3. Pengulangan

Overview

Pengulangan (*Loop*) merupakan sebuah konsep yang penting dalam pemrograman. Dengan struktur pengulangan, program dapat berjalan beberapa kali sesuai inisialisasi, jumlah iterasi dan kondisi berhenti yang ditentukan. Hal ini dapat menyederhanakan program yang kompleks menjadi lebih sederhana. Dalam C disediakan berbagai perintah *Loop*, dimana setiap perintah *loop* memiliki keunikan tersendiri. Di dalam bab ini akan dipraktrekkan tentang struktur pengulangan dalam bahasa C serta contoh soal dan latihan menggunakan pengulangan.

Tujuan

1. Mengimplementasikan struktur pengulangan dalam bahasa C.
2. Menerapkan sintaks-sintaks pengulangan dalam menyelesaikan persoalan
3. Mahasiswa mampu menyelesaikan persoalan tentang pengulangan

3.1. Konsep Pengulangan

Pengulangan merupakan sebuah konsep pemrograman yang penting karena konsep ini memungkinkan pengguna menggunakan sekumpulan baris program berulang kali dengan tiga komponen yang mengendalikannya, yaitu:

- **Inisialisasi**; menentukan kondisi awal dilakukannya pengulangan.
- **Jumlah iterasi**; menunjukkan berapa kali pengulangan akan dilakukan.
- **Kondisi berhenti**; menentukan kondisi yang dapat mengakhiri pengulangan.

Ketika mengimplementasikan dalam program, ketiga komponen ini tidak selalu dapat didefinisikan dalam struktur pengulangan. Mungkin saja salah satu komponen tersebut tidak didefinisikan. Pengulangan tetap dapat berjalan, asal komponen yang tidak didefinisikan tersebut dapat diketahui secara tersirat berdasarkan komponen lain yang didefinisikan. Hal lain yang perlu diperhatikan adalah bahwa **pengulangan harus berhenti**. Jika pengulangan tidak pernah berhenti, maka logika program salah. Pengulangan akan berhenti jika jumlah iterasi yang diminta sudah tercapai atau kondisi berhenti bernilai benar. Maka, dalam setiap pengulangan, pemrogram perlu menentukan jumlah iterasi atau kondisi berhenti dan langkah pencapaian menuju kondisi berhenti tersebut.

Pada bab ini akan dijelaskan 3 struktur perulangan di dalam C, yaitu struktur perulangan While , For, dan Do While

3.2. Sintaks WHILE

Pengulangan dengan menggunakan WHILE merupakan sebuah pengulangan yang dikendalikan oleh suatu kondisi tertentu, dimana kondisi tersebut yang akan menentukan apakah perulangan itu akan terus dilaksanakan atau dihentikan. Kondisi tersebut akan dicek disetiap awal iterasi, apakah sebuah kondisi terpenuhi atau tidak. Jika kondisi terpenuhi (bernilai benar), maka iterasi akan dilanjutkan. Jika kondisi tidak terpenuhi, maka iterasi dihentikan.

Perulangan dengan WHILE dapat digunakan pada struktur perulangan yang diketahui jumlah iterasinya dan juga pada struktur perulangan yang tidak diketahui jumlah iterasinya, tetapi harus selalu terdapat kondisi berhenti.

Bentuk umum pernyataan ***while*** pada C :

```
while (kondisi)
{
    //pernyataan;
}
```

Contoh implementasi pengulangan dengan while :

1. Pengulangan untuk menampilkan kalimat “Halo apa khabar ? “ sebanyak 5 kali.

```
1  #include <stdio.h>

2  main()
3  {
4  int i; //deklarasi variabel pencacah
5  i=0;   //inisialisasi variabel pencacah
6  while(i<5) //pengecekan kondisi
7  {
8  printf("\nHalo apa khabar ?");
9  i++;  //penambahan bilangan pencacah
10 }
11 }
```

Output sintaks diatas :

```
Halo apa khabar ?
Halo apa khabar ?
Halo apa khabar ?
Halo apa khabar ?
Halo apa khabar ?
```

