Perulangan _ 3

Ira Prasetyaningrum

Perulangan do while

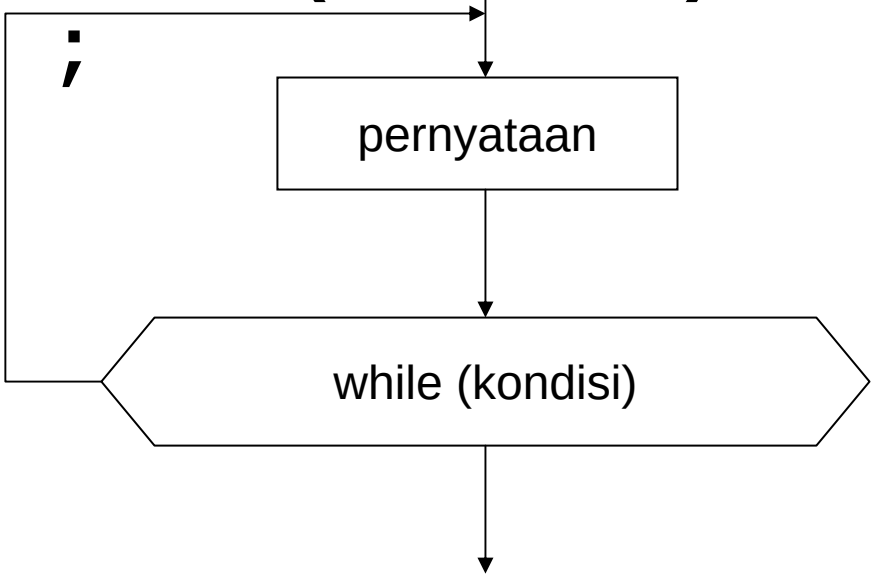
- Pernyataan *do while* memungkinkan perulangan berlanjut selama kondisi dalam while masih bernilai TRUE (non-zero).
- Formatnya :

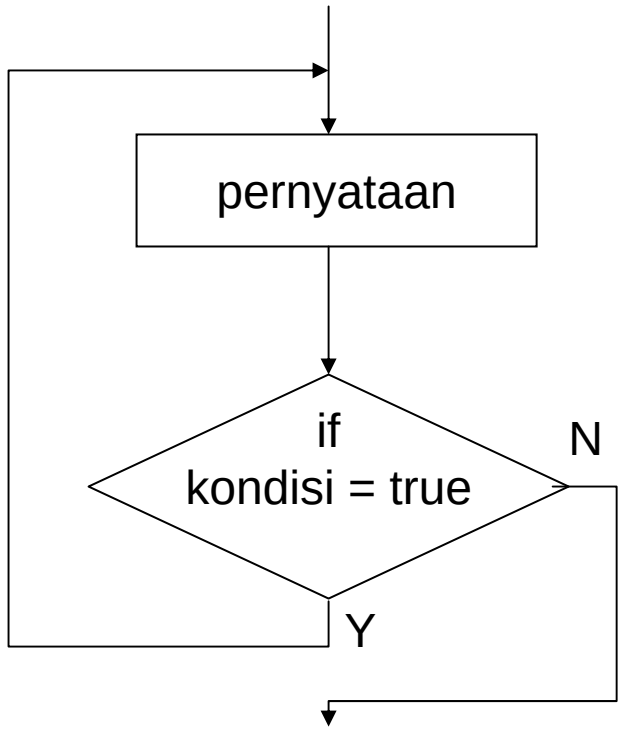
```
do {
    pernyataan;
} while(kondisi);
```

- The loop is executed at least once.

