien,karena : 9, 15, 21, 27 dapat diwakili oleh 3, dan 25 dapat diwakili oleh 5. Jadi sebenarnya cukup dengan bilangan prima <= 31 yaitu : 3 5 7 11 13 17 23 29 31

98

Konsep pemikiran ketiga (Cara-3) :

Membuat batas bukan = N-1, melainkan = $\sqrt{N}$

Sehingga pemeriksaan apakah N bilangan prima atau bukan, dilakukan sebagai berikut :

1. Bila N = 2 maka cetak "PRIMA", dan proses selesai.

2. Bila N bukan 2 maka periksa :
   - Bila N bilangan GENAP (habis dibagi 2) maka cetak "BUKAN PRIMA" dan proses selesai.

   - Bila N bilangan GANJIL, maka periksa apakah bilangan tersebut Prima atau bukan dengan cara sebagai berikut:

     1) Cari nilai **BATAS** tertinggi pembagi = integer dari $\sqrt{N}$ (akar kuadrat N)

     2) Kemudian periksa apakah **N** habis dibagi dengan **X** dimana **X** adalah salah satu bilangan ganjil mulai dari 3, 5, 7, 9, 11, 13, 15, ...

     Proses ini dilakukan secara berulang-ulang selama **(X <= BATAS)**

     - Bila **N** habis dibagi X, maka N adalah **"BUKAN PRIMA"** dan proses selesai.

     - Bila **X > BATAS** berarti nilai **N** tidak habis dibagi dengan 3, 5, 7, ….dan seterusnya sampai **BATAS** yang berarti **N** adalah **"PRIMA"** dan proses selesai.

Contoh : untuk N bernilai ganjil.

1) Misal N = **97**, maka nilai batas = integer $\sqrt{97}$ = **9**
jadi bilangan yang disusun sebagai pembagi adalah : **3, 5, 7, 9**
tidak perlu : 2, 3, 4, 5, ,6 96
seperti cara pertama,
juga tidak perlu : 3, 5, 7, 9, 11, 13, ... 93, 95
semua bilangan ganjil dibawah 97.

Karena 97, tidak habis dibagi dengan salah satu diantara 3, 5, 7, dan 9
maka 97 adalah bilangan **prima.**

2) Misal N = **91**, maka nilai batas = integer $\sqrt{91}$ = 9
jadi bilangan yang disusun sebagai pembagi adalah : **3, 5, 7, 9**

Ternyata 91 habis dibagi dengan 7, maka 91 **bukan prima**

99

Cara-3

```
#include <stdio.h>
#include <math.h>
void main()
{    int N, X, Batas, Flag;
     cin >> N;
     if(N == 2) //satu-satunya PRIMA yang bukan ganjil
          cout << "PRIMA";
     else
        { if ( N%2 == 0 )//N genap
             cout << "BUKAN PRIMA";
          else
             { X = 3;//Prima terkecil selain 2
               Batas = sqrt(N);
               Flag = 0;
               while( Flag == 0 && X <= Batas)
                    { if ( N % X == 0 ) Flag = 1; //N habis dibagi X
                      X = X + 2; //menyiapkan bilangan ganjil berikutnya
                    }
               if(Flag == 0)
                    cout << "PRIMA";
               else
                    cout << "BUKAN PRIMA";
             }
        }
}
```

include **<math.h>**
Karena menggunakan
Fungsi Pustaka
sqrt()

Batas = √ N

Prima

Nilai X
3
5
7
9
11
dst

Program inipun masih **kurang efisien**,
karena masih menyiapkan **bilangan ganjil** dari 3 - - - √N.
Seharusnya menyiapkan **bilangan prima** dari 3 - - - - - <= √N

Contoh :

Bila nilai N berkisar antara : **2705** sampai dengan **2807**, maka √N = 52.
Jadi perlu disiapkan sebagai pembagi adalah :

3 5 7 9 11 13 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51

Sebenarnya cukup disiapkan **bilangan prima <=51,**

yaitu : **3 5 7 11 13 17 19 23 29 31 37 41 43 47**

Tapi hal ini cukup berarti atau cukup efisien bila nilai yang diperiksa besar sekali. Kalau nilainya cukup kecil, maka program diatas cukup efisien, karena menyiapkan bilangan prima lebih kecil atau sama dengan √N memerlukan proses yang cukup rumit, dan waktu yang cukup banyak, bahkan bisa kurang efisien bila dibandingkan dengan program diatas.