2. Pengulangan untuk menampilkan bilangan 1 hingga 5

```
1  #include <stdio.h>

2  main()
3  {
4  int i=1;
5  while(i<=5)
6  {
7  printf("\nBilangan ke-%d",i);
8  i++;
9  }
10 }
```

Output sintaks diatas

```
Bilangan ke-1
Bilangan ke-2
Bilangan ke-3
Bilangan ke-4
Bilangan ke-5
```

3. Pengulangan untuk meminta user memasukkan nilai

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

```
2  main()
3  {
4  int i=1;
5  int input;
6  while(i<=5)
7  {
8  printf("\nMasukkan bilangan ke-%d : ",i);
9  scanf("%d",&input);
10 printf("Bilangan yang Anda masukkan :
   %d",input);
11 i++;
12 }
13 }
```

Output dari sintaks diatas bila diinputkan : 4, 5, 6, 2, dan 9.

```
Masukkan bilangan ke-1 : 4
Bilangan yang Anda masukkan : 4
Masukkan bilangan ke-2 : 5
Bilangan yang Anda masukkan : 5
Masukkan bilangan ke-3 : 6
Bilangan yang Anda masukkan : 6
Masukkan bilangan ke-4 : 2
Bilangan yang Anda masukkan : 2
Masukkan bilangan ke-5 : 9
Bilangan yang Anda masukkan : 9
```

Contoh no 1-3 merupakan contoh bentuk pengulangan yang telah diketahui jumlah iterasi yaitu 5 kali. Jumlah iterasi yang telah diketahui akan menjadi kondisi yg digunakan untuk mengecek apakah perulangan akan dilanjutkan atau tidak.
Berikut adalah contoh dimana jumlah perulangan akan ditentukan oleh user yaitu dengan cara meminta inputan dari user.

4. Modifikasi contoh no 3 dimana user diminta untuk menentukan jumlah perulangan yang akan dilakukan

```
1  #include <stdio.h>

2  main()
3  {
4  int jml,i;
5  int input;
6  printf("Jumlah Bilangan : ");
7  scanf("%d",&jml);
8  i=1;
9  while(i<=jml)
10 {
11 printf("\nMasukkan bilangan ke-%d :
   ",i);
12 scanf("%d",&input);
13 printf("Bilangan yang Anda masukkan :
   %d",input);
14 i++;
15 }
16 }
```

Telah dikatakan di awal bahwa perulangan dengan WHILE dapat digunakan untuk struktur pengulangan yang belum diketahui jumlah iterasinya, tetapi tetap mempunyai kondisi berhenti.

Berikut adalah contoh pengulangan yang belum diketahui jumlah iterasinya, tetapi tetap mempunyai kondisi berhenti.

5. Menampilkan menu sesuai pilihan user

```
1  #include <stdio.h>

2  main()
3  {
4  int pil=1; //inisialisasi kondisi
5  while(pil!=5)
6  {
7  printf("\nMENU : ");
8  printf("\n1. Red");
9  printf("\n2. Blue");
10 printf("\n3. Yellow");
11 printf("\n4. Green");
12 printf("\n5. Exit");
```

```
13 printf("\nPilihan : ");
14 scanf("%d",&pil);
15 switch(pil)
16 {
17 case 1 : printf("Anda memilih Merah");break;
18 case 2 : printf("Anda memilih Biru");break;
19 case 3 : printf("Anda memilih Kuning");break;
20 case 4 : printf("Anda memilih Hijau");break;
21 case 5 : printf("Anda memilih Keluar ...
   bye");break;
22 default : printf("Pilihan anda salah");
23 }
24 }
25 }
```

Output dari sintaks diatas bila menu yg dipilih 2, 4, 6, dan 5

```
MENU :
1. Red
2. Blue
3. Yellow
4. Green
5. Exit
Pilihan : 2
Anda memilih Biru
MENU :
1. Red
2. Blue
3. Yellow
4. Green
5. Exit
Pilihan : 4
Anda memilih Hijau
MENU :
1. Red
2. Blue
3. Yellow
4. Green
5. Exit
Pilihan : 6
Pilihan anda salah
MENU :
```

```
1. Red
2. Blue
3. Yellow
4. Green
5. Exit
Pilihan : 5
Anda memilih Keluar ... bye
```

