

PERTEMUAN 5

Pengulangan

Hermanto, S.Kom. M.Kom.

Pola yang berulang

Salah satu kelebihan komputer adalah kemampuannya untuk mengerjakan pekerjaan yang sama berulang kali tanpa mengenal lelah.



Pengulangan merupakan kegiatan mengerjakan sebuah atau sejumlah aksi yang sama sebanyak jumlah yang ditentukan atau sesuai dengan kondisi yang diinginkan.

Pola yang berulang

1. Sebagai contoh : 1,2,3,4,5,6,....


Kita dapat melihat pola angka yang muncul adalah angka sebelumnya ditambah 1. Dengan kata lain, jika elemen sekarang adalah i , maka elemen selanjutnya adalah $i + 1$ atau dapat di tulis dalam notasi algoritma: $i \leftarrow i + 1$

2. Perhatikan deret integer berikut : 2+4+6+8+10,

Seperti halnya contoh pertama, deret ini merupakan penjumlahan elemen-elemen bilangan genap. Elemen-elemen di dalam deret diperoleh dengan menambahkan elemen sebelumnya dengan 2. Dengan kata lain, jika elemen sekarang adalah k , maka elemen selanjutnya adalah $k + 2$ atau dapat ditulis dalam notasi algoritma : $k \leftarrow k + 2$

Struktur Pengulangan

Secara umum, struktur pengulangan di dalam program terdiri atas dua bagian penting :

1. **kondisi pengulangan**, yaitu ekspresi bernilai boolean yang harus dipenuhi untuk melaksanakan pengulangan. Kondisi (syarat) ini ada yang dinyatakan secara eksplisit oleh pemrogram atau dikelola sendiri oleh komputer (implisit) ;
 2. **isi atau badan pengulangan**, yaitu sekumpulan aksi yang diulang selama kondisi pengulangan dipenuhi.
- 

Struktur Pengulangan

Di samping itu, struktur pengulangan biasanya disertai dengan bagian :

1. **inisialisasi**, yaitu aksi yang dilakukan sebelum pengulangan dilakukan pertama kali;
2. **terminasi**, yaitu aksi yang dilakukan setelah pengulangan selesai dilaksanakan.

Inisialisasi dan terminasi tidak selalu harus ada, namun pada banyak kasus inisialisasi umumnya diperlukan.

Jadi, struktur pengulangan secara umum adalah sebagai berikut :

