Konsep Pemrograman

5. Perulangan Proses (*Looping*) - 1

Umi Sa'adah
Entin Martiana Kusumaningtyas
Tri Hadiah Muliawati
2020

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Overview

- Pendahuluan
- Perulangan `for`
- Perulangan `while`
- Perulangan `do while`

Pendahuluan

- Pada semua bahasa pemrograman, perulangan proses ditangani dengan suatu mekanisme yang disebut *looping*.
- Dengan menggunakan *looping*, suatu proses yang berulang (misalnya menampilkan tulisan yang sama sebanyak seratus kali pada layar) dapat diimplementasikan dengan kode program yang lebih pendek.
- User tidak perlu melakukan *copy-paste* kode program yang ingin dijalankan secara berulang berkali-kali.
- Untuk menerapkan perulangan, Bahasa C menyediakan 3 *looping statement*, antara lain:
  - `for`
  - `while`
  - `do-while`

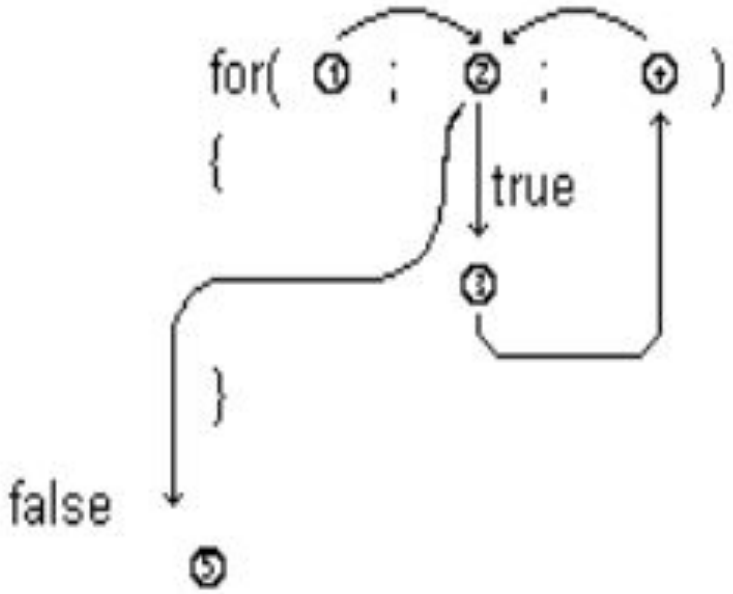
Pemilihan *Looping Statement*

- Statement `for` digunakan apabila user sudah mengetahui jumlah perulangan yang ingin dilakukan.
- Apabila jumlah perulangan belum diketahui, maka user bisa menggunakan salah satu looping statement berikut:

1. Statement `while`
   - Pengecekan kondisi akan dilakukan di awal
   - *Statement* yang berada di dalam *body loop* akan dijalankan selama masih memenuhi kondisi yang ditetapkan
   - Ada kemungkinan *statement* yang berada di dalam *body loop* tidak dijalankan sama sekali.
2. Statement `do-while`
   - Pengecekan kondisi akan dilakukan di akhir
   - *Statement* yang berada di dalam *body loop* akan dijalankan selama masih memenuhi kondisi yang ditetapkan
   - *Statement* yang berada di dalam *body loop* akan dijalankan minimal 1 kali

Statement `for`

Statement `for`



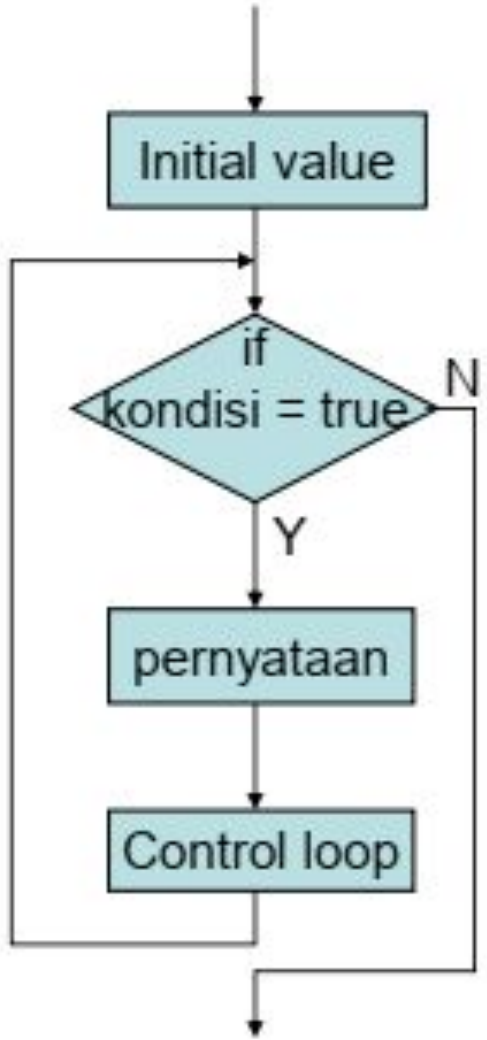
- Bentuk umum:

```
for(ekspresi1; ekspresi2; ekspresi3){
  // body of loop
  statement;
  statement;
  …
  statement;
}
```

- `ekspresi1`: inisialisasi nilai awal untuk variabel pengendali *loop.*
- `ekspresi2`: *continue condition*, kondisi yang harus terpenuhi agar *loop* tetap berjalan.
- `ekspresi3`: pengatur perubahan (naik/turun) nilai dari variabel pengendali *loop*.
- Ketiga ekspresi dipisahkan dengan tanda titik koma.
- Apabila hanya ada 1 *statement* yang berada di dalam *body of loop*, maka tidak perlu menggunakan kurung kurawal.