Contoh diatas merupakan pengulangan yang belum diketahui jumlah iterasinya tetapi mempunyai kondisi yang akan menyebabkan pengulangan dihentikan, yaitu ketika user memasukkan pilihan 5. Selama user tidak memilih angka 5, maka pengulangan akan tetap dilakukan.

6. User input angka hingga memilih keluar, angka yang telah dimasukkan akan dijumlah dan dihitung jumlah iterasi yang telah dilakukan.

```
1  #include <stdio.h>

2  main()
3  {
4  int i, input, jmlInput;
5  i=0;
6  jmlInput=0;
7  input=0; //inisialisasi kondisi
8  while (input != 99 )
9  {
10 printf("Masukkan angka (input '99' untuk
   keluar) : ");
11 scanf("%d",&input);
12 jmlInput=jmlInput+input;
13 i++;
14 }
15 printf("\nAnda telah input angka sebanyak %d
   kali",i-1);
16 printf("\nJumlah inputan anda : %d",jmlInput
   -99);
17 }
```

Output

```
Masukkan angka (input '99' untuk keluar) : 5
Masukkan angka (input '99' untuk keluar) : 6
Masukkan angka (input '99' untuk keluar) : 8
```

```
Masukkan angka (input '99' untuk keluar) : 4
Masukkan angka (input '99' untuk keluar) : 99

Anda telah input angka sebanyak 4 kali
Jumlah inputan anda : 23
```

Pada contoh ke-5, variable 'i' digunakan untuk menghitung jumlah iterasi yang telah dilakukan (iterasi memasukkan angka yg dijumlah). Variabel 'input' digunakan untuk menampung inputan dari user, dan variabel 'jmlInput' digunakan untuk menampung hasil penjumlahan tiap inputan user. Kondisi berhenti yang digunakan adalah jika user memasukkan angka 99, maka pengulangan akan dihentikan. Selama user tidak memasukkan angka 99, maka pengulangan akan diteruskan. Pada program contoh no 5, variabel 'input' diinisialisasi dengan 0, hal ini dilakukan akan pada pengecekan kondisi akan memberikan nilai true sehingga pengulangan akan dijalankan untuk menampilkan Menu.

Bentuk pengulangan yang membutuhkan agar pengulangan dilakukan tanpa adanya pengecekan kondisi di awal akan lebih cocok jika menggunakan struktur *do...while* daripada struktur *while*.

3.3 Sintaks DO…WHILE

Sintaks DO... WHILE... melakukan pengulangan serupa dengan sintaks WHILE. Penggunaan sintaks ini juga tidak harus menyebutkan jumlah pengulangan yang harus dilakukan, karena dapat digunakan untuk perulangan dengan jumlah iterasinya yang belum diketahui, tetapi harus mempunyai kondisi berhenti.

Bedanya, jika pada sintaks WHILE kondisi dievaluasi/ diuji sebelum aksi pengulangan dilakukan, sedangkan pada sintaks DO...WHILE pengujian kondisi dilakukan setelah aksi pengulangan dilakukan.

Bentuk umum pernyataan ***do...while*** pada C adalah:

```
do
{
  //pernyataan
}while(kondisi);
```

Pada struktur pengulangan dengan sintaks DO... WHILE..., aksi akan terus dilakukan hingga kondisi yang dicek di akhir pengulangan, bernilai benar. Dengan sintaks ini, pengulangan pasti dilakukan minimal satu kali, yakni pada iterasi pertama sebelum pengecekan kondisi. WHILE dengan DO WHILE seringkali memberikan hasil yang sama, tetapi ada kalanya hasilnya akan

berbeda, sehingga harus berhati-hati dalam penggunaan kondisi antara WHILE dengan DO WHILE.