<inisialisasi>

awal pengulangan

badan pengulangan

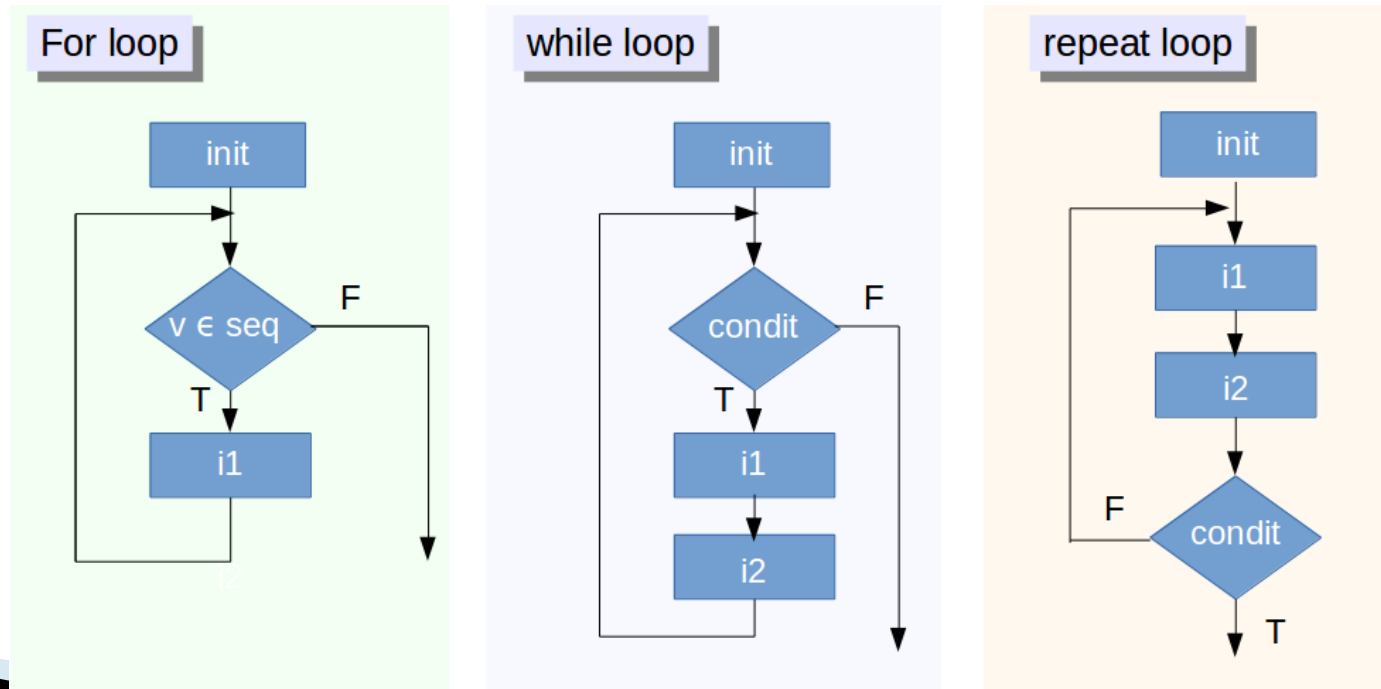
akhir pengulangan

<terminasi>

Struktur Pengulangan

Di dalam algoritma terdapat beberapa notasi konstruksi pengulangan yang dapat digunakan, diantara lain :

1. Konstruksi *FOR*;
2. Konstruksi *WHILE*;
3. Konstruksi *REPEAT*.



Konstruksi *FOR*

Notasi FOR adalah konstruksi pengulangan tanpa kondisi (unconditional looping), artinya instruksi-instruksi di dalam badan pengulangan diulangi sejumlah kali (yang dispesifikasikan oleh pemrogram). Dalam hal ini, jumlah pengulangan sudah diketahui sebelum konstruksi pengulangan eksekusi.

Bentuk umum konstruksi FOR ada dua macam : menaik (*ascending*) atau menurun (*descending*) :

FOR menaik :

```
for pencacah ←<nilai_awal> to <nilai_akhir> do  
    aksi  
end for
```

Konstruksi *FOR*

Contoh :

Misalkan anda diminta membuat program untuk mencetak string “Teknologi Bisnis Digital” sebanyak 10 kali. Intruksi tanpa for :

```
cout('Teknologi Bisnis Digital')
```

```
.
```

```
.
```

```
cout('Teknologi Bisnis Digital')
```

Intruksi FOR

```
for i ← 1 to 10 do
```

```
    cout('Teknologi Bisnis Digital')
```

```
end for
```



Konstruksi *FOR*

```
for i ← 1 to N do  
    cout('Teknologi Bisnis Digital')  
end for
```

PROGRAM CetakBanyakKalimat

{Mencetak kalimat 'Teknologi Bisnis Digital' sebanyak N kali}

DEKLARASI

i : **integer** {pencacah pengulangan}

N : **integer** {jumlah pengulangan}

ALGORITMA

cin(N)

for i ← 1 to N do

cout('Teknologi Bisnis Digital')

end for

Konstruksi *FOR*

Contoh :

Misalkan kita ingin menghitung jumlah angka-angka dari deret 1 sampai N.

PROGRAM PenjumlahanDeret

{Menjumlahkan deret yang merupakan bilangan bulat positif}

DEKLARASI

i : integer {pencacah pengulangan}

N : integer {jumlah pengulangan}

sum : integer {pencatat jumlah data}

ALGORITMA

cin(N)

