

Konsep Pemrograman

6. Perulangan Proses (*Looping*) - 2

Umi Sa'adah

Entin Martiana Kusumaningtyas

Tri Hadiah Muliawati

2021



Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Overview

- *Statement Nested Loop* (*loop* yang berada di dalam *loop* yang lain)
- *Statement* `break`
- *Statement* `continue`
- *Statement* `exit()`

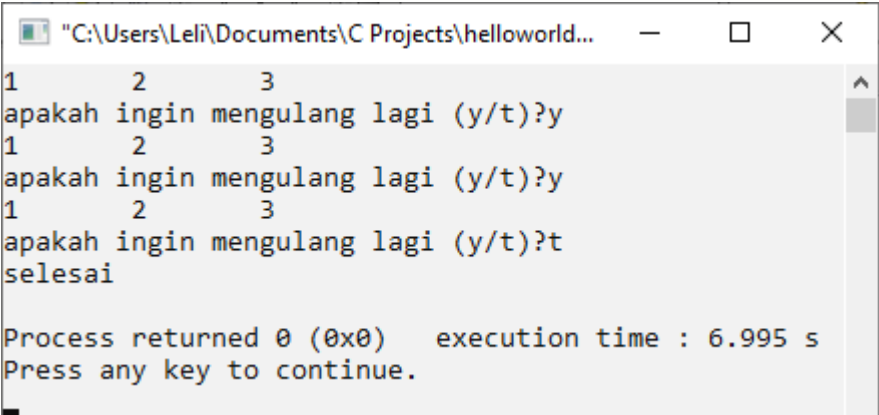
Nested Loop

Nested Loop

- Dalam suatu *loop* bisa terkandung *loop* yang lain. Hal ini disebut juga sebagai *nested loop*.
- Pada *nested loop*, *loop* luar disebut sebagai *outer loop*. Sedangkan, *loop* yang berada di dalam disebut sebagai *inner loop*.
- Ketiga *looping statement* (`for`, `while`, dan `do-while`) dapat digunakan untuk menyusun *nested loop*.

Contoh 1

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char jawab;
6      do{
7          int i;
8          for(i = 1; i <= 3;i++){
9              printf("%d\t", i);
10             }
11             printf("\n");
12             printf("apakah ingin mengulang lagi (y/t)?");
13             scanf("%c", &jawab);
14             fflush(stdin);
15         }while(jawab == 'y');
16         printf("selesai\n");
17     }
```



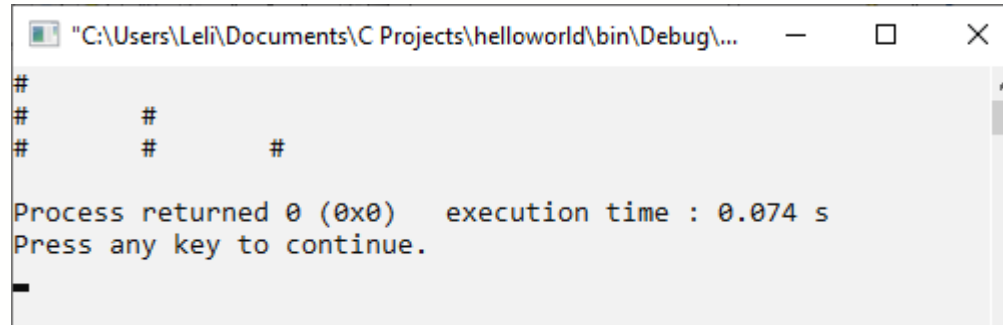
```
"C:\Users\Leli\Documents\C Projects\helloworld..."
1      2      3
apakah ingin mengulang lagi (y/t)?y
1      2      3
apakah ingin mengulang lagi (y/t)?y
1      2      3
apakah ingin mengulang lagi (y/t)?t
selesai

Process returned 0 (0x0)   execution time : 6.995 s
Press any key to continue.
```

- Inner loop menggunakan statement `for`
- Outer loop menggunakan statement `do-while`

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }
```