Berikut ini adalah contoh penggunaan sintaks do...while pada C dengan studi kasus serupa dengan contoh yang terdapat pada struktur while :

7. Pengulangan untuk menampilkan kalimat "Halo apa khabar ? " sebanyak 5 kali.

```
1  #include <stdio.h>

2  main()
3  {
4  int i; //deklarasi variabel pencacah
5  i=0;   //inisialisasi variabel pencacah
6  do
7  {
8  printf("\nHalo apa khabar ?");
9  i++;  //penambahan bilangan pencacah
10 }
11 while(i<5); //pengecekan kondisi
12 }
```

8. Pengulangan untuk menampilkan bilangan 1 hingga 5

```
1  #include <stdio.h>

2  main()
3  {
4  int i=1;
5  do
6  {
7  printf("\nBilangan ke-%d",i);
8  i++;
9  }
10 while(i<=5);
11 }
```

9. Pengulangan untuk meminta user memasukkan nilai

```
1  #include <stdio.h>

2  main()
3  {
4  int i=1;
```

```
5  int input;
6  do
7  {
8  printf("\nMasukkan bilangan ke-%d :
   ",i);
9  scanf("%d",&input);
10 printf("Bilangan yang Anda masukkan :
   %d",input);
11 i++;
12 }
13 while(i<=5);
14 }
```

10. Modifikasi contoh no 9 dimana user diminta untuk menentukan jumlah perulangan yang akan dilakukan

```
1  #include <stdio.h>

2  main()
3  {
4  int jml,i;
5  int input;
6  printf("Jumlah Bilangan : ");
7  scanf("%d",&jml);
8  i=1;
9  do(
10 {
11 printf("\nMasukkan bilangan ke-%d : ",i);
12 scanf("%d",&input);
13 printf("Bilangan yang Anda masukkan :
   %d",input);
14 i++;
15 }
16 while(i<=jml);
17 }
```

Sintaks do...while juga dapat digunakan untuk pengulangan yang belum diketahui jumlah iterasinya dan contoh berikut juga akan memperlihatkan keuntungan menggunakan sintaks do...while, dimana tidak perlu adanya inisialisasi kondisi agar dapat memasuki pengulangan.

11. Menampilkan menu sesuai pilihan user

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

```
2  main()
3  {
4  int pil; //deklarasi kondisi
5  do
6  {
7  printf("\nMENU : ");
8  printf("\n1. Red");
9  printf("\n2. Blue");
10 printf("\n3. Yellow");
11 printf("\n4. Green");
12 printf("\n5. Exit");
13 printf("\nPilihan : ");
14 scanf("%d",&pil);
15 switch(pil)
16 {
17 case 1 : printf("Anda memilih Merah");break;
18 case 2 : printf("Anda memilih Biru");break;
19 case 3 : printf("Anda memilih Kuning");break;
20 case 4 : printf("Anda memilih Hijau");break;
21 case 5 : printf("Anda memilih Keluar ...
   bye");break;
22 default : printf("Pilihan anda salah");
23 }
24 }
25 while(pil!=5);
26 }
```

12. User input angka hingga memilih keluar, angka yang telah dimasukkan akan dijumlah dan dihitung jumlah iterasi yang telah dilakukan.

```
1  #include <stdio.h>

2  main()
3  {
4  int i, input, jmlInput;
5  i=0;
6  jmlInput=0;
7  do
8  {
9  printf("Masukkan angka (input '99' untuk
   keluar) : ");
10 scanf("%d",&input);
11 jmlInput=jmlInput+input;
12 i++;
13 }
14 while (input != 99 );
15 printf("\nAnda telah input angka sebanyak %d
   kali",i-1);
16 printf("\nJumlah inputan anda : %d",jmlInput-
   99);
17 }
```

3.4 Sintaks FOR

Sintaks pengulangan FOR merupakan sintaks yang relatif paling mudah digunakan. Sintaks ini serupa dengan sintaks WHILE... DO... dalam hal pengecekan kondisi dilakukan di awal. Dalam menggunakan struktur pengulangan dengan sintaks FOR, pemrogram harus mendefinisikan nilai awal dan nilai akhir pencacah yang menunjukkan jumlah iterasi. Setiap kali iterasi berlangsung, nilai pencacah akan diubah. Jika pencacah sudah mencapai nilai akhir yang ditentukan, maka pengulangan akan berhenti.