sum \leftarrow 0 {inisialisasi jumlah deret dengan 0}

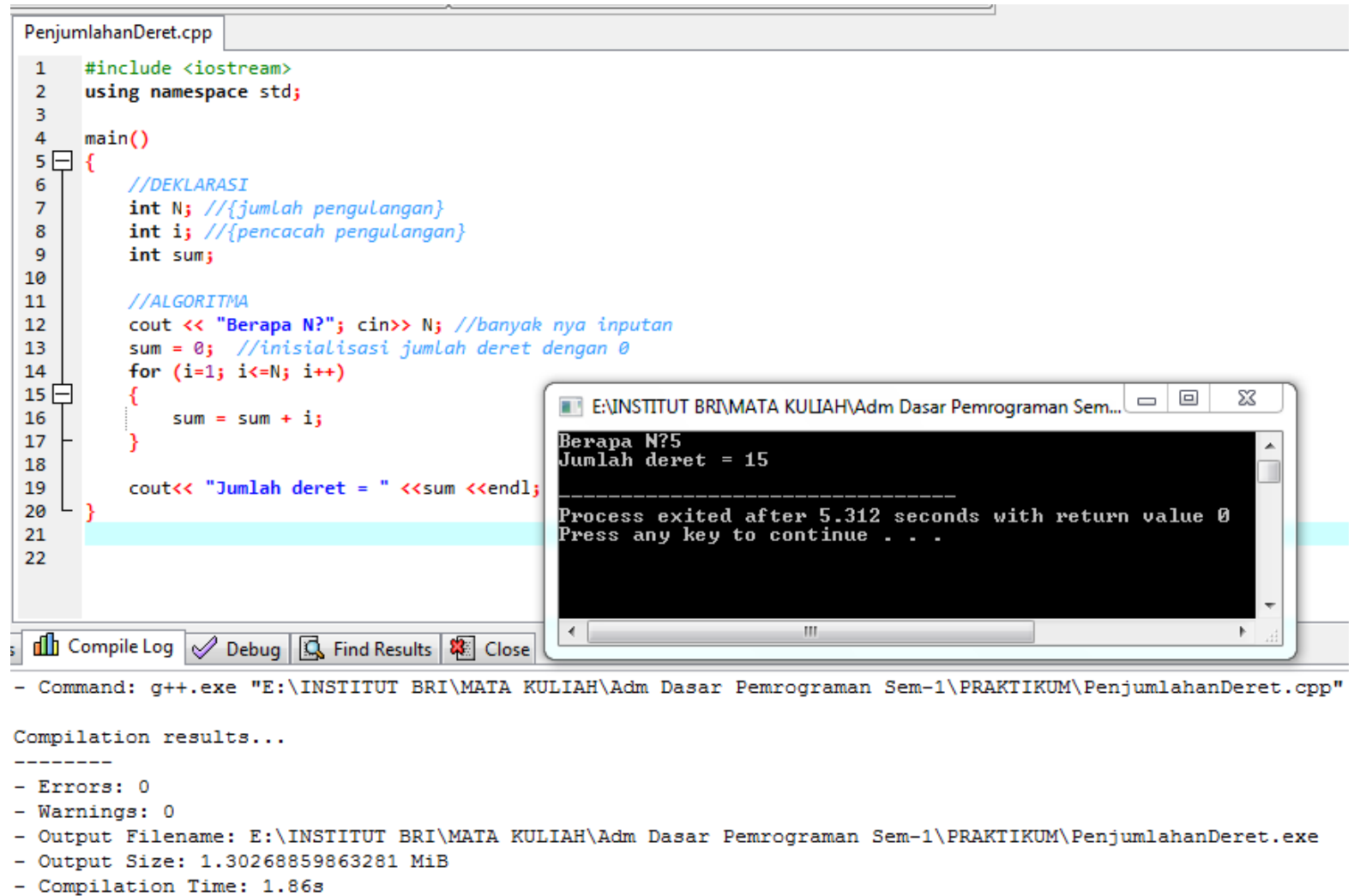
for i \leftarrow 1 to N do {ulangi penjumlahan elemen deret sebanyak N kali}

 sum \leftarrow sum + i

end for

cout(sum)

Konstruksi *FOR*



```
PenjumlahanDeret.cpp
1  #include <iostream>
2  using namespace std;
3
4  main()
5  {
6      //DEKLARASI
7      int N; //{jumlah pengulangan}
8      int i; //{pencacah pengulangan}
9      int sum;
10
11     //ALGORITMA
12     cout << "Berapa N?"; cin>> N; //{banyak nya inputan}
13     sum = 0; //{inisialisasi jumlah deret dengan 0}
14     for (i=1; i<=N; i++)
15     {
16         sum = sum + i;
17     }
18
19     cout<< "Jumlah deret = " <<sum <<endl;
20 }
21
22
```

Command: g++.exe "E:\INSTITUT BRI\MATA KULIAH\Adm Dasar Pemrograman Sem-1\PRAKTIKUM\PenjumlahanDeret.cpp"

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\INSTITUT BRI\MATA KULIAH\Adm Dasar Pemrograman Sem-1\PRAKTIKUM\PenjumlahanDeret.exe
- Output Size: 1.30268859863281 MiB
- Compilation Time: 1.86s

Konstruksi *FOR*

FOR menurun :

```
for pencacah ← <nilai_akhir>  
downto <nilai_awal> do  
    aksi  
end for
```

Contoh :

Sebuah roket diluncurkan dengan hitungan mundur (count down), dimulai dari 100, 99, 98, ..., 1, 0



Konstruksi *FOR*

PROGRAM *PeluncuranRoket*

{Hitung mundur peluncuran roket}

DEKLARASI

i : integer {pencacah pengulangan}

N : integer {jumlah pengulangan}

ALGORITMA

cin(N)