Sebagai tempat untuk menyimpan bilangan prima < = √N diperlukan array satu dimensi

100

Cara-3 sebelumnya :

Input : N misal = 2000 , → √ N = 44

Siapkan nilai X : **3** dan **5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43**

Periksa : Bila N habis dibagi dengan X (N%X == 0) berarti N bukan PRIMA

Meny iapkan 20 nilai, 6 diantarany a BUKAN PRIMA

Cara-4. Siapkan bilangan : **5 7 11 13 17 19 23 25 29 31 35 37 41 43**

Hany a meny iapkan 14 buah nilai, 2 diantarany a BUKAN PRIMA

Bilangan **I*6-1** dan **I*6+1**

Dimana **I** = 1, 2 ,3 4, dan seterusny a

Terlihat nanti ada beberapa nilai y ang bukan prima, tapi jumlah nilai sepertiga lebih sedikit dibandingkan dengan jumlah nilai-nilai pada Cara-3

Lebih lengkap sampai dengan **I = 30**

X = I*6-1
Y = I*6+1

Cara-4 **Menghemat 1/3 jumlah nilai pembagi**

```
#include <iostream.h>
#include <math.h>
void main()
{ int N, U, X, U,Y, Batas, Flag;
  cin >> N;
  if(N == 2 || N == 3) // 2 dan 3 diperlakukan tersendiri
       cout << "PRIMA";
  else
    { if ( N%2 == 0 )//N genap
         cout << "BUKAN PRIMA";
      else
        { U=1; X = 5; Y=7;//X=6*U-1  dan  Y=6*U+1
          Batas = sqrt(N);
          Flag = 0;
          while( Flag == 0 && X <= Batas)
             { if ( N % X == 0 ) Flag = 1; //N habis dibagi X
               if ( N % Y == 0 ) Flag = 1; //N habis dibagi Y
               U++; X=U*6-1; Y=U*6+1;
             }
           if(Flag == 0)
               cout << "PRIMA";
           else
               cout "BUKAN PRIMA";
        }
    }
}
```

include **<math.h>** Karena menggunakan Fungsi Pustaka **sqrt()**

Batas = √ N

X	Y
5	7
11	13
17	19
23	25
29	31
35	37
41	43
47	49
53	55
59	61
65	67
71	73
77	79
83	85
89	91
95	97
101	103
107	109
113	115
119	121
125	127
131	133
137	139
143	145
149	151
155	157
161	163
167	169
173	175
179	181

Terlihat, diantara 60 nilai ada 20 nilai y ang BUKAN PRIMA ( )

101

Soal-44

Susun program untuk mencetak bilangan-bilangan prima yang nilainya dibawah atau lebih kecil dari 100.

Bilangan prima yang lebih kecil dari 100 adalah :

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 47, 53, 61, 67, 71, 73, 79, 83, 89, 97

```
#include <iostream.h>
#include <math.h>
void main()
{ int I, X, Flag;
  cout << 2 <<" ";
  X = 3;
  while( X < 100)
  {
   I = 3;
   Flag = 0;
   while(I <= sqrt(X) && Flag == 0)
      {
        if( X %  I == 0 )
            Flag = 1;// bukan PRIMA
        I = I + 2;
      }

   if(Flag == 0)
        cout << X << " ";
   X = X + 2;
  }
}
```

2 adalah prima langsung dicetak

Bilangan prima berikutnya adalah sebagian dari bilangan ganjil dimulai dari 3

Memeriksa apakah **X** prima, Bila **X** Mod `I` == 0, berarti X bukan bilangan prima. `I` adalah bilangan ganjil dimulai dari 3 sampai dengan sqrt(X)

Setelah keluar dari loop, bila nilai Flag masih = 0, berarti X adalah prima, dan dicetak.

X adalah bilangan ganjil, mulai dari 3 dan selalu naik 2. Sebagian dari bilangan ganjil tersebut, ada yang merupakan bilangan prima

X	sqrt (X)	I	while (I <=sqrt(X))	Cetak
3	1	3	3 > 1 → keluar	3
5	2	3	3 > 2 → keluar	5
7	2	3	3 > 2 → keluar	7
9	3	3	X % I == 0 Bukan prima	
11	3	3 5	5 > 3 → keluar	11
13	3	3 5	5 > 3 → keluar	13
15	3	3	X % I == 0 Bukan prima	
17	4	3 5	5 > 4 → keluar	17

`I` = bilangan ganjil mulai dari 3. Untuk X = 100 nilai `I` hanya sampai 9 : → 3, 5, 7, 9

`Sqrt(100) = 10.`

Bilangan ganjil dibawah 10 = 9

SOAL **Ubah program diatas untuk nilai X sampai dengan 1000, Dengan menggunakan nilai `I*6-1` dan `I*6+1` Sebagai pengganti bilangan ganjil 5,7,9,11,. . .31**