Bentuk umum pengulangan dengan sintaks ***for*** pada C adalah:

```
for (inisialisasi ; kondisi ; perubahan)
{
      //pernyataan
}
```

Dimana :

- Inisialisasi : untuk memberikan nilai awal untuk variabel pencacah.
- Kondisi : kondisi pengulangan akan berhenti atau tidak.
- Perubahan : pengubahan nilai variabel pencacah untuk mencapai kondisi berhenti, dapat berupa kenaikan ataupun penurunan. Pengubah variabel pencacah tidak harus selalu naik atau turun satu, tetapi dapat dilakukan pengubahan variabel pencacah lebih dari satu.
- Pernyataan : aksi yang akan diulang

Berikut ini adalah contoh penggunaan sintaks for pada C dengan studi kasus serupa dengan contoh yang terdapat pada struktur while dan do ...while :

1. Pengulangan untuk menampilkan kalimat "Halo apa khabar ? " sebanyak 5 kali.

```
1  #include <stdio.h>

2  main()
3  {
4  int i; //deklarasi variabel pencacah
5  //inisialisasi variabel pencacah;
6  //pengecekan kondisi;
7  //penambahan bilangan pencacah
8  for(i=0;i<5;i++)
9  {
10 printf("\nHalo apa khabar ?");
11 }
12 }
```

2. Pengulangan untuk menampilkan bilangan 1 hingga 5

```
1  #include <stdio.h>

2  main()
3  {
4  int i;
5  for(i=1;i<=5;i++)
6  {
7  printf("\nBilangan ke-%d",i);
8      }
9  }
```

3. Pengulangan untuk meminta user memasukkan nilai

```
1  #include <stdio.h>

2  main()
3  {
4  int i;
5  int input;
6  for(i=1;i<=5;i++)
7  {
8  printf("\nMasukkan bilangan ke-%d : ",i);
9  scanf("%d",&input);
10 printf("Bilangan yang Anda masukkan :
   %d",input);
11    }
12 }
```

4. Modifikasi contoh no 15 dimana user diminta untuk menentukan jumlah perulangan yang akan dilakukan

```
1  #include <stdio.h>

2  main()
3  {
4  int jml,i;
5  int input;
6  printf("Jumlah Bilangan : ");
7  scanf("%d",&jml);
8  for(i=1;i<=jml;i++)
9  {
10 printf("\nMasukkan bilangan ke-%d :
   ",i);
11 scanf("%d",&input);
12 printf("Bilangan yang Anda masukkan :
   %d",input);
13 }
14 }
```

Sintaks for juga dapat digunakan untuk pengulangan yang belum diketahui jumlah iterasinya.

5. Menampilkan menu sesuai pilihan user

```
#include <stdio.h>

main()
{
int pil=0; //inisialisasi kondisi
for (;pil!=5;)
{
printf("\nMENU : ");
printf("\n1. Red");
printf("\n2. Blue");
printf("\n3. Yellow");
printf("\n4. Green");
printf("\n5. Exit");
printf("\nPilihan : ");
scanf("%d",&pil);
switch(pil)
{
case 1 : printf("Anda memilih Merah");break;
case 2 : printf("Anda memilih Biru");break;
case 3 : printf("Anda memilih Kuning");break;
case 4 : printf("Anda memilih Hijau");break;
case 5 : printf("Anda memilih Keluar ...
bye");break;
default : printf("Pilihan anda salah");
}
 }
}
```