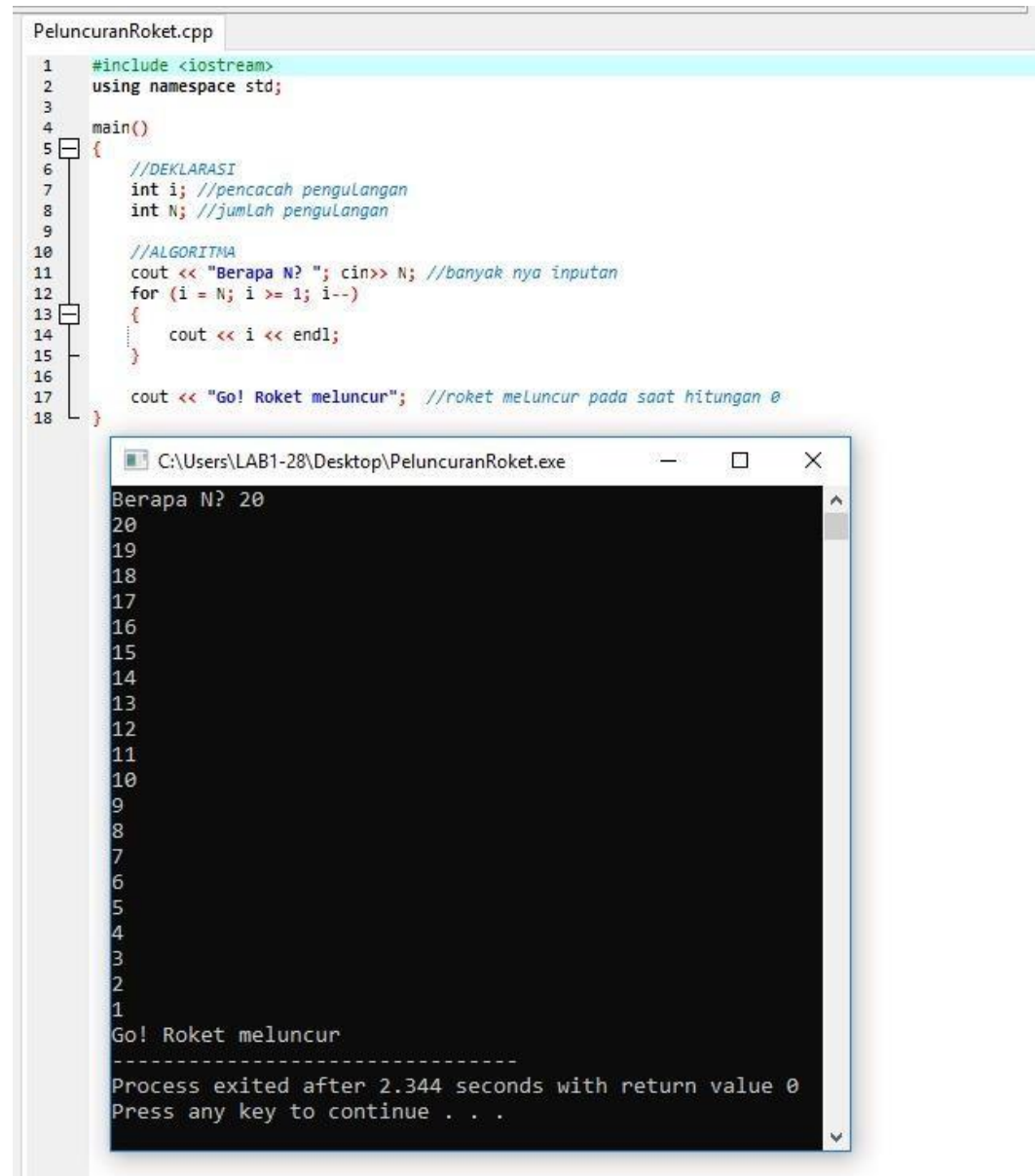
for i ← N downto 0 do {ulangi deret sebanyak N kali}

cout(i)

end for

cout('Go!') {roket meluncur pada saat hitungan 0}

Konstruksi *FOR*



The image shows a C++ program named `PeluncuranRoket.cpp` and its execution output. The program uses a `for` loop to count down from a user input `N` to 1, printing each number. After the loop, it prints "Go! Roket meluncur". The execution window shows the user inputting 20, and the program printing the numbers 20 down to 1, followed by the rocket launch message.

```
PeluncuranRoket.cpp
1  #include <iostream>
2  using namespace std;
3
4  main()
5  {
6      //DEKLARASI
7      int i; //pencacah pengulangan
8      int N; //jumlah pengulangan
9
10     //ALGORITMA
11     cout << "Berapa N? "; cin >> N; //banyak nya inputan
12     for (i = N; i >= 1; i--)
13     {
14         cout << i << endl;
15     }
16
17     cout << "Go! Roket meluncur"; //roket meluncur pada saat hitungan 0
18 }
```

```
C:\Users\LAB1-28\Desktop\PeluncuranRoket.exe
Berapa N? 20
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
Go! Roket meluncur
-----
Process exited after 2.344 seconds with return value 0
Press any key to continue . . .
```

Konstruksi *WHILE*

Bentuk umum konstruksi WHILE adalah :

```
while kondisi do  
    aksi  
end while
```

Aksi dikerjakan berulang kali selama kondisi masih benar (true). Jika kondisi salah (false), maka badan pengulangan tidak akan dimasuki, yang berarti pengulangan selesai.