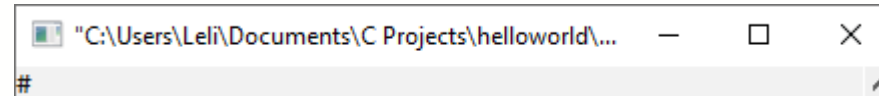


```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\..."
#
#  #
#
Process returned 0 (0x0)   execution time : 0.074 s
Press any key to continue.
```

Inner loop dan outer loop menggunakan statement `for`

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }
```



>>

Outer loop iterasi ke-1:

$j = 1$

$j \leq 3 \rightarrow 1 \leq 3 \rightarrow \text{TRUE}$

Baris ke-7 dijalankan.

Outer loop iterasi ke-1 dan *Inner loop* iterasi ke-1:

$i = 1$

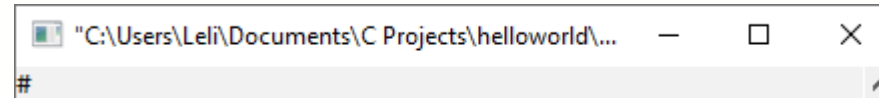
$i \leq j \rightarrow 1 \leq 1 \rightarrow \text{TRUE}$

Baris ke-8 dijalankan, sehingga muncul output ke layar "# "

$i++ \rightarrow i = 1+1 \rightarrow i = 2$

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }
```



Outer loop iterasi ke-1 dan *Inner loop* iterasi ke-2:

$i = 2$

$i \leq j \rightarrow 2 \leq 1 \rightarrow \text{FALSE}$

Keluar dari *inner loop*

Baris ke-10 dijalankan.

$j++ \rightarrow j = 1+1 = 2$

>>

Outer loop iterasi ke-2:

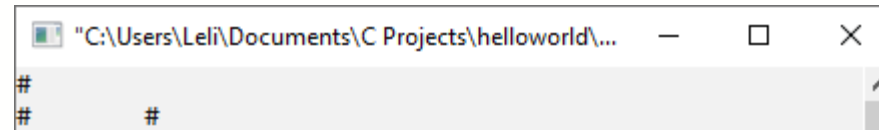
$j = 2$

$j \leq 3 \rightarrow 2 \leq 3 \rightarrow \text{TRUE}$

Baris ke-7 dijalankan.

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }
```



Outer loop iterasi ke-2 dan *Inner loop* iterasi ke-1:

$i = 1$

$i \leq j \rightarrow 1 \leq 2 \rightarrow \text{TRUE}$

Baris ke-8 dijalankan, sehingga muncul output ke layar "# "

$i++ \rightarrow i = 1+1 \rightarrow i = 2$

Outer loop iterasi ke-2 dan *Inner loop* iterasi ke-2:

$i = 2$

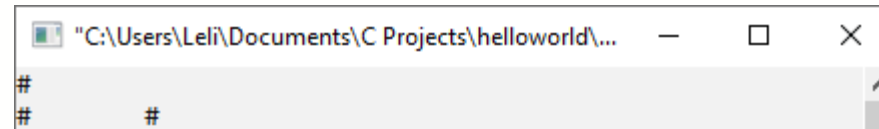
$i \leq j \rightarrow 2 \leq 2 \rightarrow \text{TRUE}$

Baris ke-8 dijalankan, sehingga muncul output ke layar "# "

$i++ \rightarrow i = 2+1 \rightarrow i = 3$

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }
```



Outer loop iterasi ke-2 dan *Inner loop* iterasi ke-3:

$i = 3$

$i \leq j \rightarrow 3 \leq 2 \rightarrow \text{FALSE}$

Keluar dari *inner loop*

Baris ke-10 dijalankan.

$j++ \rightarrow j = 2+1 = 3$

>>

Outer loop iterasi ke-3:

$j = 3$

$j \leq 3 \rightarrow 3 \leq 3 \rightarrow \text{TRUE}$

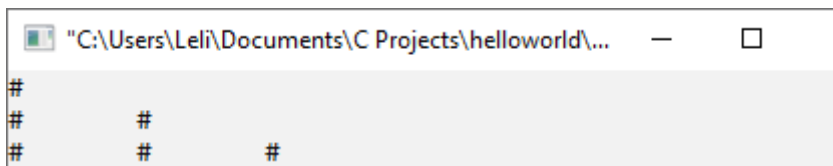
Baris ke-7 dijalankan.