Contoh 1

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 1; i<=3; i++){
        printf("ini baris ke-%d\n", i);
    }
}
```



- `ekspresi1`: *initial value* → i = 1
- `ekspresi2`: *continue condition* → i <= 3
- `ekspresi3`: pengatur perubahan nilai dari variabel pengendali *loop*.
  → i++

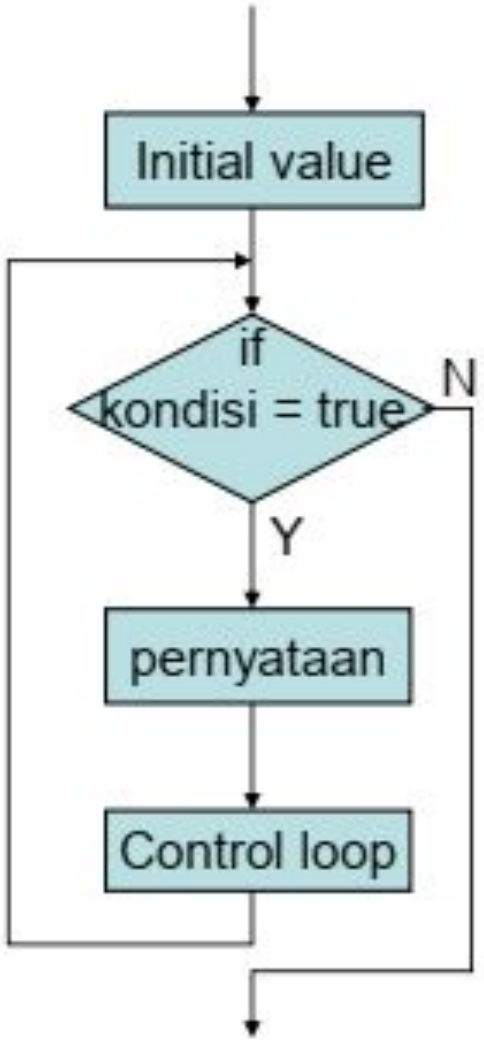
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 1; i<=3; i++){
        printf("ini baris ke-%d\n", i);
    }
}
```

Result

```
$gcc -o main *.c
$main
ini baris ke-1
```

Output iterasi ke-1



- Iterasi 1
  i = 1
  i <= 3 □ 1 <= 3 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “ini baris ke-1”
  i++ □ i = i+1 □ i = 1+1 = 2

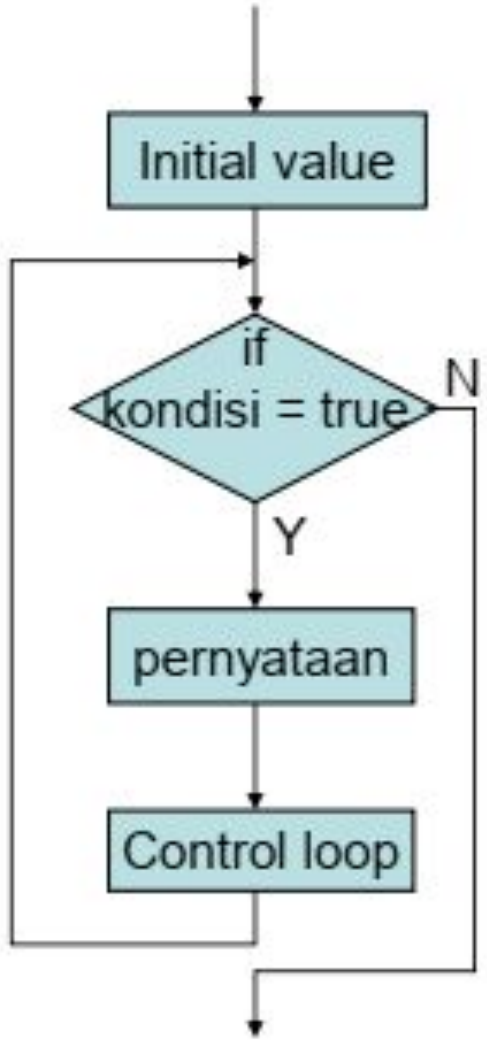
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 1; i<=3; i++){
        printf("ini baris ke-%d\n", i);
    }
}
```

Result

```
$gcc -o main *.c
$main
ini baris ke-1
ini baris ke-2
```

Output iterasi ke-2



- Iterasi 2
  i = 2
  i <= 3 □ 2 <= 3 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “ini baris ke-2”
  i++ □ i = i+1 □ i = 2+1 = 3

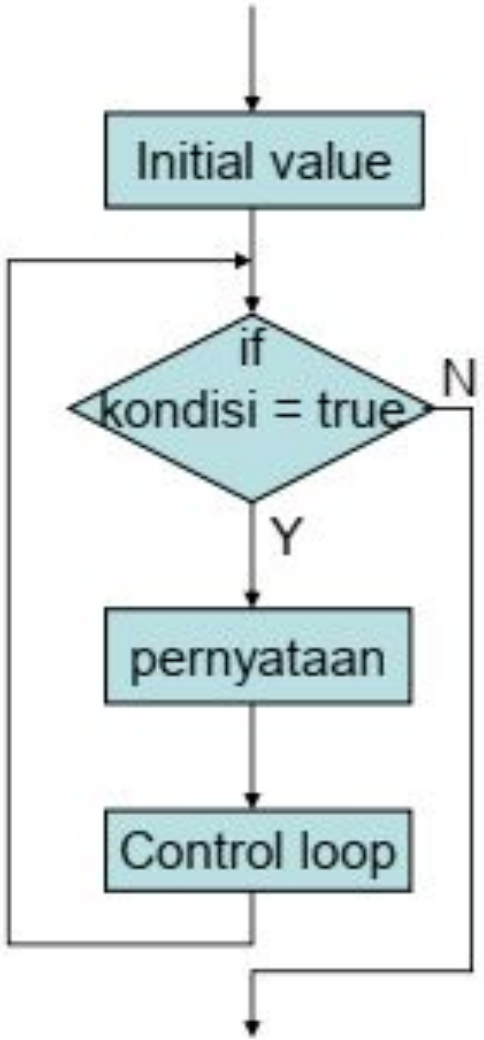
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 1; i<=3; i++){
        printf("ini baris ke-%d\n", i);
    }
}
```