Contoh:

Mencetak pesan 'Teknologi Bisnis Digital' sebanyak N kali.



Konstruksi *WHILE*

PROGRAM CetakBanyakKalimat

{ Mencetak 'Teknologi Bisnis Digital' sebanyak N kali }

DEKLARASI

i : integer {pencacah pengulangan}

N : integer {jumlah pengulangan}

ALGORITMA

cin(N)

i ← 1 {inisialisasi pencacah pengulangan dengan 1}

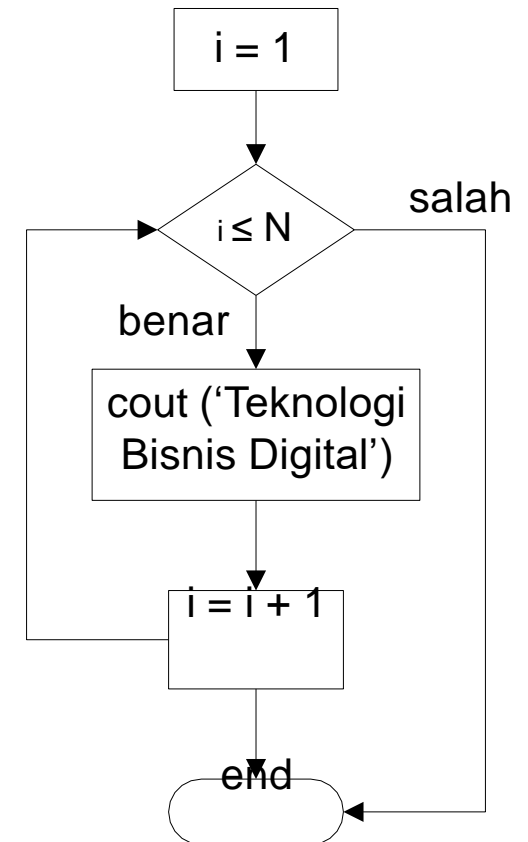
while i ≤ N **do**

cout('Teknologi Bisnis Digital')

i ← i + 1 {naikkan cacah pengulangan}

end while

{i > N → loop invariant}



Konstruksi *WHILE*

Misalkan kita ingin menghitung jumlah angka-angka dari deret 1 sampai N.

PROGRAM *Penjumlahan Deret*

{ Menjumlahkan deret dengan N adalah bilangan bulat positif }

DEKLARASI

i : integer { pencacah pengulangan }

N : integer { jumlah pengulangan }

sum : integer

ALGORITMA

cin(N)