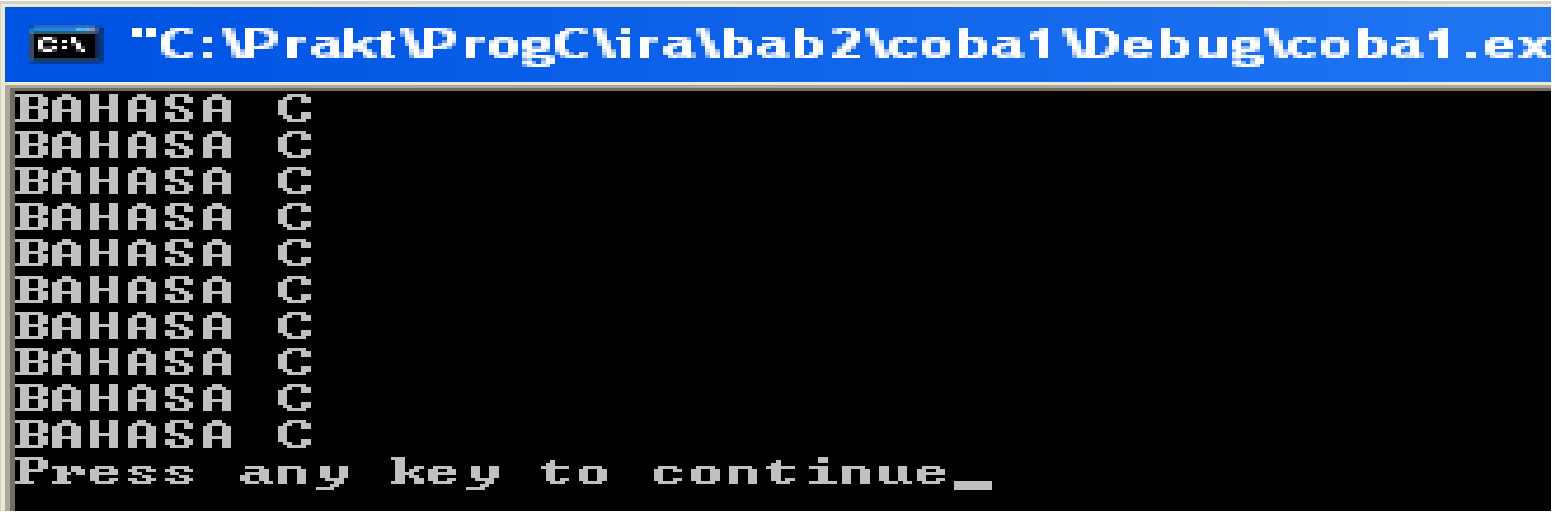
Diagram Alir Perulangan *do while*

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





```c
/* file program : intro1.c */
#include<stdio.h>
main()
{
        int i = 0;
        do
        {   printf("BAHASA C \n");
            i++;
        } while(i<10);
    }
```



Soal do while

- Gunakan loop *while* untuk membuat program yang dapat mencari total angka, min, max,rata-rata yang dimasukkan dengan tampilan sebagai berikut :
- Masukkan bilangan ke-1 : 5
- Mau memasukkan data lagi [y/t] ? y
- Masukkan bilangan ke-2 : 3
- Mau memasukkan data lagi [y/t] ? t
- Total bilangan = 8
- Min=3
- Max=5
- Rata-rata = 4

Nested loop

- Dalam suatu *loop* bisa terkandung *loop* yang lain.
- *Loop* yang terletak di dalam *loop* biasa disebut dengan *loop* di dalam *loop* (nested *loop*).
- Contoh :

```
for(eksp1; eksp2; eksp3)
   for(eksp1; eksp2; eksp3)
        pernyataan;
```

Loop Di Dalam Loop

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	4	6	8	10	12	14	16
3	3	6	9	12	15	18	21	24
4	4	8	12	16	20	24	28	32
5	5	10	15	20	25	30	35	40
6	6	12	18	24	30	36	42	48
7	7	14	21	28	35	42	49	56
8	8	16	24	32	40	48	56	64

Loop Di Dalam Loop

```c
/* file program : intro1.c */
#include<stdio.h>

#define MAKS 8
main()
{
     int baris;
     int kolom;
     int hasil_kali;
     for (baris = 1; baris <= MAKS; baris++)
     {
          for (kolom = 1; kolom <= MAKS; kolom++)
          {
               hasil_kali = baris * kolom;
               printf (" %2d ", hasil_kali);
          }
          printf("\n");      /* pindah baris */
     }
}
```