Result

```
$gcc -o main *.c

$main

ini baris ke-1
ini baris ke-2
ini baris ke-3
```

Output iterasi ke-3



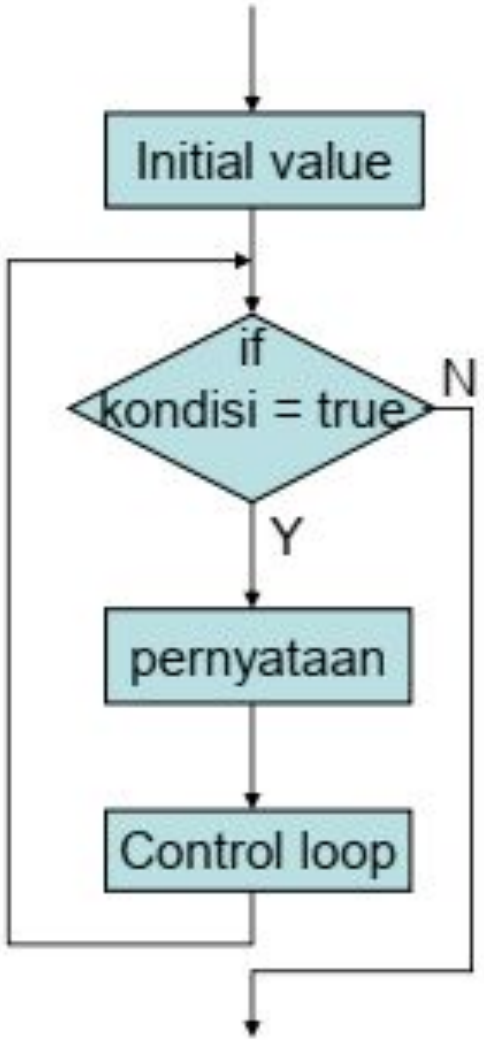
- Iterasi 3
  i = 3
  i <= 3 □ 3 <= 3 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “ini baris ke-3”
  i++ □ i = i+1 □ i = 3+1 = 4

Contoh 1

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 1; i<=3; i++){
        printf("ini baris ke-%d\n", i);
    }
}
```

Result

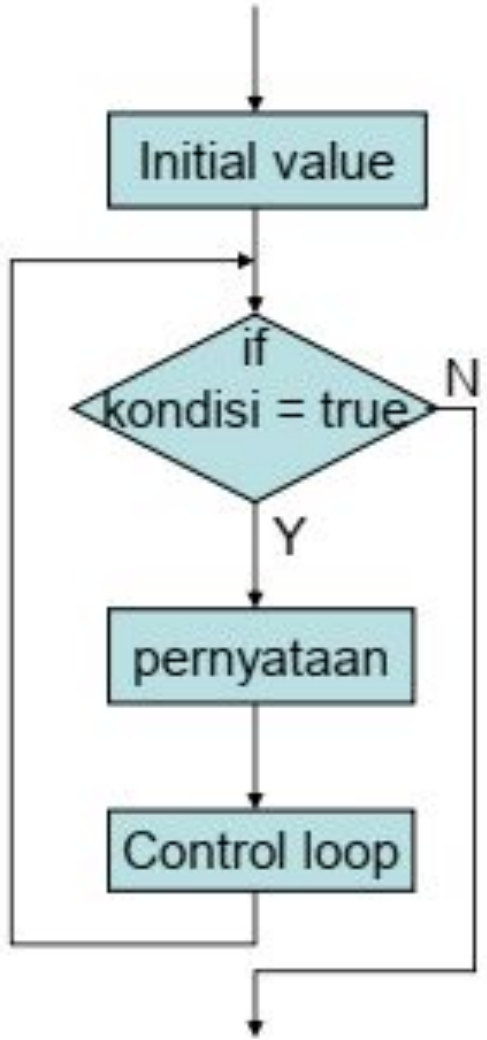
```
$gcc -o main *.c

$main

ini baris ke-1
ini baris ke-2
ini baris ke-3
```



- Iterasi 4
  i = 4
  i <= 3 □ 4 <= 3 □ FALSE
  continue condition tidak terpenuhi, maka keluar dari *loop*.

Contoh 2

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10; i>=1; i-=3){
        printf("%d\t", i);
    }
}
```



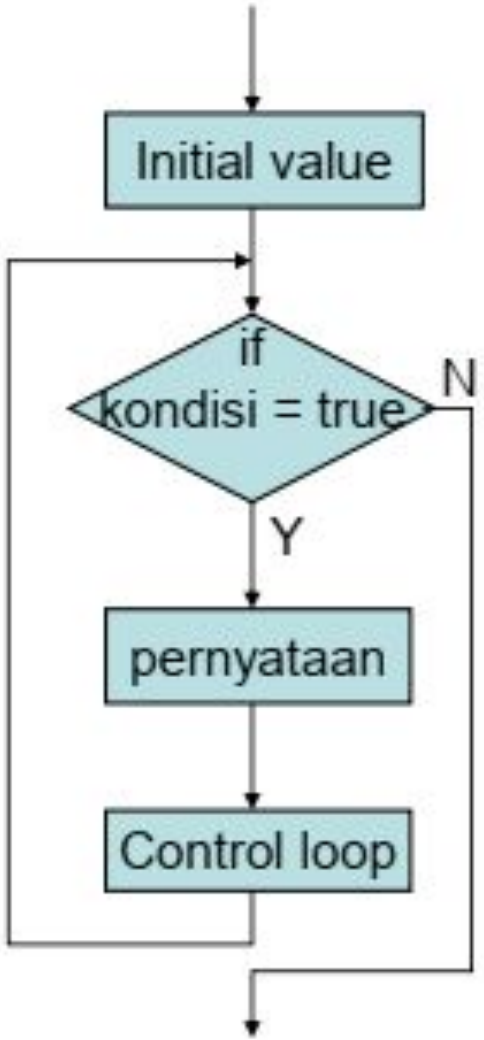
- `ekspresi1`: *initial value* → i = 10
- `ekspresi2`: *continue condition* → i >= 1
- `ekspresi3`: pengatur perubahan nilai dari variabel pengendali *loop*.
  → i-=3 → i = i - 3