sum \leftarrow 0 { inisialisasi jumlah deret dengan 0 }

i \leftarrow 1

while i \leq N **do** { ulangi penjumlahan elemen deret sebanyak N kali }

 sum \leftarrow sum + i

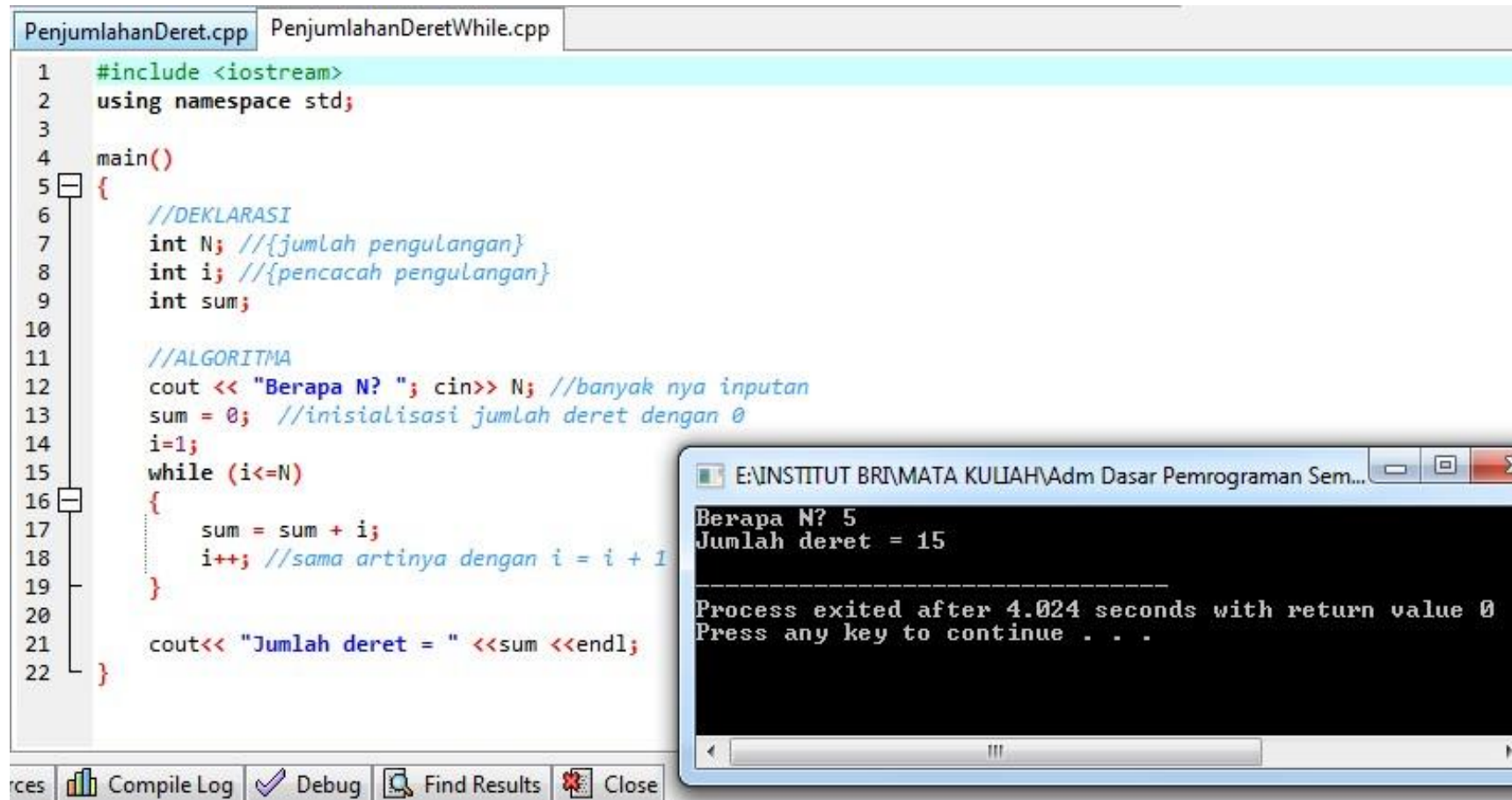
i \leftarrow i + 1 { naikan cacah pengulangan }

end while

{ i > N }

cout(sum)

Konstruksi *WHILE*



The image shows a C++ IDE with two tabs: 'PenjumlahanDeret.cpp' and 'PenjumlahanDeretWhile.cpp'. The active tab is 'PenjumlahanDeretWhile.cpp', which contains the following code:

```
1  #include <iostream>
2  using namespace std;
3
4  main()
5  {
6      //DEKLARASI
7      int N; //{jumlah pengulangan}
8      int i; //{pencacah pengulangan}
9      int sum;
10
11     //ALGORITMA
12     cout << "Berapa N? "; cin >> N; //banyak nya inputan
13     sum = 0; //inisialisasi jumlah deret dengan 0
14     i = 1;
15     while (i <= N)
16     {
17         sum = sum + i;
18         i++; //sama artinya dengan i = i + 1
19     }
20
21     cout << "Jumlah deret = " << sum << endl;
22 }
```

Below the code editor, there is a console window titled 'E:\INSTITUT BRI\MATA KULIAH\Adm Dasar Pemrograman Sem...'. It displays the output of the program:

```
Berapa N? 5
Jumlah deret = 15

-----
Process exited after 4.024 seconds with return value 0
Press any key to continue . . .
```

The IDE's status bar at the bottom includes buttons for 'Compile Log', 'Debug', 'Find Results', and 'Close'.

Konstruksi *WHILE*

Algoritma peluncuran roket
dengan hitungan mundur mulai
dari 100, 99, 98, ..., 1, 0

PROGRAM PeluncuranRoket

{ Hitung mundur peluncuran roket}

DEKLARASI

i : **integer** {pencacah pengulangan}

N : **integer** {jumlah pengulangan}

ALGORITMA

cin(N)

i \leftarrow N

while i \geq 0 **do**

cout(i)


i \leftarrow i - 1 {turunkan cacah pengulangan}

end while

{i < 0}

cout("Go!") {roket meluncur pada saat hitungan 0}

Kapan Sebaiknya Menggunakan *WHILE*?