Contoh 2

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }

```



```

"C:\Users\Leli\Documents\C Projects\helloworld\...
#
#  #
#  #  #

```

Outer loop iterasi ke-3 dan *Inner loop* iterasi ke-1:

$i = 1$

$i \leq j \rightarrow 1 \leq 3 \rightarrow \text{TRUE}$

Baris ke-8 dijalankan, sehingga muncul output ke layar "# "

$i++ \rightarrow i = 1+1 \rightarrow i = 2$

Outer loop iterasi ke-3 dan *Inner loop* iterasi ke-2:

$i = 2$

$i \leq j \rightarrow 2 \leq 3 \rightarrow \text{TRUE}$

Baris ke-8 dijalankan, sehingga muncul output ke layar "# "

$i++ \rightarrow i = 2+1 \rightarrow i = 3$

Outer loop iterasi ke-3 dan *Inner loop* iterasi ke-3:

$i = 3$

$i \leq j \rightarrow 3 \leq 3 \rightarrow \text{TRUE}$

Baris ke-8 dijalankan, sehingga muncul output ke layar "# "

$i++ \rightarrow i = 3+1 \rightarrow i = 4$

Outer loop iterasi ke-3 dan *Inner loop* iterasi ke-4:

$i = 4$

$i \leq j \rightarrow 4 \leq 3 \rightarrow \text{FALSE}$

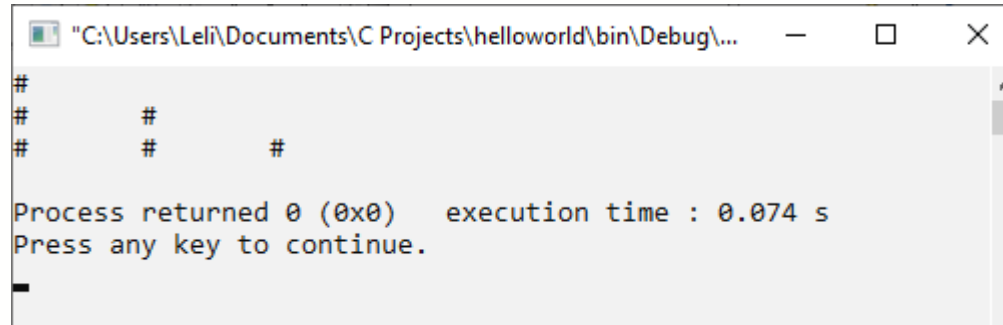
Keluar dari *inner loop*

Baris ke-10 dijalankan.

$j++ \rightarrow j = 3+1 = 4$

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,j;
6      for(j = 1; j <= 3; j++){
7          for(i = 1; i <= j; i++){
8              printf("#\t", i);
9          }
10         printf("\n");
11     }
12 }
```



>>

Outer loop iterasi ke-4

$j = 4$

$j \leq 3 \rightarrow 4 \leq 3 \rightarrow \text{FALSE}$

Keluar dari *outer loop*

Statement break

Statement break

- Pada `switch-case`, `break` digunakan untuk menuju ke akhir (keluar dari) struktur `switch-case`.
- Dalam *looping*, *statement* ini berfungsi untuk keluar secara 'paksa' dari *loop* `for`, `do-while` dan `while` apabila kondisi terpenuhi.
- Jika *statement* `break` berada dalam *loop* yang bertingkat (*nested loop*), maka pernyataan `break` hanya akan membuat proses keluar dari *loop* yang bersangkutan (tempat *statement* `break` dituliskan), bukan keluar dari semua *loop*



Contoh 1

```
1  #include <stdio.h>
2
3  int main()
4  {
5      while(1){
6          int bilangan;
7          printf("masukkan bilangan bulat: ");
8          scanf("%d", &bilangan);
9          if(bilangan%2==1) break;
10         printf("bilangan yang dimasukkan adalah %d\n", bilangan);
11     }
12 }
```