Contoh 2

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10; i>=1; i-=3){
        printf("%d\t", i);
    }
}
```

```
Result

$gcc -o main *.c

$main

10
```

Output iterasi ke-1



- Iterasi ke-1
  i = 10
  i >= 1 □ 10 >= 1 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “10 ”
  i-= 3 □ i = i-3 □ i = 10-3 = 7

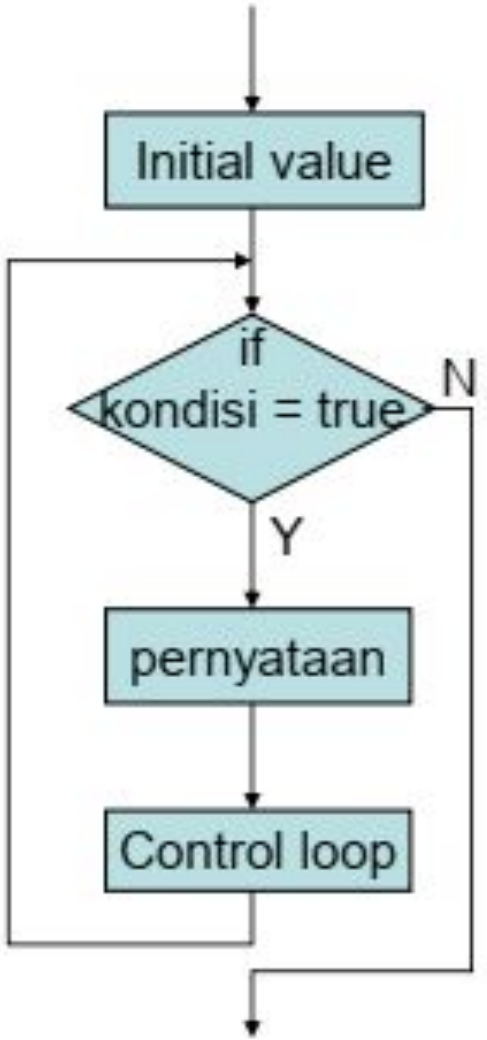
Contoh 2

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10; i>=1; i-=3){
        printf("%d\t", i);
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7
```

Output iterasi ke-2



- Iterasi ke-2
  i = 7
  i >= 1 □ 7 >= 1 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “7 ”
  i-= 3 □ i = i-3 □ i = 7-3 = 4