1. Sekilas antara FOR dan WHILE sama saja kegunaannya. Namun, WHILE memiliki keunggulan yang tidak dimiliki oleh FOR.
 2. Pada kasus-kasus di mana jumlah pengulangan diketahui di awal program, WHILE dapat digunakan sebaik penggunaan FOR. Namun, untuk proses yang jumlah pengulangannya tidak dapat ditentukan di awal, hanya struktur WHILE yang dapat kita gunakan, sebab kondisi pengulangan diperiksa di awal pengulangan.
 3. Jadi, meskipun kita tidak mengetahui kapan persisnya WHILE ini berhenti, tetapi kita menjamin bahwa kondisi bernilai salah, maka pengulangan pasti berhenti.
- 

Konstruksi *REPEAT*

- ▶ Bentuk umum konstruksi REPEAT adalah : **repeat**
 - ▶ aksi
 - ▶ **until** kondisi
-
1. Konstruksi REPEAT mendasarkan pengulangan pada kondisi yang bernilai Boolean. Pemeriksaan kondisi dilakukan pada **akhir setiap pengulangan**.
 2. Aksi dikerjakan berulang-ulang sampai kondisi terpenuhi (bernilai true). Dengan kata lain, jika kondisi masih false, proses pengulangan masih terus dilakukan.
 3. Konstruksi REPEAT memiliki makna yang serupa dengan WHILE, dan dalam beberapa masalah kedua konstruksi tersebut komplement satu sama lain

Konstruksi *REPEAT*

PROGRAM *CetakBanyakKalimat*

{ Mencetak 'Teknologi Bisnis Digital' sebanyak N kali }

DEKLARASI

i : integer {pencacah pengulangan}

N : integer {jumlah pengulangan}

ALGORITMA

cin(N)

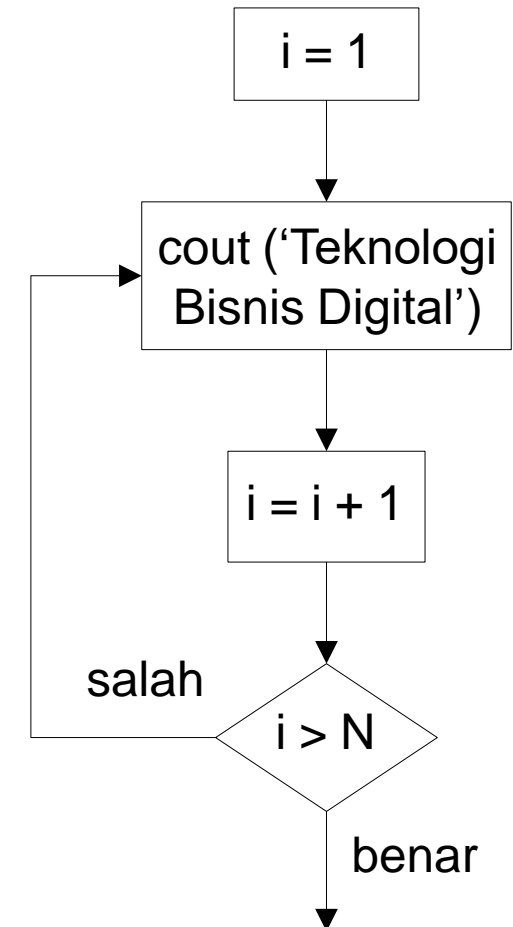
i ← 1 {inisialisasi pencacah pengulangan dengan 1}

repeat

cout('Teknologi Bisnis Digital')

i ← i + 1 {naikkan cacah pengulangan}

until i > N



Konstruksi *REPEAT*

Misalkan kita ingin menghitung jumlah angka-angka dari deret 1 sampai N.

PROGRAM *Penjumlahan Deret*

{ Menjumlahkan deret dengan N adalah bilangan bulat positif }

DEKLARASI

i : integer {pencacah pengulangan}

N : integer {jumlah pengulangan}

sum : integer

ALGORITMA

cin(N)

sum \leftarrow 0 {inisialisasi jumlah deret dengan 0}

i \leftarrow 1

repeat

 sum \leftarrow sum + *i*

i \leftarrow *i* + 1 {naikkan cacah pengulangan}

until *i* > N

cout(sum)

Konstruksi *REPEAT*

Algoritma peluncuran roket dengan hitungan mundur mulai dari 100, 99, 98, ..., 1, 0

PROGRAM *PeluncuranRoket*

{ Hitung mundur peluncuran roket}

DEKLARASI

i : **integer** {pencacah pengulangan}

N : **integer** {jumlah pengulangan}

ALGORITMA

cin(N)

i ← N

repeat

cout(i)

i ← i - 1 {turunkan cacah pengulangan}

until i < 0

cout('Go!') {roket meluncur pada saat hitungan 0}

WHILE atau REPEAT ?