Untuk nilai 1000, akan disiapkan 3 dan

5, 7, 11, 13, 17, 19, 23, 25, 29, 31

= Bukan prima

5= 6*1 - 1
7= 6*1 + 1

102

Soal-45 Algoritma Euclides (Euclidean Algorithm)

faktor persekutuan terbesar

dua buah bilangan bulat (m dan n)

faktor persekutuan terbesar

Catatan : Faktor Persekutuan Terbesar adalah pembagi persekutuan terbesar

Diketahui ada 2 buah bilangan bulat m dan n.
m = 30 dan n = 105

Cari : **faktor Persekutuan Terbesar** kedua buah bilangan tersebut :

Faktor Persekutuan terbesar

Jawab :
Pembagi habis nilai 30 adalah : 1 2 3 5 6 12 15 30
Pembagi habis nilai 105 adalah : 1 3 5 7 15 21 35 105

Dari hasil diatas terlihat bahw a 30 dan 105, mempunyai pembagi habis yang sama yaitu : `1, 3, 5, 15.`
Diantara nilai pembagi yang sama ini, yang terbesar adalah : `15`
Jadi Faktor Persekutuan Terbesar adalah : `15`

Soal : Susun algoritma untuk menginput dua buah bilangan integer (yang berbeda). Kemudian dengan menggunakan **Euclidean Algorithm**, cetaklah Faktor Persekutuan Terbesar kedua bilangan yang diinput tadi.

```
//Euclidean
#include<iostream.h>
void main()
{
  int m,n,x, sisa;
  cin >> m >> n;
  if(m < n)
      { x = m; m = n; n = x; }
  while(n != 0)
   { sisa = m % n;
     m = n;
     n = sisa;
   }
  cout <<  m;
}
```

```
Untuk : m = 105, dan
        n = 30
Selama (n != 0)
Proses :
```

m	n	sisa
105	30	15
30	15	0
15	0	selesai

Catatan : Selesai bila n == 0

```
Terakhir nilai m = 15
```
Faktor Persekutuan Terbesar = 15

Richard Johnsonbaugh,
"Discrete Mathematics",
Prentice Hall, Inc. 1998
Alih bahasa : Drs Didiek Djunaedi

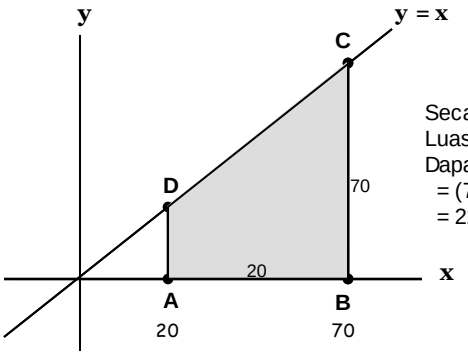
Catatan : **Program diatas menggunakan algoritma yang didukung oleh pengetahuan matematika. Sebenarnya secara algoritma yang menggunakan 'akal-akalan' persoalan diatas dapat juga diselesaikan dengan teknik yang lebih mudah dipahami yaitu dengan menyimpan angka-angka pembagi habis, masing-masing kedalam array satu dimensi, kemudian membandingka isi kedua buah array dan mencari nilai terbesar yang sama.**
Karena array belum dipelajari, maka contoh ini akan dibahas kemudian

103

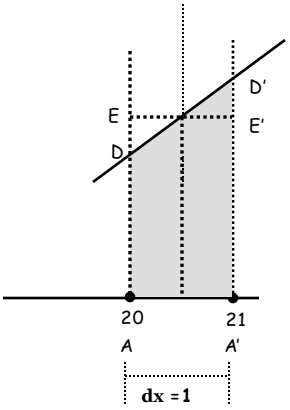
Soal-46

Menghitung luas bidang yang dibatasi oleh garis

Diberikan sebuah trapesium ABCD dengan gambar sebagai berikut.



Secara matematik,
Luas trafesium ABCD,
Dapat dihitung
= (70 + 20) * 50 / 2
= 2250



Trapesium ini dibentuk oleh empat buah garis :
y = 0,
y = x,
x = 20, dan
x = 70

Soal-46a:

Susun algoritma untuk menghitung luas trapesium ABCD diatas, dengan cara membagi trapesium tersebut dibagi menjadi 50 bagian-bagian kecil sehingga dx = 1 (baca delta x = 1)
Perhatikan luas satu bagian seperti gambar kedua diatas.