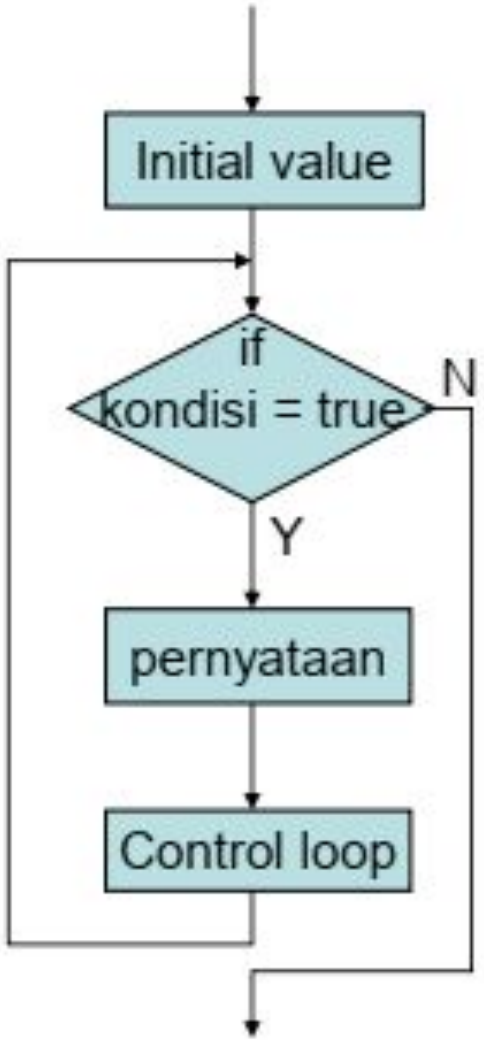
Contoh 2

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10; i>=1; i-=3){
        printf("%d\t", i);
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7       4
```

Output iterasi ke-3



- Iterasi ke-3
  i = 4
  i >= 1 □ 4 >= 1 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “4 ”
  i-= 3 □ i = i-3 □ i = 4-3 = 1

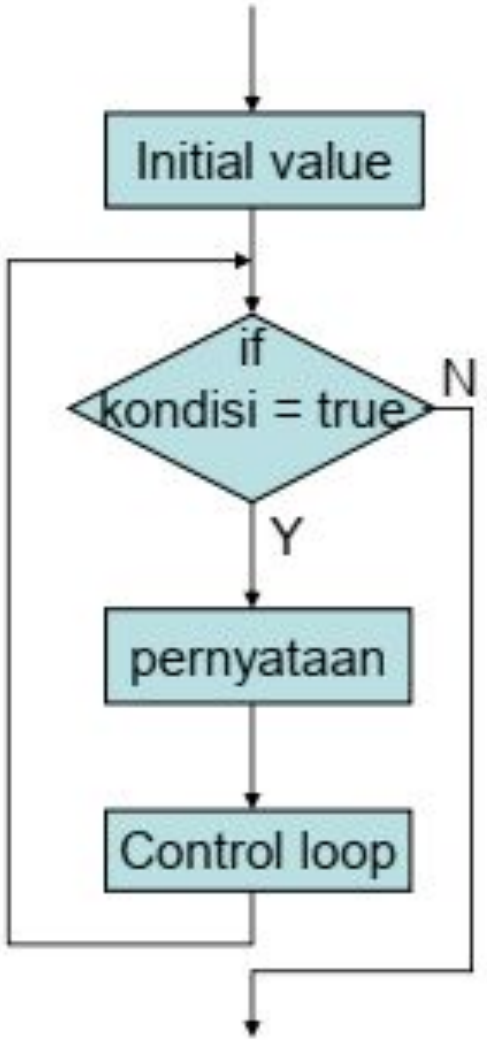
Contoh 2

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10; i>=1; i-=3){
        printf("%d\t", i);
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7       4       1
```

Output iterasi ke-4



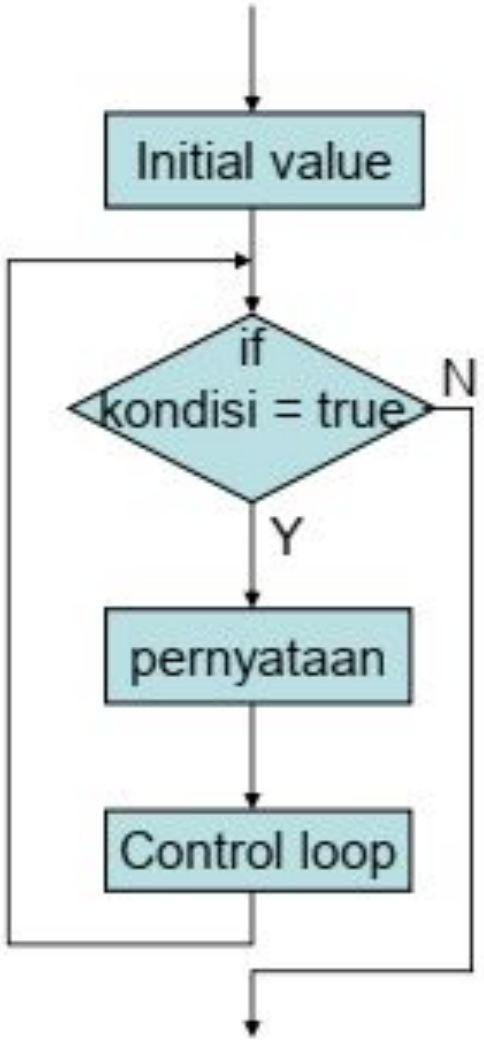
- Iterasi ke-4
  i = 1
  i >= 1 □ 1 >= 1 □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “1 ”
  i-= 3 □ i = i-3 □ i = 1-3 = -2

Contoh 2

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 10; i>=1; i-=3){
        printf("%d\t", i);
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7       4       1
```



- Iterasi ke-5
  i = -2
  i >= 1 □ -2 >= 1 □ FALSE
  continue condition tidak terpenuhi, maka keluar dari *loop*.

Statement `while`

Statement `while`

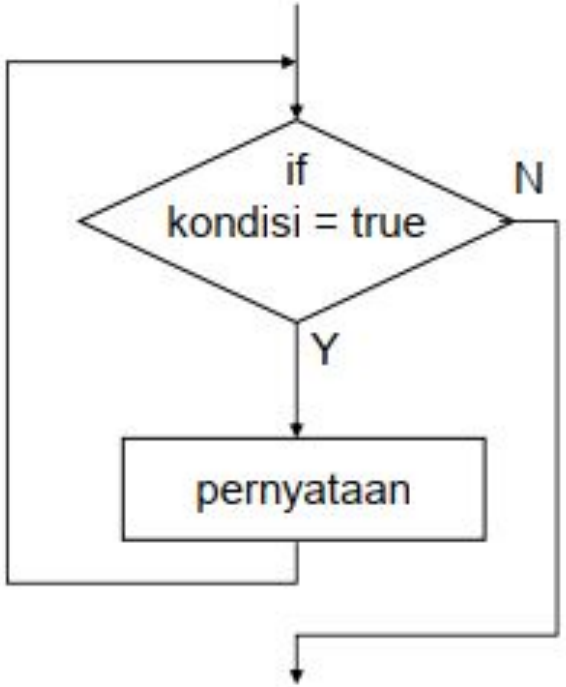
- Bentuk umum:

```
while(continue_condition) {
  //body_of_loop
  statement;
  statement;
  …
  statement;
}
```

- `continue_condition`: kondisi yang harus terpenuhi agar *loop* tetap berjalan.
- Apabila hanya ada 1 *statement* yang berada di dalam *body of loop*, maka tidak perlu menggunakan kurung kurawal.
- Agar *loop* dapat berhenti, maka di dalam *body of loop* perlu ada *statement* yang bisa merubah nilai kondisi sehingga *looping* dapat berhenti.

Contoh 1