Perbedaannya adalah :

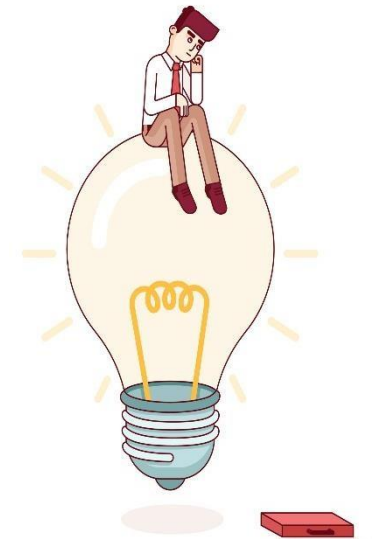
1. Pada konstruksi **REPEAT**, kondisi pengulangan diperiksa pada **akhir pengulangan**. Jadi, instruksi di dalam badan pengulangan dilaksanakan dulu, barulah pengecekan kondisi dilakukan. Konsekuensinya, badan pengulangan dilaksanakan paling sedikit satu kali.
2. Sebaliknya, Pada kondisi **WHILE**, kondisi pengulangan diperiksa pada **awal pengulangan**. Jadi, instruksi di dalam badan pengulangan hanya dapat dilaksanakan bila pengecekan kondisi menghasilkan nilai true. Konsekuensinya, badan pengulangan mungkin tidak akan pernah dilaksanakan bila kondisi pengulangan pertama kali bernilai false



WHILE atau *REPEAT* ?

Berdasarkan perbedaan yang telah di bahas sebelumnya, maka kita dapat menarik kesimpulan kapan menggunakan *WHILE* dan kapan menggunakan *REPEAT* :

- Gunakan konstruksi *WHILE* pada kasus yang mengharuskan terlebih dahulu pemeriksaan kondisi objek sebelum objek tersebut dimanipulasi.
- Gunakan konstruksi *REPEAT* pada kasus yang terlebih dahulu memanipulasi objek, baru kemudian memeriksa kondisi objek tersebut.



WHILE atau REPEAT ?

Kita menginginkan dapat memilih menu manapun berkali-kali sampai menu yang kita pilih adalah nomor 5 (keluar program). Bagaimana hal ini dapat dilakukan? Menggunakan REPEAT atau WHILE?

PROGRAM *SimulasiMenuProgram*

{ Menampilkan menu, membaca pilihan menu,
dan menampilkan nomor menu yang dipilih }

DEKLARASI

NomorMenu : integer

ALGORITMA

repeat {cetak menu}

cout(' MENU ')

cout('1. Baca data')

cout('2. Cetak data')

cout('3. Ubah data')

cout('4. Hapus data')

cout('5. Keluar Program')

cout('Masukkan pilihan anda (1/2/3/4/5)? ')

cin(NomorMenu) {input menu}

case (NomorMenu):

1 : cout('Anda memilih menu nomor 1')

2 : cout('Anda memilih menu nomor 2')

3 : cout('Anda memilih menu nomor 3')

4 : cout('Anda memilih menu nomor 4')

5 : cout('Keluar Program')

Otherwise cout ('Nomor pilihan anda salah!')

end case

until NomorMenu = 5

REPEAT (benar)

WHILE atau REPEAT ?

PROGRAM *SimulasiMenuProgram*

{ Menampilkan menu, membaca pilihan menu,
dan menampilkan nomor menu yang dipilih }

DEKLARASI

NomorMenu : integer

ALGORITMA

```
while NomorMenu ≠ 5 do
    cout(' MENU ')
    cout('1. Baca data')
    cout('2. Cetak data')
    cout('3. Ubah data')
    cout('4. Hapus data')
    cout('5. Keluar Program')
    cout('Masukkan pilihan anda (1/2/3/4/5)? ')
```

```
cin(NomorMenu) {input menu}
```

```
case (NomorMenu):
```

```
1 : cout('Anda memilih menu nomor 1')
```

```
2 : cout('Anda memilih menu nomor 2')
```

```
3 : cout('Anda memilih menu nomor 3')
```

```
4 : cout('Anda memilih menu nomor 4')
```

```
5 : cout('Keluar Program')
```

```
otherwise cout ('Nomor pilihan anda salah!')
```

```
end case
```

```
end while
```

```
{NomorMenu = 5}
```

WHILE (salah)



Terima
kasih

Latihan Soal

1. Tulislah program untuk menampilkan semua bilangan ganjil yang kurang dari 100. Hasilnya:

1

3

5

...

99

2. Lakukan analisa pada persoalan nomor 1, apakah persoalan tersebut dapat menggunakan semua konstruksi pengulangan FOR, WHILE dan REPEAT? Jelaskan?