Dengan Integral

$$\text{Luas} = \int_{20}^{70} x\,dx$$

$$= \frac{1}{2}x^2 \Big]_{20}^{70}$$

= 0.5*4900 – 0.5*400
= 0.5 * 4500
= 2250

Pandanglah satu bagian kecil diatas :
Luas A A' D' D = luas A A' E' E
bila garis tingginya diambil tepat diantara 20 dan 21 yaitu di 20.5
karena persamaan garis y = x,
maka tingginya = 20.5
sehingga luas A A' E' E
= dx kali y
= 1 x 20.5
= 20.5

```
#include<iostream.h>
void main()
{ double Awal, Akhir, dx, TotalLuas;
  double X, Y, Luasdx;
  Awal = 20;
  Akhir = 70;
  dx =  (Akhir - Awal) / 50;
  TotalLuas = 0;
  while(Awal < Akhir)
    { X = Awal + (0.5 * dx);
      Y = X;
      Luasdx  =  Y * dx;
      TotalLuas = TotalLuas + Luasdx;
      Awal = Awal + dx;
    }
  cout << TotalLuas;
}
```

Untuk garis Y yang bukan linear, semakin kecil **dx** semakin teliti penghitungan luas

Untuk pertama kali,
X = 20 + 0.5 * dx
= 20 + 0.5 * 1
= 20.5

Sesuai dengan persamaan garis Y = X
Soal lain mungkin saja Y = X + 2,
atau $Y = X^2 + 3X + 5$, dan sebagainya

Kalau program ini dijalankan maka akan tercetak : 2250
Sama dengan perhitungan secara matematik

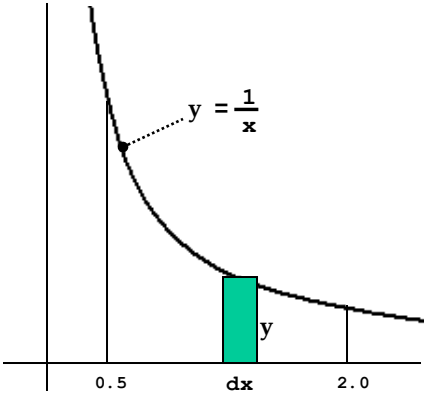
104

Soal-46b:

Susun algoritma untuk

Menghitung luas area yang dibatasi oleh :

$y = \frac{1}{x}$, $y = 0$, $x = 0.5$, dan $x = 2.0$



Penyelesaian secara matematik

Menggunakan integral

$$\text{Luas} = \int_{0.5}^{2.0} \frac{1}{x}\,dx$$

$$= \ln x \Big]_{0.5}^{2.0}$$

$$= \ln 2.0 - \ln 0.5$$

$$= 0.693147181 - -0.69314718$$

$$= 1.386294361$$

Benarkah ?

$$\int \frac{1}{x}\,dx = \ln x$$

Catatan :
Matematik mengatakan

$$\int \frac{1}{x}\,dx = \ln x + C$$

Untuk persoalan tertentu, seperti contoh ini C = 0

Penyelesaian menggunakan program komputer

```
#include<iostream.h>
#include<math.h>
void main()
{ double y, x,dx, Luas, dLuas;
  dx = 0.00001;
  Luas = 0.0;
  for(x=0.5000005; x<2.0; x=x+dx)
    { y = 1/x;
      dLuas = y * dx;
      Luas = Luas + dLuas;
    }
  cout << Luas;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAM
1.3863
```

Dengan program didapat Luas area dari x = 0.5 sampai x=2.0 sebagai berikut :

Bila dx	Dengan nilai awal X =	tercetak Luas =
0.1	0.55	1.384743246
0.01	0.505	1.386278737
0.001	0.5005	1.386294205
0.0001	0.50005	1.386294360
0.00001	0.500005	1.386294361

Makin kecil dx, Makin teliti hasil perhitungan

Sama untuk dx yang kecil sekali

Jadi terbukti benar bahwa :

$$\int \frac{1}{x}\,dx = \ln x$$

Catatan : Bila menggunakan tipe data float, hasilnya kurang teliti.