6. User input angka hingga memilih keluar, angka yang telah dimasukkan akan dijumlah dan dihitung jumlah iterasi yang telah dilakukan.

```
#include <stdio.h>

main()
{
int i, input, jmlInput;
i=0;
jmlInput=0;
input=0;
for(;input!= 99;)
{
```

```
10 printf("Masukkan angka (input '99' untuk
   keluar) : ");
11 scanf("%d",&input);
12 jmlInput=jmlInput+input;
13 i++;
14 }
15 printf("\nAnda telah input angka sebanyak %d
   kali",i-1);
16 printf("\nJumlah inputan anda : %d",jmlInput-
   99);
17 }
```

Semua struktur pengulangan yang telah dijelaskan diatas dapat menggunakan variabel pencacah dengan hitungan menurun. Berikut adalah contoh penggunaan pencacah mundur untk ketiga struktur diatas

```
//sintaks while
#include <stdio.h>

main()
{
      int i=5;
      printf("Hitung Mundur !!!");
      while (i>0)
      {
            printf("\n%d",i);
            i--;
      }
      printf("\nDORRR!!!!");
}
```

```
//sintaks do...while
#include <stdio.h>

main()
{
      int i=5;
      printf("Hitung Mundur !!!");
      do
      {
            printf("\n%d",i);
            i--;
      }
      while (i>0);
```

```
      printf("\nDORRR!!!!");
}
```

```
//sintaks for
#include <stdio.h>

main()
{
      int i;
      printf("Hitung Mundur !!!");
      for (i=5;i>0;i--)
      {
            printf("\n%d",i);
      }
      printf("\nDORRR!!!!");
}
```

Output dari program diatas :

```
Hitung Mundur !!!
5
4
3
2
1
DORRR!!!!
```

3.5 *Sintaks Pengulangan Bersarang*

Sama halnya dengan struktur pemilihan, struktur pengulangan juga dapat disusun bersarang. Sebuah struktur pengulangan bisa berada dalam struktur pengulangan lainnya. Atau, sebuah struktur pengulangan bisa mengandung struktur pengulangan lain di dalamnya.

Berikut adalah contoh struktur pengulangan bersarang :

1. Pengulangan while di dalam while

```
1  #include <stdio.h>
2  main()
3  {
4  int i,j;
5  i=1;
6  while(i<=5)
7  {
8  j=1;
```

```
9  while(j<=5)
10 {
11 if(i==j)
12 printf("%d",j);
13 else
14 printf(" ");
15 j++;
16 }
17 printf("\n");
18 i++;
19 }
20 }
```

2. Pengulangan do…while di dalam do…while

```
1  #include <stdio.h>

2  main()
3  {
4  int i,j;
5  i=1;
6  do
7  {
8  j=1;
9  do
10 {
11 if(i==j)
12 printf("%d",j);
13 else
14 printf(" ");
15 j++;
16 }
17 while(j<=5);
18 printf("\n");
19 i++;
20 }
21 while(i<=5);
22 }
```

3. Pengulangan for di dalam for

```
1  #include <stdio.h>
2  main()
3  {
4  int i,j;
5  for(i=1;i<=5;i++)
```

```
6   {
7   for(j=1;j<=5;j++)
8   {
9   if(i==j)
10  printf("%d",j);
11  else
12  printf(" ");
13  }
14  printf("\n");
15  }
16  }
```

4. Pengulangan gabungan for di dalam while

```
1   #include <stdio.h>

2   main()
3   {
4   int i,j;
5   for(i=1;i<=5;i++)
6   {
7   j=1;
8   while(j<=5)
9   {
10  if(i==j)
11  printf("%d",j);
12  else
13  printf(" ");
14  j++;
15  }
16  printf("\n");
17  }
18  }
```

Setiap struktur pengulangan dapat digabung penggunaannya dengan struktur pengulangan yang lainnya, tetapi tetap harus memperhatikan karakter tiap struktur pengulangan.

Penggabungan penggunaan struktur pengulangan pada pengulangan bersarang dapat menggunakan lebih dari 2 pengulangan, bisa saja terjadi kebutuhan menggunakan struktur bersarang dengan kedalaman 3 atau lebih.

