Perulangan

Bahasa C

Baharuddin Aziz

Semester I 2019/2020

Tujuan Perkuliahan

- Mengenal konsep perulangan
- Memahami penerapan perulangan pada bahasa C

Pendahuluan [1/2]

- ***Apa yang akan dilakukan saat diminta cetak kalimat berulang?***
- Misalnya:
  - Tolong tampilkan kalimat "`Belajar Pemrograman!`" ke layar!
- Mungkin dapat dilakukan dengan fungsi `printf()` sebanyak 10 kali.

```
#include <stdio.h>
void main(){
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
}
```

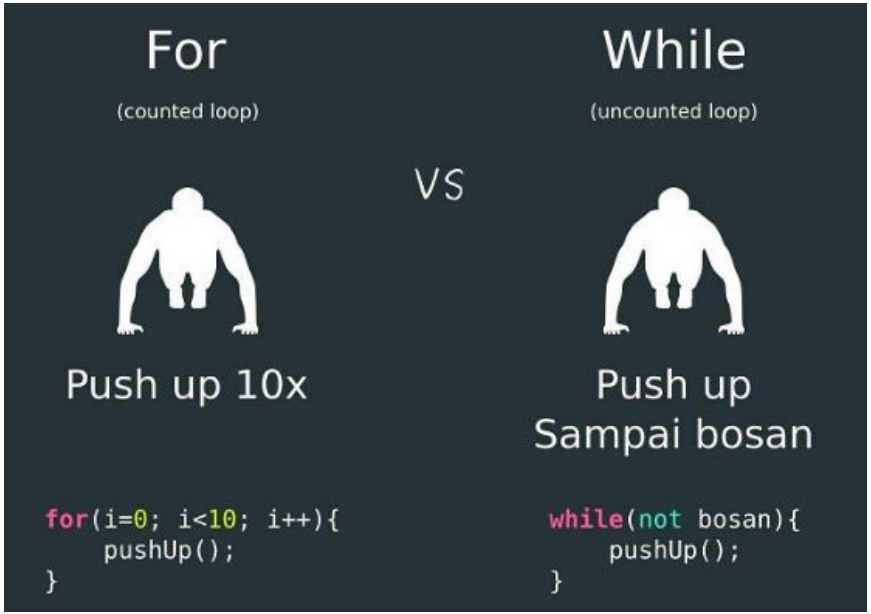
Pendahuluan [2/2]

- Apakah boleh seperti ini?
  - Ya, boleh-boleh saja.
- Tapi, ***bagaimana jika menampilkan 1000 kali?***
  - *Bisa, tapi ribet!*
- Karena itu, perlu **perulangan**.
- Perulangan membantu mengeksekusi kode yang berulang berapapun yang diinginkan.

```
#include <stdio.h>
void main(){
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
    printf("Belajar Pemrograman!\n");
}
```

Perulangan

- Secara umum, perulangan dibagi 2:
  a. ***Counted** Loop*:
  perulangan yang jelas dan sudah tentu banyak perulangannya.
  Contoh: `for`
  b. ***Uncounted** Loop*:
  perulangan yang tidak jelas berapa kali ia harus mengulang.
  Contoh: `while`, `do-while`



Perulangan

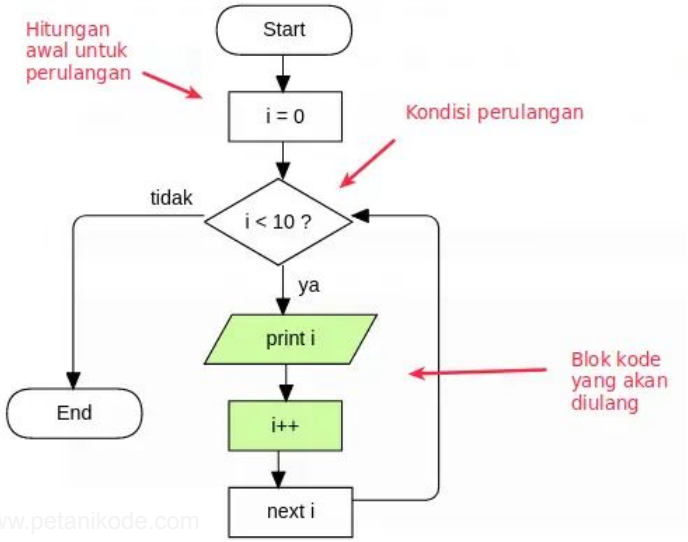
`for`

Perulangan `for` [1/2]