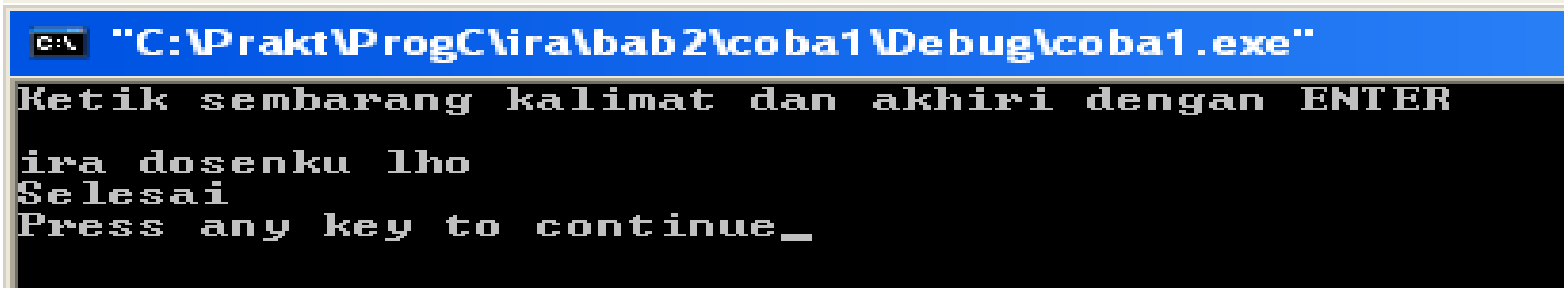
```
"C:\Prakt\ProgC\ira\bab2\coba1\Debug\coba1.exe"
  1    2    3    4    5    6    7    8
  2    4    6    8   10   12   14   16
  3    6    9   12   15   18   21   24
  4    8   12   16   20   24   28   32
  5   10   15   20   25   30   35   40
  6   12   18   24   30   36   42   48
  7   14   21   28   35   42   49   56
  8   16   24   32   40   48   56   64
Press any key to continue
```

Kondisional : *break*

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

Contoh penggunaan Break

```c
/* File program : tamat.c */
#include <stdio.h>
main()
{
     char kar;

     printf("Ketik sembarang kalimat");
     printf(" dan akhiri dengan ENTER\n\n");
for ( ; ; )
     {
          kar = getchar();
          if(kar == '\n')
               break;
     }
     printf("Selesai\n");
}
```



exit ( ) : Menghentikan Eksekusi Program

- 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 argumen *exit()* untuk menunjukkan penghentian program yang normal → **`exit(0);`**

Contoh penggunaan exit()

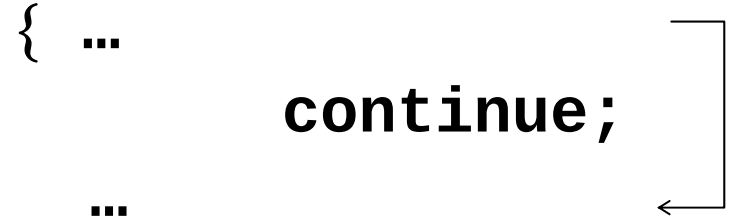
```c
/* File program : keluar.c */
#include <stdio.h>
#include <stdlib.h>
main()
{
     char kar;

     printf("Tekanlah X untuk menghentikan program.\n");
     for ( ; ;)
     {
          while ((kar = getchar()) == 'X')
               exit(0);
     }
}
```

```
"C:\Prakt\ProgC\ira\bab2\coba1\Debug\coba1.exe"
Tekanlah X untuk menghentikan program.
ui
X
x
c
f
X
Press any key to continue_
```