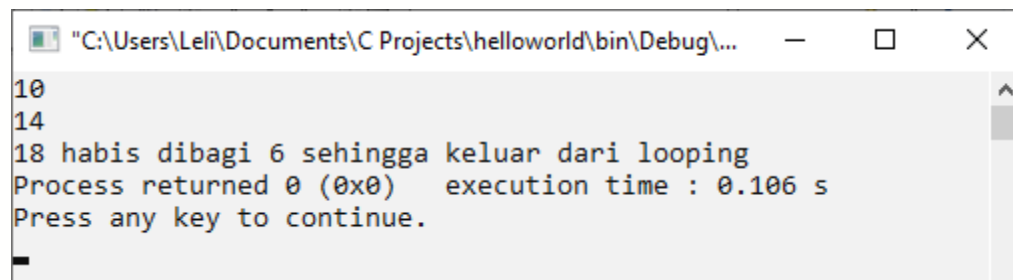


```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\..."
masukkan bilangan bulat: 10
bilangan yang dimasukkan adalah 10
masukkan bilangan bulat: 12
bilangan yang dimasukkan adalah 12
masukkan bilangan bulat: 13
Process returned 0 (0x0)   execution time : 6.983 s
Press any key to continue.
```

- Statement `break` digunakan pada baris ke-9.
- Apabila bilangan yang dimasukkan user adalah bilangan ganjil (`bilangan%2 == 1`), maka looping akan berhenti dan perintah baris ke-10 tidak lagi dieksekusi.

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      for(i = 10; i<=30; i+=4){
7          if(i%6 == 0) {
8              printf("%d habis dibagi 6 sehingga keluar dari looping", i);
9              break;
10         }
11         printf("%d\n", i);
12     }
13
14 }
```

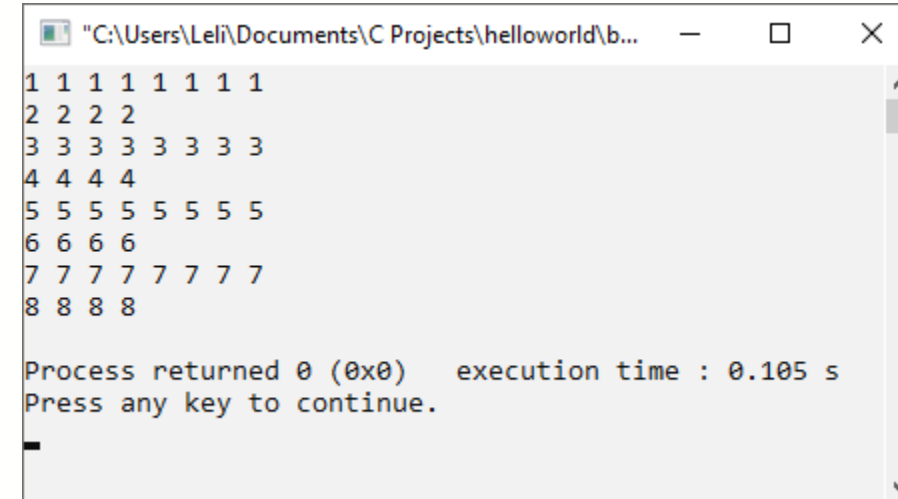


```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\..."
10
14
18 habis dibagi 6 sehingga keluar dari looping
Process returned 0 (0x0) execution time : 0.106 s
Press any key to continue.
```

- Statement `break` digunakan pada baris ke-9.
- Apabila variabel `i` habis dibagi 6 (`i%6 == 0`), maka looping akan berhenti dan perintah baris ke-11 tidak lagi dieksekusi.

Contoh 3

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i, j;
6      for(i = 1; i<=8; i++){
7          for(j = 1; j <= 8; j++){
8              if((i%2 == 0)&&(j > 4)) break;
9              printf("%d ", i);
10             }
11             printf("\n");
12         }
13
14 }
```



```
"C:\Users\Leli\Documents\C Projects\helloworld\b...  -  □  X
1 1 1 1 1 1 1 1
2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4
5 5 5 5 5 5 5 5
6 6 6 6
7 7 7 7 7 7 7 7
8 8 8 8
Process returned 0 (0x0)  execution time : 0.105 s
Press any key to continue.
-
```

- Statement `break` digunakan pada baris ke-8.
- Apabila variabel `i` habis dibagi 2 (`i%2 == 0`) dan variabel `j` lebih besar dari 4 (`j%>4`), maka *inner loop* akan berhenti dan program akan langsung menjalankan perintah baris ke-11.