Catatan :
Matematik mengatakan

$$\int \frac{1}{x}\,dx = \ln x + C$$

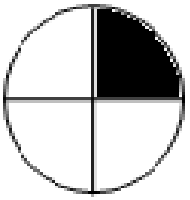
Untuk persoalan tertentu, seperti contoh ini, C = 0
Karena pada saat awal perhitungan Luas = 0

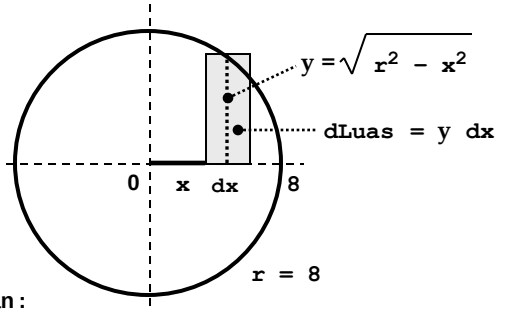
105

Soal-46c:

Hitung Luas Lingkaran Bila jari-jari = 8 cm

Yang dihitung cukup seperempatpat lingkaran, misal bagian yang berada di kuadran ke- I





Rumus yang digunakan :

Dihitung dengan matematik

$$\int \sqrt{a^2-u^2}\,du = \frac{1}{2}\,u\,\sqrt{a^2-u^2} + \frac{1}{2}\,a^2\,\arcsin\frac{u}{a} + C$$

$$\text{Luas } \frac{1}{4} \text{ lingkaran} = \int_0^8 y\,dx$$

$$= \int \sqrt{r^2 - x^2}\,dx$$

$$= \frac{1}{2}\,x\,\sqrt{r^2-x^2} + \frac{1}{2}r^2\,\arcsin\frac{x}{r}\Bigg]_0^8$$

$$= \frac{1}{2}\,8\,\sqrt{8^2-8^2} + \frac{1}{2}\,8^2\,\arcsin\frac{8}{8}$$

$$= \left(\frac{1}{2}\,8\,\sqrt{8^2-8^2}\right) + \left(\frac{1}{2}\,8^2\,\arcsin\frac{8}{8}\right)$$

$$= 0$$

$$= 32\,\arcsin 1$$

$$= 32\,\frac{\pi}{2}$$

$$= 16\,\pi$$

$$= 16 * 3.1416$$

$$= 50.2656$$

(ini luas seperempat lingkaran)

$$= \frac{\pi}{2}$$

sin 90° = 1
arc sin 1 = 90°
90^0 dalam radian adalah $= \frac{\pi}{2}$

Soal-46c:

Susun program dalam Bahasa C++ untuk menghitung luas seperempat lingkaran, bila diketahui jar-jari lingkaran tersebut = 8.

```
//LuasL01.cpp
#include<iostream.h>
#include<math.h>
void main()
{ double r,x,y,dx,Luas,dluas;
  r = 8.0;    dx = 0.01;
  Luas = 0.0;
  for(x=0.005; x<r; x=x+dx)
    { y = sqrt(r*r - x*x);
      dluas = y * dx;
      Luas = Luas + dluas;
    }
  cout <<  Luas;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAM
50.2657
```

Ada sedikit perbedaan karena ketelitian komputer menyimpan nilai pecahan

Jadi luas lingkaran dengan jari-jari = 8, adalah 4 * 50.2656 = 201.0624

sama dengan hasil hitungan degan rumus biasa yaitu : $\pi\,r^2$
= 3.1416 * 64 = 201.0624

106

4.2.7 Mengurutkan nilai dalam array satu dimensi

Soal-47 Susun program dalam Bahasa C++ untuk menginputkan 10 buah nilai, kemudian mencetaknya urut mulai nilai terkecil sampai dengan nilai terbesar

```
#include<iostream.h>
#define n 10
void main()
{ int  A[n];
  int  X, I, K;
  for( I = 0; I <= n-1; I++ )
     { cin >> A[I] ;
     }
  cout << "\nSebelum di sort\n";
  for ( I=0; I <= n-1; I++)
        cout << A[I] << " ";
     )
  cout << "\n";

  for( K=0; K <= n-2; K++ )
     { for(I=0; I <= n-1-K ; I++ )
          { if A[I] > A[I+1]
              { X = A[I];
                A[I] = A[I+1]
                A[I+1] = X;
              }
              I++;
          }
     }

  cout << "\nSesudah Sort \n");
  for(I=0; I <= n-1; I++)
      cout <<  A[I] << " ";
}
```

Ada banyak metode pengurutan (sorting) untuk data yang berada dalam array vsatu dimensi, antara lain:

1. Bubble Sort.
2. Selection Sort
3. Insertion Sort.
4. Shell Sort.
5. Merge Sort
6. Radix Sort
7. Quick Sort.
8. Heap Sort

Yang dicontohkan disini adalah pengurutan dengan menggunakan metode BUBBLE SORT

Bila diinput : 15 10 7 22 17 5 12 25 19 10
Maka Tercetak : 5 7 10 10 12 15 17 19 22 25

107

selesai

Sessi-4

108