- **Perulangan `for`**:
  perulangan yang termasuk ke dalam *counted loop* karena sudah jelas berapa kali akan diulang.
- **Contoh**:

```
for(int i = 0; i < 10; i++){
    printf("Perulangan ke-%i\n", i);
}
```

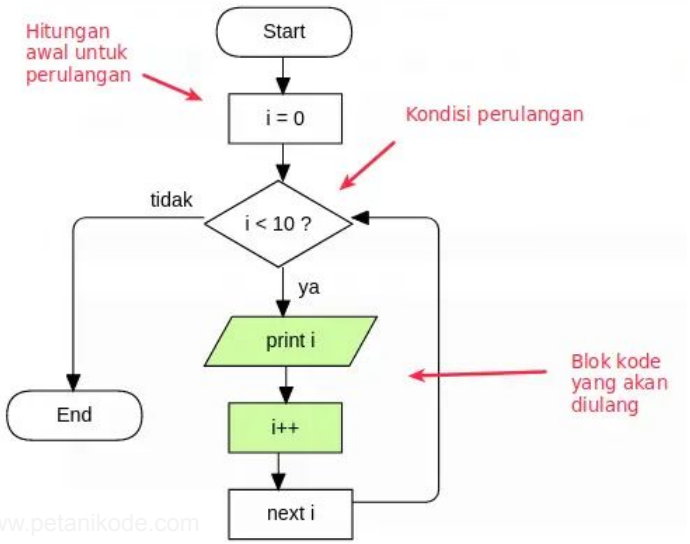
- Perlu diperhatikan:
  kondisi yang ada di dalam kurung setelah kata `for`.



Perulangan `for` [2/2]

```
for(int i = 0; i < 10; i++){
    printf("Perulangan ke-%i\n", i);
}
```

- Kondisi ini akan menentukan:
  - Hitungan akan **dimulai** dari 0 (`i = 0`);
  - Sampai berapa? **Sampai** `i < 10`;
  - Lalu di setiap perulangan i akan **bertambah** +1 (`i++`).
  - Variabel `i` pada perulangan `for` berfungsi menyimpan nilai hitungan.
- Setiap perulangan yang dilakukan akan selalu bertambah satu karena telah ditentukan di bagian `i++`.



Contoh Program: `perulangan.c`

```c
#include <stdio.h>

void main(){
   for(int i = 0; i < 10; i++){
      printf("Perulangan ke-%i\n", i);
   }
}
```

- ***Apakah nama variabel harus `i`?***
  - Tidak, boleh pakai nama lain.

```
D:\>gcc -o perulangan.exe perulangan.c

D:\>perulangan
Perulangan ke-0
Perulangan ke-1
Perulangan ke-2
Perulangan ke-3
Perulangan ke-4
Perulangan ke-5
Perulangan ke-6
Perulangan ke-7
Perulangan ke-8
Perulangan ke-9
```

Contoh Program: `perulangan1.c`

```c
#include <stdio.h>

void main(){
   for(int count = 0; count < 10; count+=2){
       printf("Perulangan ke-%i\n",  count);
   }
}
```

- ***Apakah nama variabel harus `i`?***
  - Tidak, boleh pakai nama lain.
- Pada contoh di samping, dilakukan perulangan dimulai dari nol 0.
  - Di setiap perulangan nilai variabel **`count`** akan ditambah 2 (**`count+=2`**).

```
D:\>gcc -o perulangan1.exe perulangan1.c

D:\>perulangan1
Perulangan ke-0
Perulangan ke-2
Perulangan ke-4
Perulangan ke-6
Perulangan ke-8
```

Contoh Program: `perulangan2.c` [1/2]

```c
#include <stdio.h>

void main(){
   for(int counter = 5; counter > 0; counter--){
      printf("Perulangan ke-%i\n",  counter);
   }
}
```

```
D:\>gcc -o perulangan2.exe perulangan2.c

D:\>perulangan2
Perulangan ke-5
Perulangan ke-4
Perulangan ke-3
Perulangan ke-2
Perulangan ke-1
```

- ***Bagaimana counter perulangan dimulai dari angka yang lebih besar ke kecil? (hitung mundur)***
- Isi nilai *counter* dengan nilai terbesarnya.
- Misal: hitungan 5 sampai 1.
  - Isi nilai awal, **`counter = 5`**.
  - Kondisi perbandingan, **`counter > 0`**. Artinya: perulangan dilakukan selama nilai counter lebih besar dari 0.
  - Kurangi (-1) nilai *counter* di setiap perulangan (**`counter--`**).