Statement continue

Statement continue

- Digunakan untuk mengarahkan eksekusi ke iterasi berikutnya pada *loop* yang sama (*skip the current iteration, continue to the next iteration*)
- Pada *loop* `do-while` dan `while`, `continue` menyebabkan eksekusi menuju ke pengecekan *continue condition*.

```

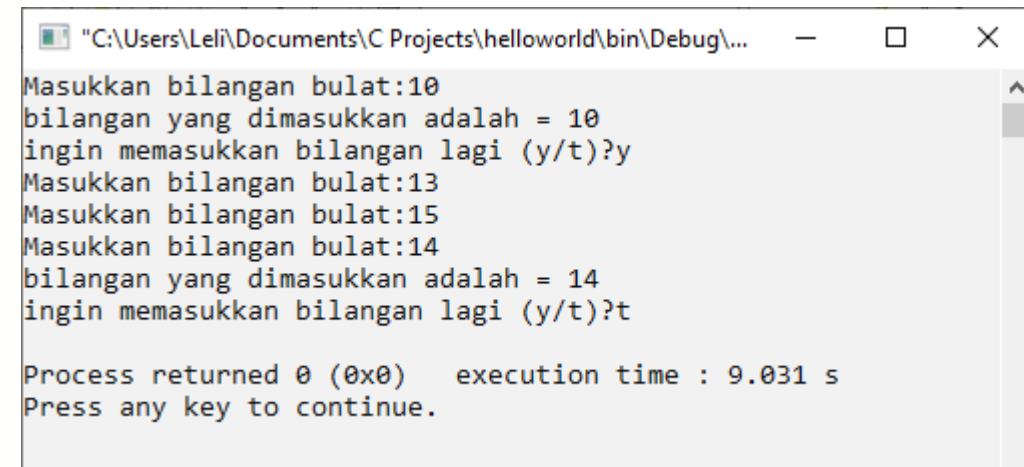
while(kondisi)
{ ...
  continue;
  ...
}

do
{ ...
  ...
  continue;
} while (kondisi)

```

Contoh 1

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int bilangan;
6      char jawab;
7      do{
8          printf("Masukkan bilangan bulat:");
9          scanf("%d", &bilangan);
10         if(bilangan%2 == 1) continue;
11         printf("bilangan yang dimasukkan adalah = %d\n", bilangan);
12         fflush(stdin);
13         printf("ingin memasukkan bilangan lagi (y/t)?");
14         scanf("%c", &jawab);
15     }while(jawab == 'y');
16
17 }
```

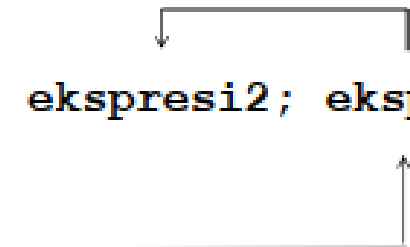


```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\..."
Masukkan bilangan bulat:10
bilangan yang dimasukkan adalah = 10
ingin memasukkan bilangan lagi (y/t)?y
Masukkan bilangan bulat:13
Masukkan bilangan bulat:15
Masukkan bilangan bulat:14
bilangan yang dimasukkan adalah = 14
ingin memasukkan bilangan lagi (y/t)?t
Process returned 0 (0x0)   execution time : 9.031 s
Press any key to continue.
```

- Statement `continue` digunakan pada baris ke-10.
- Apabila bilangan yang dimasukkan user adalah bilangan ganjil (`bilangan%2 == 1`), maka baris ke-11 s.d baris ke-14 tidak akan dieksekusi. Program akan langsung melakukan pengecekan *continue condition*. Karena tidak ada perubahan pada nilai variabel *control loop* (variabel `jawab`), maka *body of loop* akan kembali dijalankan.

Statement `continue`

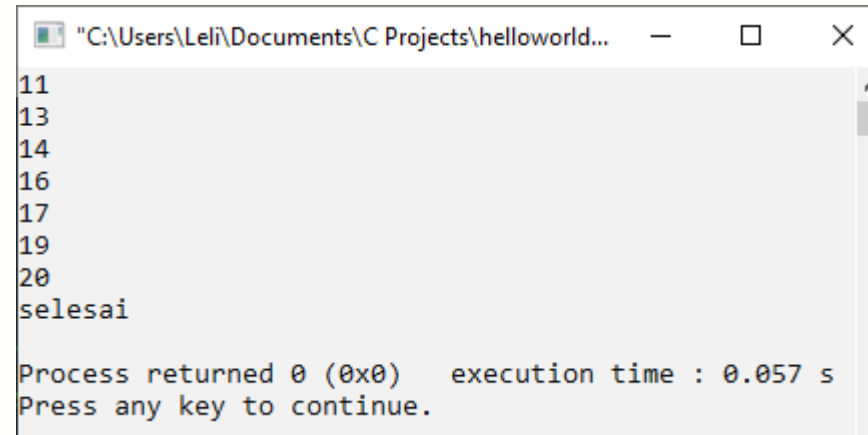
- Sedangkan, pada *loop* `for`, `continue` bagian perubahan nilai *control loop* (`ekspresi3`) dikerjakan, lalu dilanjutkan dengan pengecekan *continue condition* (`ekspresi2`).