```c
#include <stdio.h>

int main()
{
    int i = 10;
    while(i>=1){
        printf("%d\t", i);
        i -= 3;
    }
}
```



- *Continue condition*: i >= 1
- Variabel yang dicek pada *continue condition* perlu diberi nilai awal, yakni i
- *Statement* yang merubah nilai kondisi: i -= 3

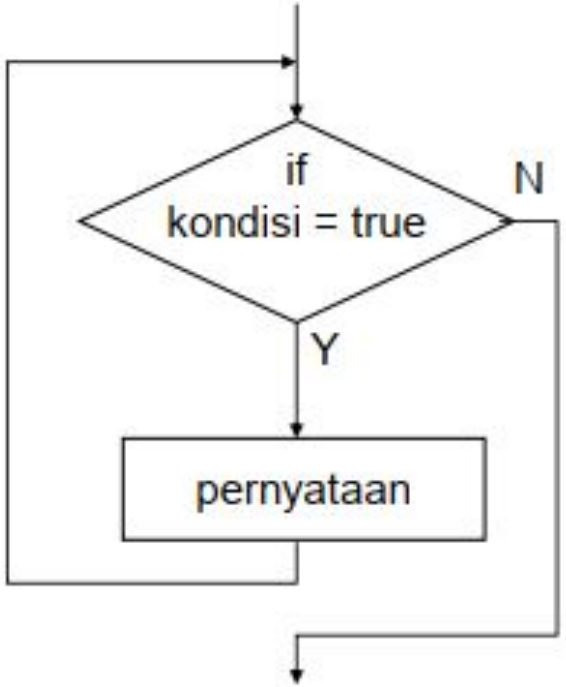
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i = 10;
    while(i>=1){
        printf("%d\t", i);
        i -= 3;
    }
}
```

Result

```
$gcc -o main *.c
$main
10
```

Output iterasi ke-1



- Iterasi ke-1
  continue condition □ i >= 1 □ 10 >= 1 □ TRUE
  baris ke-7 dijalankan, sehingga muncul output “10 ”
  baris ke-8 dijalankan, sehingga nilai i berubah menjadi 10-3 = 7

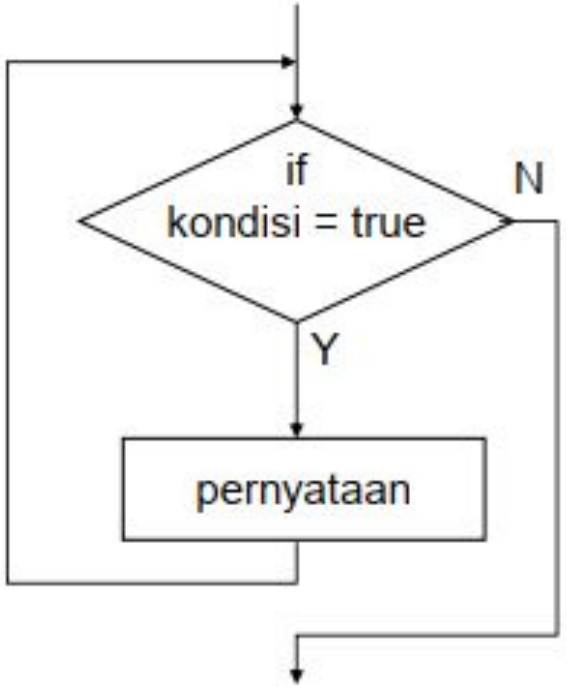
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i = 10;
    while(i>=1){
        printf("%d\t", i);
        i -= 3;
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7
```

Output iterasi ke-2



- Iterasi ke-2
  continue condition □ i >= 1 □ 7 >= 1 □ TRUE
  baris ke-7 dijalankan, sehingga muncul output “7 ”
  baris ke-8 dijalankan, sehingga nilai i berubah menjadi 7-3 = 4

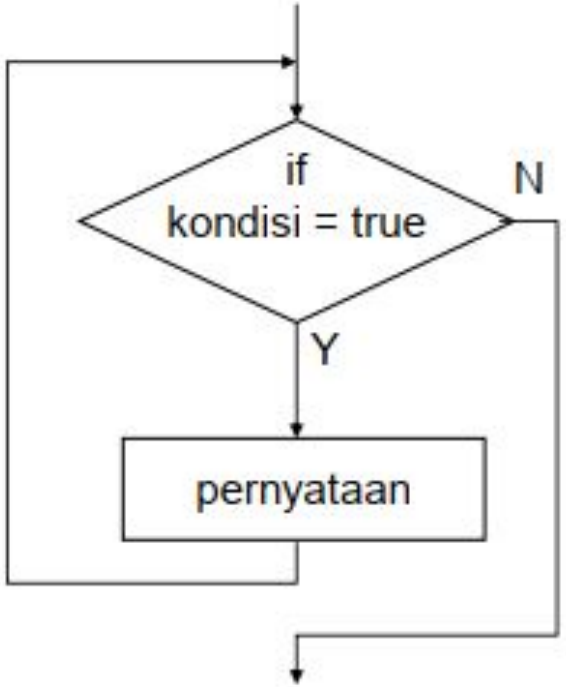
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i = 10;
    while(i>=1){
        printf("%d\t", i);
        i -= 3;
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7       4
```

Output iterasi ke-3



- Iterasi ke-3
  continue condition □ i >= 1 □ 4 >= 1 □ TRUE
  baris ke-7 dijalankan, sehingga muncul output “4 ”
  baris ke-8 dijalankan, sehingga nilai i berubah menjadi 4-3 = 1

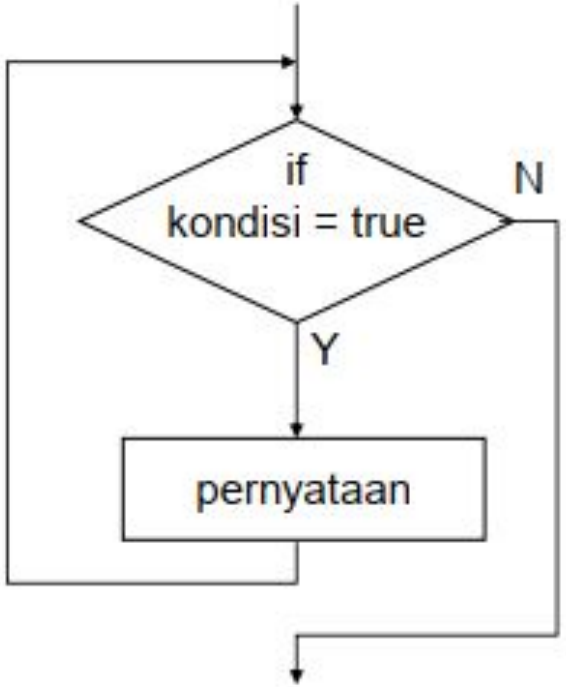
Contoh 1

```c
#include <stdio.h>

int main()
{
    int i = 10;
    while(i>=1){
        printf("%d\t", i);
        i -= 3;
    }
}
```