Contoh Program: `perulangan2.c` [2/2]

```c
#include <stdio.h>

void main(){
   for(int counter = 5; counter > 0; counter--){
      printf("Perulangan ke-%i\n",  counter);
   }
}
```

```
D:\>gcc -o perulangan2.exe perulangan2.c

D:\>perulangan2
Perulangan ke-5
Perulangan ke-4
Perulangan ke-3
Perulangan ke-2
Perulangan ke-1
```

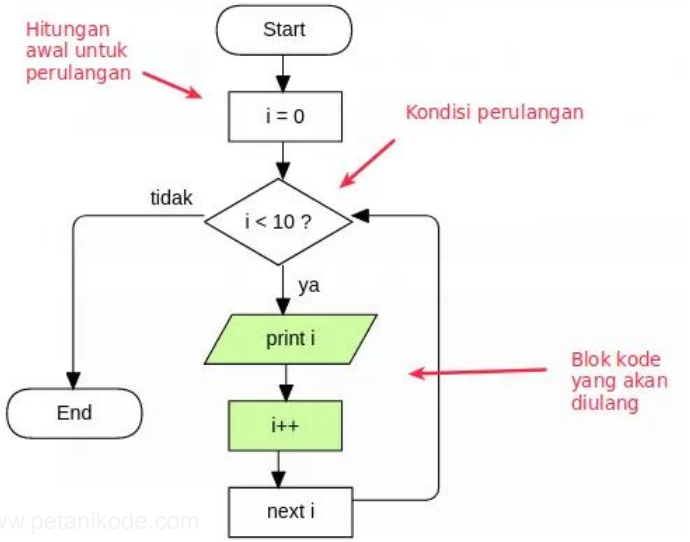
- ***Mengapa tidak sampai nol (0)?***
  - Karena kondisi **`counter > 0.`**
  - Apabila *counter* bernilai 0, maka kondisi ini akan menjadi *false*.
- Kecuali menggunakan operator lebih besar sama dengan (>=)
  - Jika *counter* bernilai 0, maka kondisi akan menjadi *true*.

Perulangan

`while`

Perulangan `while`

- **Perulangan `while`:**
  perulangan yang termasuk dalam perulangan *uncounted loop*.
- Perulangan `while` juga dapat menjadi perulangan yang *counted loop* dengan memberikan counter di dalamnya.
- Bentuk *flowchart*-nya sama seperti *flowchart* `for`.



Contoh Program: `perulangan3.c` [1/2]

```c
#include <stdio.h>

void main(){
    char ulangi = 'y';
    int counter = 0;

    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    while(ulangi == 'y'){
        printf("Apakah mau mengulang?\n");
        printf("Jawab (y/t): ");
        scanf(" %c", &ulangi);
        // increment counter
        counter++;
    }
    printf("----------\n");
    printf("Perulangan Selesai!\n");
    printf("Perulangan = %i kali.\n", counter);
}
```

```
D:\>gcc -o perulangan3.exe perulangan3.c

D:\>perulangan3
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 3 kali.

D:\>perulangan3
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 0 kali.
```

Contoh Program: `perulangan3.c` [2/2]

```c
#include <stdio.h>

void main(){
    char ulangi = 'y';
    int counter = 0;

    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    while(ulangi == 'y'){
        printf("Apakah mau mengulang?\n");
        printf("Jawab (y/t): ");
        scanf(" %c", &ulangi);
        // increment counter
        counter++;
    }
    printf("----------\n");
    printf("Perulangan Selesai!\n");
    printf("Perulangan = %i kali.\n", counter);
}
```

- Perhatikan pada bagian ini:

```c
while(ulangi == 'y'){
    printf("Apakah mau mengulang?\n");
    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    // increment counter
    counter++;
}
```