```
for(ekspresi1; ekspresi2; ekspresi3)
{
    ...
    continue;
    ...
}
```

Contoh 2

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      for(i=11; i<=20;i++){
7          if(i%3==0) continue;
8          printf("%d\n", i);
9      }
10     printf("selesai\n");
11 }
```



```
11
13
14
16
17
19
20
selesai

Process returned 0 (0x0)   execution time : 0.057 s
Press any key to continue.
```

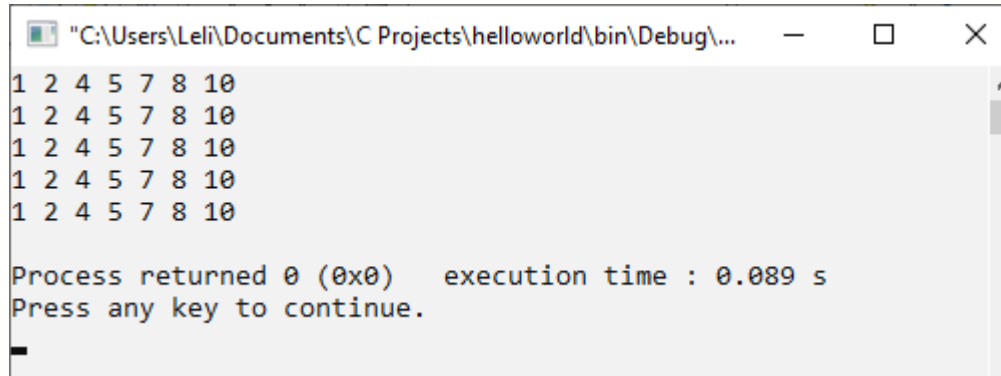
- Statement `continue` digunakan pada baris ke-7.
- Apabila nilai variabel `i` merupakan bilangan kelipatan 3 ($i \% 3 == 0$), maka baris ke-8 tidak akan dieksekusi. Program akan melakukan perubahan nilai variabel *control loop* (variabel `i` → `i++`) sebelum melakukan pengecekan ulang pada *continue condition* (`i <= 20`).
- Apabila *continue condition* terpenuhi, maka *body of loop* (baris ke-7 dan ke-8) akan dijalankan lagi.

Statement continue

- Jika `statement continue` berada dalam *loop* yang bertingkat (*nested loop*), maka pernyataan tersebut hanya akan mempengaruhi *loop* yang bersangkutan (tempat *statement continue* dituliskan), bukan semua *loop*.