3.6 *Sintaks BREAK dan CONTINUE*

Sintaks *BREAK* dan *CONTINUE* merupakan sintaks yang digunakan untuk menghentikan pengulangan dan melanjutkan ke perintah atau aksi berikutnya. Sintaks BREAK dan CONTINUE dapat digunakan baik di dalam struktur pengulangan WHILE, DO...WHILE, dan FOR.

Sintaks *BREAK* digunakan untuk menghentikan pengulangan kemudian keluar dari struktur pengulangan tanpa melanjutkan perintah di dalam struktur pengulangan. Berikut adalah contoh penggunaan sintaks break :

5. Menampilkan 6 angka kemudian break ketika mencapai angka 4

```
1  #include <stdio.h>

2  main()
3  {
4  int i;
5  i=1;
6  while(i<=6)
7  {
8  printf("\nBilangan ke-%d",i);
9  if (i==4)
10 {
11 printf("\nOooops...break");
12 break;
13 }
14 i++;
15 }
16 }
```

6. Menampilkan menu sesuai pilihan user

```
1  #include <stdio.h>
2  main()
3  {
4  int i, input, jmlInput;
5  i=0;
6  jmlInput=0;
7  while (1)
8  {
9  printf("Masukkan angka (input '99' untuk
   keluar) : ");
10 scanf("%d",&input);
11 if (input==99)
```

```
12 {
13 break;
14 }
15 jmlInput=jmlInput+input;
16 i++;
17 }
18 printf("\nAnda telah input angka
   sebanyak %d kali",i);
19 printf("\nJumlah inputan anda :
   %d",jmlInput);
20 }
```

Program diatas merupakan modifikasi pemberhentian pengulangan dengan menggunakan sintaks Break, perhatikan perbedaan program diatas dengan contoh program no 5.
Pada contoh diatas, didalam while kondisi diberikan angka 1, angka 1 mengindikasikan true dimana sintaks di dalam while akan dijalankan jika kondisi bernilai true. Sehingga `while (1)` akan menyebabkan pengulangan akan dijalankan terus menerus, padahal setiap pengulangan membutuhkan kondisi berhenti, kondisi berhenti ini tidak dideklarasikan di dalam statement `while (kondisi)` tetapi berada di dalam tubuh while yaitu `if(input==99)`. Sedangkan untuk mengeluarkan dari pengulangan maka digunakan sintaks `break`.

Sintaks *CONTINUE* digunakan untuk kembali ke awal pengulangan tanpa menjalankan perintah berikutnya.
Contoh penggunaan sintaks CONTINUE adalah sebagai berikut :

7. Menampilkan 6 angka, ketika mencapai angka 3 maka akan di teruskan (continue) ke angka berikutnya

```
1  #include <stdio.h>

2  main()
3  {
4  int i;
5  for(i=1;i<=6;i++)
6  {
7  if ( i==4)
8  {
9  printf("\nContinue");
10 continue;
11 }
```

```
12 printf("\nBilangan ke-%d",i);
13 }
14 }
```

8. Contoh aplikasi dengan menggunakan break dan continue

```
1  #include <stdio.h>

2  main()
3  {
4  while (1)
5  {
6  char pil;
7  float nil1,nil2,nil3,rerata;
8  int i;
9  printf("\nHitung Rerata\n");
10 printf("Nilai ke 1 : ");
11 scanf("%f",&nil1);
12 printf("Nilai ke 2 : ");
13 scanf("%f",&nil2);
14 printf("Nilai ke 3 : ");
15 scanf("%f",&nil3);
16 rerata=(nil1+nil2+nil3)/3;
17 printf("Rerata : %f",rerata);
18 printf("\nHitung Rerata lagi(y/n): ");
19 scanf("%s",&pil);
20 if (pil=='y' || pil=='Y')
21 continue;
22 else
23 break;
24 }
25 }
```

Rangkuman

1. Struktur pengulangan memungkinkan program melakukan satu atau lebih aksi beberapa kali.
2. Tiga komponen pengendali aksi adalah inisialisasi, jumlah iterasi, dan kondisi berhenti.