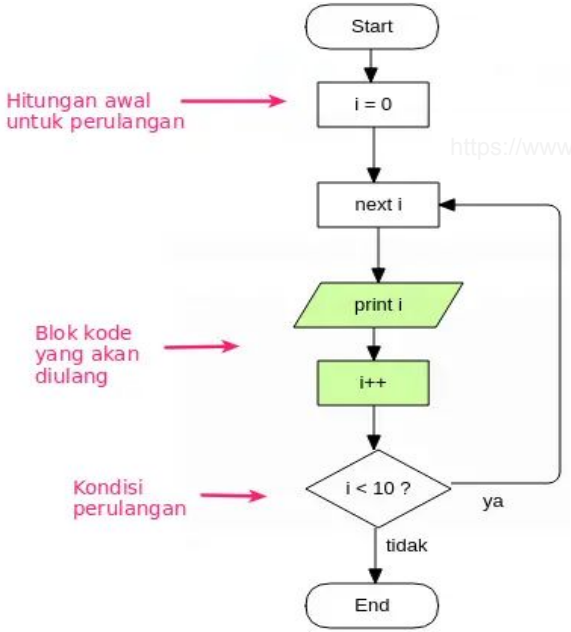
  - Perulangan terjadi selama variabel `ulangi='y'`.
  - Fungsi `scanf()` untuk ambil *input*.
  - Selama input = y perulangan terus dilakukan. Tapi jika jawab lain, perulangan akan dihentikan karena kondisi perulangannya tidak terpenuhi.

Perulangan

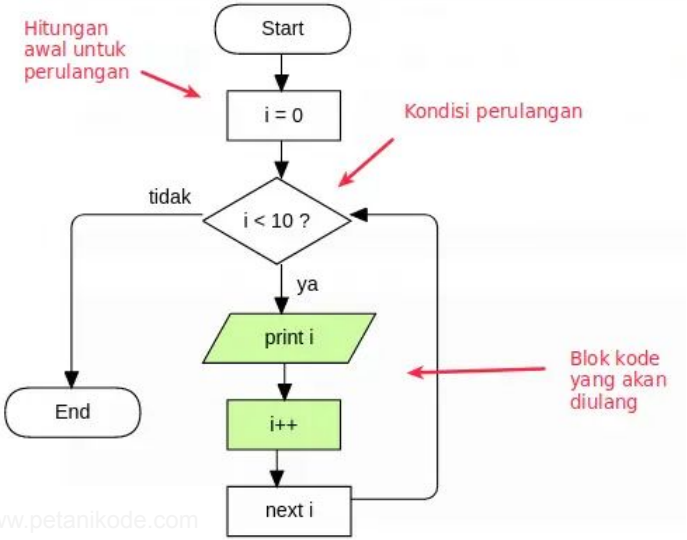
`do-while`

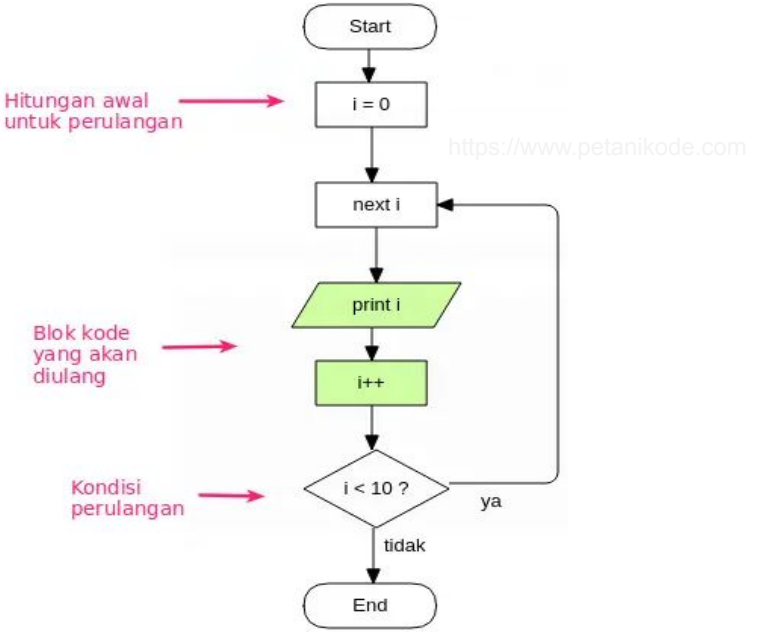
Perulangan `do-while` [1/4]

- Perulangan `do-while` sama seperti perulangan `while`.
- Perbedaannya:
  Perulangan `do-while` melakukan perulangan sebanyak **1 kali lebih dahulu**, lalu mengecek kondisi yang ada di dalam kurung `while`.