Result

```
$gcc -o main *.c
$main
10      7       4       1
```

Output iterasi ke-4



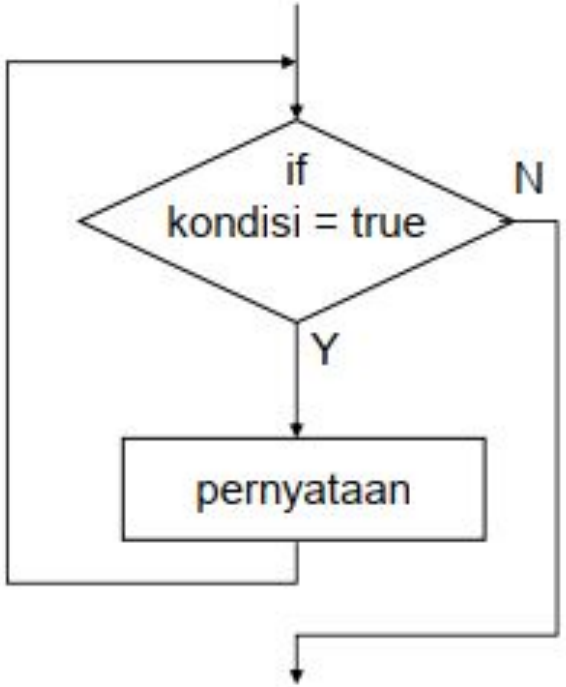
- Iterasi ke-4
  continue condition □ i >= 1 □ 1 >= 1 □ TRUE
  baris ke-7 dijalankan, sehingga muncul output “1 ”
  baris ke-8 dijalankan, sehingga nilai i berubah menjadi 1-3 = -2

Contoh 1

```c
#include <stdio.h>

int main()
{
    int i = 10;
    while(i>=1){
        printf("%d\t", i);
        i -= 3;
    }
}
```

Result

```
$gcc -o main *.c

$main
10      7       4       1
```



- Iterasi ke-5
  continue condition □ i >= 1 □ -2 >= 1 □ FALSE
  keluar dari *looping*, sehingga baris ke-7 dan ke-8 tidak dijalankan

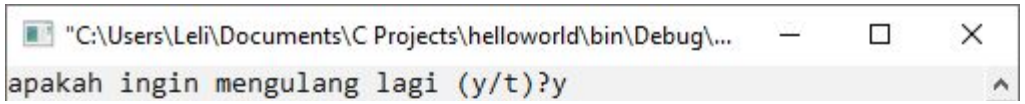
Contoh 2

```c
#include <stdio.h>

int main()
{
    char jawab = 'y';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
}
```

- *Continue condition*: `jawab == 'y'`
- Variabel yang dicek pada *continue condition* perlu diberi nilai awal, yakni `jawab = 'y'`
- *Statement* yang merubah nilai kondisi:
  input nilai untuk variabel `jawab` □ `scanf("%c", &jawab);`

Contoh 2

```c
#include <stdio.h>

int main()
{
    char jawab = 'y';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
}
```

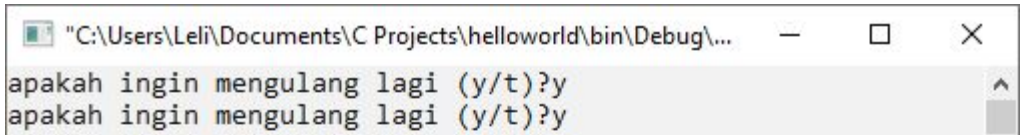


Output iterasi ke-1

- Iterasi ke-1
  jawab = ‘y’
  jawab == ‘y’ □ ‘y’ == ‘y’ □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “apakah ingin mengulang lagi(y/t)?”
  Baris ke-8 dijalankan, sehingga user bisa memasukkan karakter, yakni ‘y’
  Baris ke-9 dijalankan, sehingga input dari user dihapus dari input *buffer*

Contoh 2

```c
#include <stdio.h>

int main()
{
    char jawab = 'y';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
}
```

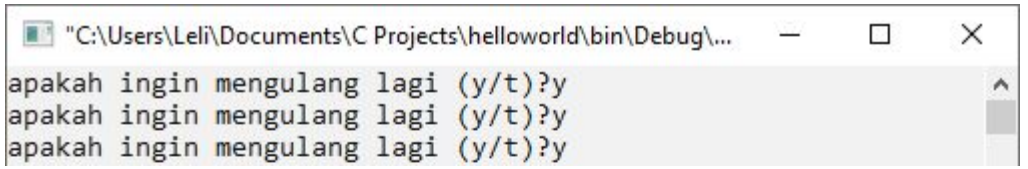


Output iterasi ke-2

- Iterasi ke-2
  jawab = ‘y’
  jawab == ‘y’ □ ‘y’ == ‘y’ □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “apakah ingin mengulang lagi(y/t)?”
  Baris ke-8 dijalankan, sehingga user bisa memasukkan karakter, yakni ‘y’
  Baris ke-9 dijalankan, sehingga input dari user dihapus dari input *buffer*

Contoh 2

```c
#include <stdio.h>

int main()
{
    char jawab = 'y';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
}
```