3. Tiga struktur pengulangan yang dapat digunakan pada bahasa pemrograman C adalah struktur pengulangan dengan sintaks WHILE, DO...WHILE, dan FOR.
4. Struktur pengulangan dapat dibuat bersarang dengan sintaks pengulangan yang sama atau berbeda, bahkan dapat digabungkan dengan struktur pemilihan.
5. Pemilihan struktur pengulangan yang digunakan disesuaikan dengan kebutuhan, karena dalam beberapa kasus masing-masing struktur pengulangan dapat memberikan hasil yang berbeda.
6. Untuk keluar dari struktur pengulangan sebelum kondisi berhenti, kita dapat menggunakan sintaks BREAK
7. Untuk melanjutkan struktur pengulangan ke awal pengulangan maka dapat digunakan sintaks CONTINUE
8. Hal yang terpenting dari struktur pengulangan adalah kondisi berhenti yang akan memberikan kondisi apakah pengulangan dilakukan atau tidak.

Soal Latihan

1. Buatlah sebuah program untuk menampilkan bilangan ganjil antara 1-20 dengan menggunakan struktur while
2. Buatlah sebuah program untuk menampilkan bilangan kelipatan 3, dengan range berupa inputan dari user. Gunakan struktur do…while
   Contoh inputan :

```
Awal  : 1
Akhir : 10
Kelipatan 3 : 3 6 9
```

```
Awal  : 15
Akhir : 30
Kelipatan 3 : 15 18 21 24 27 30
```

3. Buatlah program untuk menghitung nilai faktorial suatu bilangan yang diinputkan oleh user.
   Rumus faktorial adalah sebagai berikut :
   n! = n*(n-1)*(n-2)*…*(n-(n-1))
   n merupakan inputan dari user.
   Boleh menggunakan struktur pengulangan manapun.
   Contoh inputan

```
Faktorial dari : 3
3! = 3*2*1=6
```

```
Faktorial dari : 5
5! = 5*4*3*2*1 = 120
```

4. Modifikasi program faktorial diatas, sehingga jika user memasukkan angka <= 0 maka akan diulang kembali untuk memasukkan angka
   Hint : gunakan continue

5. Buatlah sebuah program untuk menghitung gaji n karyawan, n merupakan sebuah bilangan bulat yang akan diinputkan oleh user, kemudian akan dihitung gaji perkaryawan dengan ketentuan sebagai berikut :
   Gaji per jam : Rp 10.000,-
   Bila jam kerja > 7 jam, maka sisa jam kerja dihitung sebagai lembur yg besarnya 1.5 * gaji per jam

Kemudian ditampilkan total gaji karyawan yang harus dibayarkan oleh perusahaan.
Jumlah jam kerja tiap karyawan akan diinputkan oleh user
Contoh inputan

```
Jumlah karyawan : 3 [inputan user]
Jam kerja karyawan 1 : 8 [inputan user]
Total Gaji : 85000
Jam kerja karyawan 2 : 6 [inputan user]
Total Gaji : 60000
Jam kerja karyawan 3 : 12 [inputan user]
Total Gaji : 145000

Total Gaji karyawan : 290000
```

6. Buatlah program untuk menampilkan angka seperti berikut :
Contoh inputan :

```
Jumlah angka : 5
1   5
 2 4
  3
 2 4
1   5
```

```
Jumlah angka : 6
1    6
 2  5
  34
  34
 2  5
1    6
```

```
Jumlah angka : 9
1       9
 2     8
  3   7
   4 6
    5
   4 6
  3   7
 2     8
1       9
```

7. Buatlah program untuk menampilkan bintang berbentuk belah ketupat seperti berikut :
Contoh inputan :

```
Jumlah bintang : 5
```

```
   *
  * *
*    *
  * *
   *
```

```
Jumlah bintang : 6
   **
  *  *
*    *
*    *
  *  *
   **
```

```
Jumlah bintang : 9
     *
    * *
   *   *
  *     *
*       *
  *     *
   *   *
    * *
     *
```