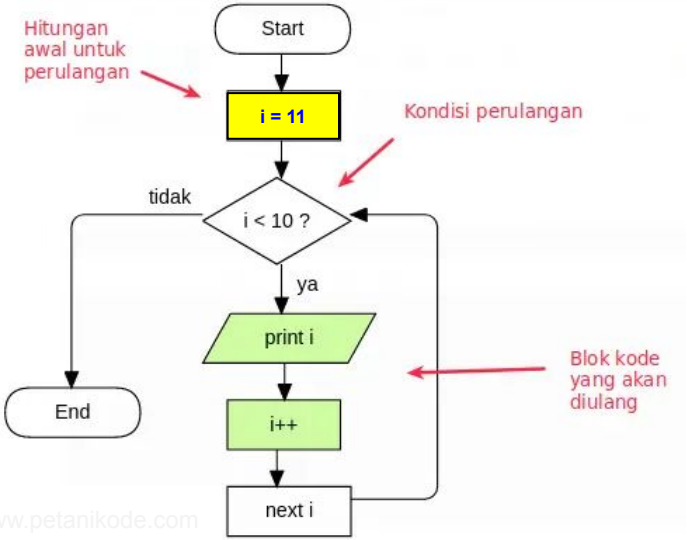


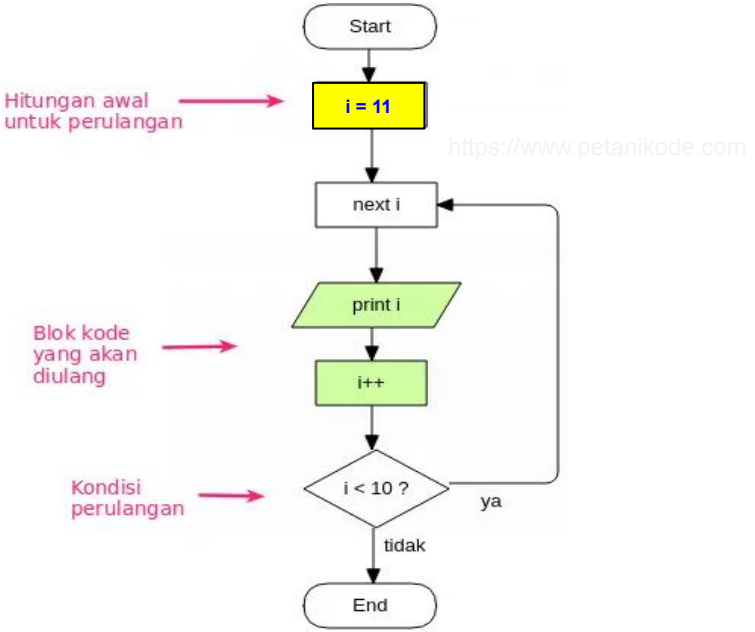
Perulangan `do-while` [2/4]





Perulangan `do-while` [3/4]





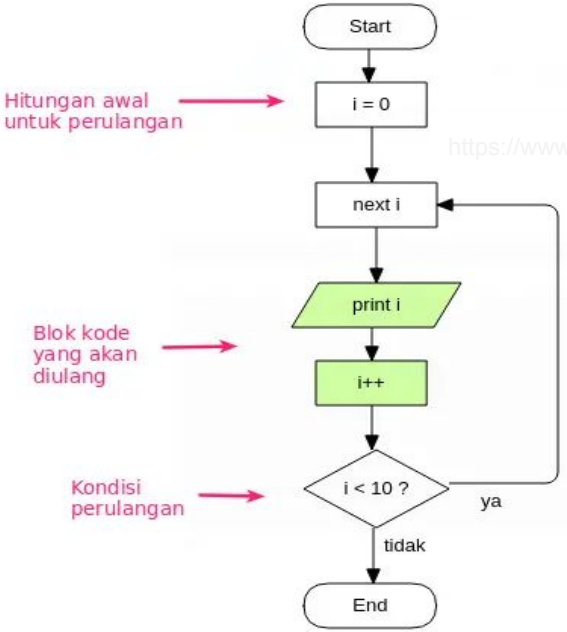
Bagaimana jika i = 11?

Perulangan `do-while` [4/4]

- Bentuk kodenya seperti ini:

```
do {
    // blok kode yang akan diulang
} while (<kondisi>);
```

- Jadi perbedaanya:
  - Perulangan **`do-while`**:
    mengecek kondisi di belakang/**akhir**
    (<u>sesudah</u> mengulang)
  - Perulangan **`while`**:
    mengecek kondisi di depan/**awal**
    (<u>sebelum</u> mengulang).