Kondisional : *continue*

- Digunakan untuk mengarahkan eksekusi ke iterasi (proses) berikutnya pada *loop* yang sama (*skip the current iteration, continue to the next iteration)*
- Pada do-while dan while, *continue* menyebabkan eksekusi menuju ke kondisi pengujian pengulangan sbb:

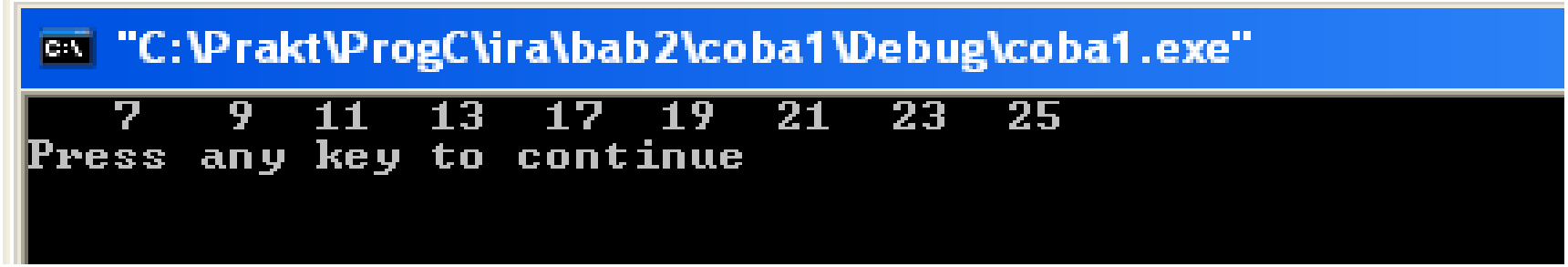
```
while(kondisi)
{ …
  continue;
  …
}
```



```
do
{ …
        continue;
  …
} while (kondisi)
```

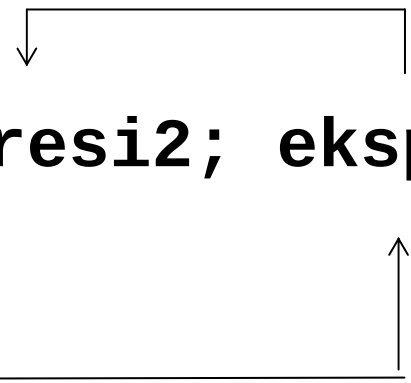
Contoh penggunaan continue

```c
/* File program : ganjil.c */
#include <stdio.h>
main()
{
     int x;
     for (x = 7; x <= 25; x += 2)
     {
          if   (x == 15)
               continue;
          printf("%4d", x);
     }
     printf("\n");
}
```



Kondisional : *continue*

- Pada loop for, *continue* menyebabkan bagian *control loop* (ekspresi3) dikerjakan dan kondisi untuk keluar dari loop for (ekspresi2) diuji kembali.



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

Soal 1

- Dengan menggunakan pernyataan *nested loop*, buatlah program berikut:
- input: n
- output:
- 1 2 3 4 5 … n
- 1 2 3 4 5 … n
- 1 2 3 4 5 … n n kali
- ……
- 1 2 3 4 5 … n

Soal 2

- Dengan menggunakan pernyataan *nested loop*, buatlah program berikut:
- input: n
- output:
- 1
- 2 2
- 3 3 3 n kali
- ……
- n n n n n … n

Soal 3

- Dengan menggunakan pernyataan *nested loop*, buatlah program berikut:
- input: n
- output:
- 2 3 5 7 11…. Bilangan prima ke n
-

Soal 4

- Dengan menggunakan pernyataan *nested loop*, buatlah program berikut:
- input: n
- output:
- 0 1 3 6 10 15 21 28 …. Bilangan ke n

Soal 5

- Pada akhir setiap 4 buah program diatas tambahkan tanyaan “apakah anda ingin keluar (y/t)?”, pertanyaan tersebut hanya bisa di jawab dengan huruf ‘y’ (y kecil) dan ‘t’(t kecil). Dan akan keluar dari program setelah dijawab dengan ‘y’ (y kecil)