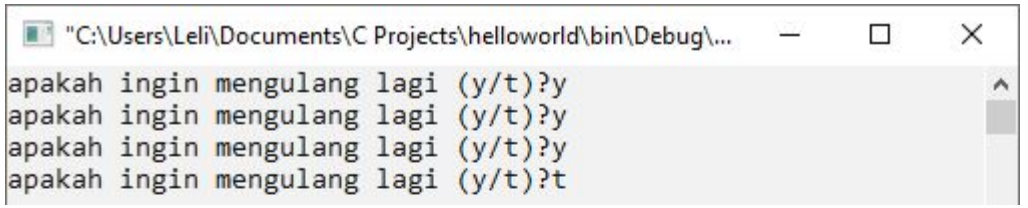


Output iterasi ke-3

- Iterasi ke-3
  jawab = ‘y’
  jawab == ‘y’ □ ‘y’ == ‘y’ □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “apakah ingin mengulang lagi(y/t)?”
  Baris ke-8 dijalankan, sehingga user bisa memasukkan karakter, yakni ‘y’
  Baris ke-9 dijalankan, sehingga input dari user dihapus dari input *buffer*

Contoh 2

```c
#include <stdio.h>

int main()
{
    char jawab = 'y';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
}
```



Output iterasi ke-4

- Iterasi ke-4
  jawab = ‘y’
  jawab == ‘y’ □ ‘y’ == ‘y’ □ TRUE
  Baris ke-7 dijalankan, sehingga muncul output ke layar “apakah ingin mengulang lagi(y/t)?”
  Baris ke-8 dijalankan, sehingga user bisa memasukkan karakter, yakni ‘t’
  Baris ke-9 dijalankan, sehingga input dari user dihapus dari input *buffer*

Contoh 2

```c
#include <stdio.h>

int main()
{
    char jawab = 'y';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
}
```

```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\...
apakah ingin mengulang lagi (y/t)?y
apakah ingin mengulang lagi (y/t)?y
apakah ingin mengulang lagi (y/t)?y
apakah ingin mengulang lagi (y/t)?t

Process returned 0 (0x0)   execution time : 5.808 s
Press any key to continue.
```

- Iterasi ke-5
  jawab = ‘t’
  jawab == ‘y’ □ ‘t’ == ‘y’ □ FALSE
  Karena *continue condition* tidak terpenuhi, maka *looping* berhenti. Sehingga baris ke-7 s.d ke-9 tidak lagi dijalankan.

Contoh 3

```c
#include <stdio.h>

int main()
{
    char jawab = 't';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
    printf("selesai\n");
}
```

- *Continue condition*: `jawab == 'y'`
- Variabel yang dicek pada *continue condition* perlu diberi nilai awal, yakni `jawab = 't'`
- *Statement* yang merubah nilai kondisi:
  input nilai untuk variabel `jawab` □ `scanf("%c", &jawab);`

Contoh 3

```c
#include <stdio.h>

int main()
{
    char jawab = 't';
    while(jawab == 'y'){
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }
    printf("selesai\n");
}
```

- Iterasi ke-1
  jawab = ‘t’
  jawab == ‘y’ □ ‘t’ == ‘y’ □ FALSE
  Karena *continue condition* tidak terpenuhi, maka *looping* berhenti. Sehingga baris ke-7 s.d ke-9 tidak dijalankan.

Statement `do-while`

Statement `do-while`

- Bentuk umum:

```
do{
  //body_of_loop
  statement;
  statement;
  …
  statement;
} while(continue_condition);
```

- `continue_condition`: kondisi yang harus terpenuhi agar *loop* tetap berjalan.
- Apabila hanya ada 1 *statement* yang berada di dalam *body of loop*, maka tidak perlu menggunakan kurung kurawal.
- Agar *loop* dapat berhenti, maka di dalam *body of loop* perlu ada *statement* yang bisa merubah nilai kondisi sehingga *loop* dapat berhenti.

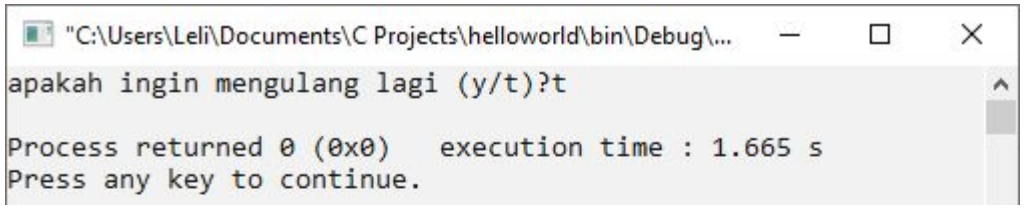
Contoh

```c
#include <stdio.h>

int main()
{
    char jawab;
    do{
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }while(jawab == 'y');
}
```

- *Continue condition*: `jawab == 'y'`
- *Statement* yang merubah nilai kondisi:
  input nilai untuk variabel `jawab` □ `scanf("%c", &jawab);`

Contoh

```c
#include <stdio.h>

int main()
{
    char jawab;
    do{
        printf("apakah ingin mengulang lagi (y/t)?");
        scanf("%c", &jawab);
        fflush(stdin);
    }while(jawab == 'y');
}
```



Output iterasi ke-1

- Iterasi ke-1
  Baris ke-7 dijalankan, sehingga muncul output ke layar “apakah ingin mengulang lagi(y/t)?”
  Baris ke-8 dijalankan, sehingga user bisa memasukkan karakter, yakni ‘t’
  Baris ke-9 dijalankan, sehingga input dari user dihapus dari input *buffer*
  Cek *continue condition* □ jawab == ‘y’ □ ‘t’ == ‘y’ □ FALSE
  Keluar dari *looping*

bridge to the future

http://www.eepis-its.edu