Contoh 3

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i, j;
6      for(i = 1; i <= 5; i++){
7          for(j = 1; j <= 10; j++){
8              if(j%3 == 0) continue;
9              printf("%d ", j);
10             }
11             printf("\n");
12         }
13     }
```

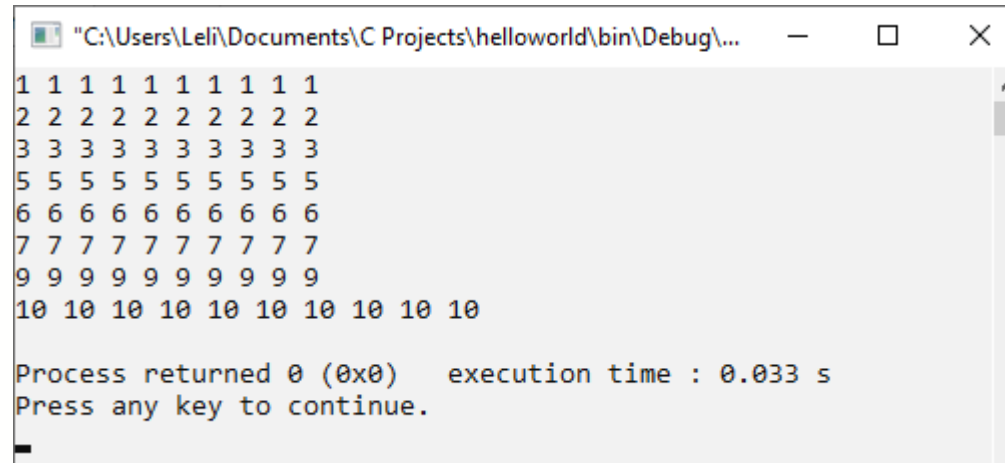


```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\...  -  □  X
1 2 4 5 7 8 10
1 2 4 5 7 8 10
1 2 4 5 7 8 10
1 2 4 5 7 8 10
1 2 4 5 7 8 10
Process returned 0 (0x0)  execution time : 0.089 s
Press any key to continue.
```

- Statement `continue` digunakan pada baris ke-8.
- Apabila nilai variabel `j` merupakan bilangan kelipatan 3 (`j % 3 == 0`), maka baris ke-9 tidak akan dieksekusi. Program akan melakukan perubahan nilai variabel *control* pada *inner loop* (variabel `j` → `j++`) sebelum melakukan pengecekan ulang pada *continue condition* (`j <= 10`).
- Apabila *continue condition* terpenuhi, maka *body of inner loop* (baris ke-8 dan ke-9) akan dijalankan lagi.

Contoh 4

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i, j;
6      for(i = 1; i <= 10; i++){
7          if(i%4 == 0) continue;
8          for(j = 1; j <= 10; j++){
9              printf("%d ", i);
10             }
11             printf("\n");
12         }
13     }
```



```
"C:\Users\Leli\Documents\C Projects\helloworld\bin\Debug\..."
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
10 10 10 10 10 10 10 10 10 10
Process returned 0 (0x0)   execution time : 0.033 s
Press any key to continue.
```

- Statement `continue` digunakan pada baris ke-7.
- Apabila nilai variabel `i` merupakan bilangan kelipatan 4 (`i%4 == 0`), maka baris ke-8 s.d baris ke-11 tidak akan dieksekusi. Program akan melakukan perubahan nilai variabel *control* pada *outer loop* (variabel `i` \rightarrow `i++`) sebelum melakukan pengecekan ulang pada *continue condition* (`i <= 10`).
- Apabila *continue condition* terpenuhi, maka *body of outer loop* (baris ke-7 s.d baris ke-11) akan dijalankan lagi.

Statement exit ()

Statement `exit()`

- Jika di dalam suatu eksekusi terdapat suatu kondisi yang tak dikehendaki, maka eksekusi program dapat dihentikan (secara normal) melalui pemanggilan fungsi `exit()`.
- Prototipe dari fungsi `exit()` didefinisikan pada file `stdlib.h`, yang memiliki deklarasi sebagai berikut :

```
void exit(int status);
```
- Menurut kebiasaan, nilai nol diberikan pada argument `exit()` untuk menunjukkan penghentian program yang normal → `exit(0);`

exit () vs break

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int i;
7      for(i = 1; i<= 10; i++){
8          if(i%4 == 0) exit(0);
9          printf("%d ", i);
10     }
11     printf("keluar dari looping");
12 }
```

Result

```
$gcc -o main *.c
$main
1 2 3
```

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      for(i = 1; i<= 10; i++){
7          if(i%4 == 0) break;
8          printf("%d ", i);
9      }
10     printf("keluar dari looping");
11 }
```

Result

```
$gcc -o main *.c
$main
1 2 3 keluar dari looping
```



bridge to the future

<http://www.eepis-its.edu>