Contoh Program: `perulangan4.c` [1/4]

```c
#include <stdio.h>

void main(){
    char ulangi;
    int counter = 0;

    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    do {
        printf("Apakah mau mengulang?\n");
        printf("Jawab (y/t): ");
        scanf(" %c", &ulangi);
        // increment counter
        counter++;
    } while(ulangi == 'y');
    printf("----------\n");
    printf("Perulangan Selesai!\n");
    printf("Perulangan = %i kali.\n", counter);
}
```

```
D:\>gcc -o perulangan4.exe perulangan4.c

D:\>perulangan4
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 3 kali.

D:\>perulangan4
Jawab (y/t): t
Apakah mau mengulang?
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 1 kali.
```

Contoh Program: `perulangan4.c` [2/4]

```c
#include <stdio.h>

void main(){
    char ulangi;
    int counter = 0;

    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    do {
        printf("Apakah mau mengulang?\n");
        printf("Jawab (y/t): ");
        scanf(" %c", &ulangi);
        // increment counter
        counter++;
    } while(ulangi == 'y');
    printf("----------\n");
    printf("Perulangan Selesai!\n");
    printf("Perulangan = %i kali.\n", counter);
}
```

- Contoh di samping sama seperti contoh pada perulangan `while`.
- Saat perulangan pertama, coba batalkan perulangan dengan beri input `t`.

Contoh Program: `perulangan4.c` [3/4]

```c
#include <stdio.h>

void main(){
    char ulangi;
    int counter = 0;

    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    do {
        printf("Apakah mau mengulang?\n");
        printf("Jawab (y/t): ");
        scanf(" %c", &ulangi);
        // increment counter
        counter++;
    } while(ulangi == 'y');
    printf("----------\n");
    printf("Perulangan Selesai!\n");
    printf("Perulangan = %i kali.\n", counter);
}
```

Contoh program: `perulangan3.c`

```c
#include <stdio.h>

void main(){
    char ulangi = 'y';
    int counter = 0;

    printf("Jawab (y/t): ");
    scanf(" %c", &ulangi);
    while(ulangi == 'y'){
        printf("Apakah mau mengulang?\n");
        printf("Jawab (y/t): ");
        scanf(" %c", &ulangi);
        // increment counter
        counter++;
    }
    printf("----------\n");
    printf("Perulangan Selesai!\n");
    printf("Perulangan = %i kali.\n", counter);
}
```

Contoh Program: `perulangan4.c` [4/4]

```
D:\>gcc -o perulangan4.exe perulangan4.c

D:\>perulangan4
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 3 kali.

D:\>perulangan4
Jawab (y/t): t
Apakah mau mengulang?
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 1 kali.
```

```
D:\>gcc -o perulangan3.exe perulangan3.c

D:\>perulangan3
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): y
Apakah mau mengulang?
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 3 kali.

D:\>perulangan3
Jawab (y/t): t
----------
Perulangan Selesai!
Perulangan = 0 kali.
```

Perulangan Bersarang
(*Nested Loop*)

Perulangan Bersarang

Semua bentuk blok percabangan di atas dapat dibuat di dalam percabangan yang lainnya, disebut dengan **perulangan bersarang** atau ***nested loop***.

Contoh Program: `perulangan5.c` [1/2]

```c
#include <stdio.h>

void main(){
    for(int i = 0; i < 5; i++){
        for(int j = 0; j < 3; j++){
            printf("Loop ke (%d, %d)\n", i, j);
        }
    }
}
```

```
D:\>gcc -o perulangan5.exe perulangan5.c

D:\>perulangan5
Loop ke (0, 0)
Loop ke (0, 1)
Loop ke (0, 2)
Loop ke (1, 0)
Loop ke (1, 1)
Loop ke (1, 2)
Loop ke (2, 0)
Loop ke (2, 1)
Loop ke (2, 2)
Loop ke (3, 0)
Loop ke (3, 1)
Loop ke (3, 2)
Loop ke (4, 0)
Loop ke (4, 1)
Loop ke (4, 2)
```

Contoh Program: `perulangan5.c` [2/2]

```c
#include <stdio.h>

void main(){
    for(int i = 0; i < 5; i++){
        for(int j = 0; j < 3; j++){
            printf("Loop ke (%d, %d)\n",  i, j);
        }
    }
}
```

- Pada perulangan di samping, digunakan 2 perulangan `for`.
  - Perulangan **pertama** menggunakan variabel **`i`** sebagai *counter*.
  - Perulangan **kedua** menggunakan variabel **`j`** sebagai *counter*.

Pertanyaan?