**BAB V**

**WHILE, DO-WHILE**

1. **Struktur While**

Struktur perulangan *WHILE* adalah pengulangan yang melakukan pemeriksaan kondisi di awal blok struktur. Pengulangan hanya akan dilakukan jika kondisi yang didefinisikan di dalamnya terpenuhi (bernilai benar). Hal ini berarti jika kondisi yang didefinisikan tidak terpenuhi (bernilai salah) maka statemen-statemen yang terdapat dalam blok pengulangan pun tidak akan pernah dieksekusi oleh program.

Cek Kondisi?

while (true)

Blok kode yang diulang

Bentuk umumnya :

|  |
| --- |
| while (kondisi) { statemen\_statemen\_yang\_akan\_diulang; perubahan;  } |

Algoritma Program :

a. Sebuah **while** loop mengecek **kondisi**

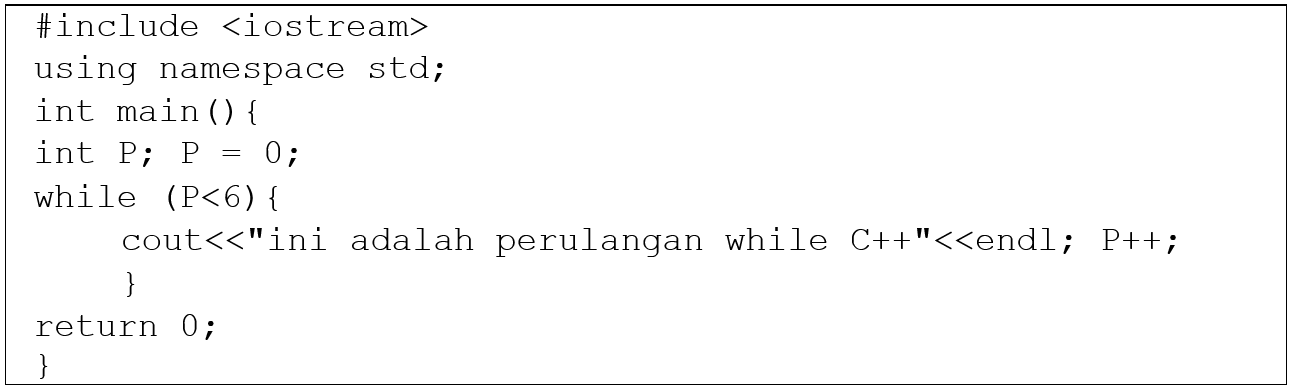
b. Selama **kondisi** yang dicek bernilai **true**, kode di dalam **while** loop di jalankan

c. **Kondisi** di cek lagi

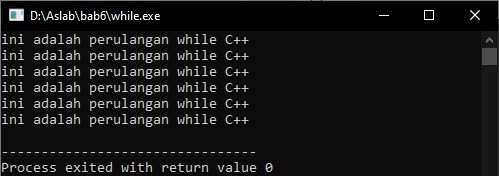
d. Proses ini berlanjut hingga saat **kondisi** bernilai **false**

e. Saat **kondisi** yang di cek bernilai **false**, perulangan berakhir

Contoh:



Tampilan :



Analisa :

Pada program diatas dimulai dengan variabel P = 0, kondisi P < 6, *output* “ini adalah perulangan while C++” kemudian P++ maka setelah kita ketahui bahwa P=0 lalu akan di cek dikondisi apakah 0 kurang dari sama dengan 6 ? jika ya maka akan mencetak seperti hasil tampilan dan melakukan perubahan yaitu P++ dan seterusnya sampai kondisi terpenuhi.

1. **Struktur Do-While**

Berbeda dengan struktur *while* yang melakukan pemeriksaan kondisi di awal blok perulangan, pada struktur *do*-*while* kondisi justru ditempatkan di bagian akhir. Hal ini tentu menyebabkan struktur pengulangan ini minimal akan **melakukan satu kali** proses walaupun kondisi yang didefinisikan tidak terpenuhi (bernilai salah).

Blok kode yang diulang

Cek Kondisi?

while (true)

Bentuk umum :

|  |
| --- |
| do {  statemen\_yang\_akan\_diulang;  perubahan;  } while (kondisi); |

Algoritma Program :

a. Kode didalam perulangan di jalankan

b. kemudian **kondisi** di cek

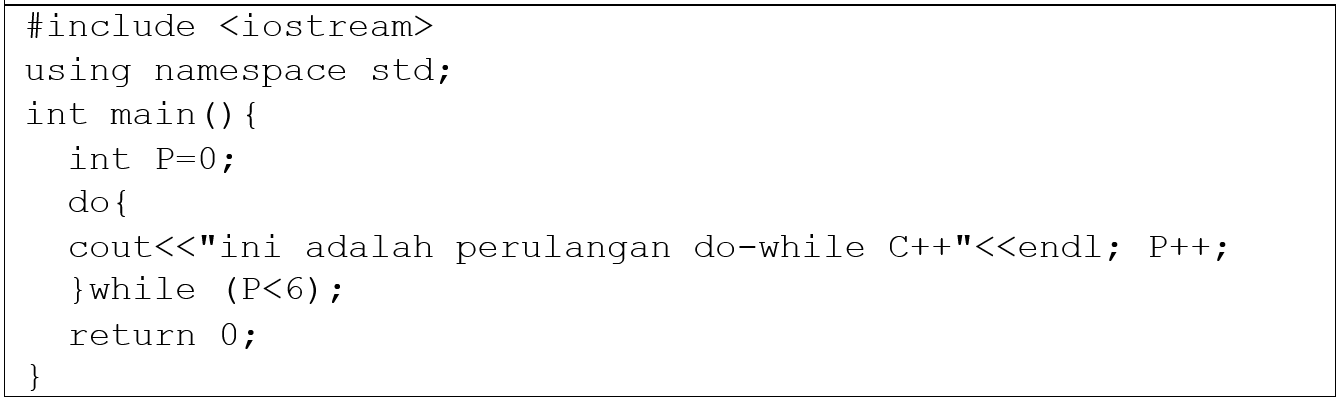
c. Selama **kondisi** yang dicek bernilai **true**, kode di dalam **do** di jalankan

d. **Kondisi** di cek lagi

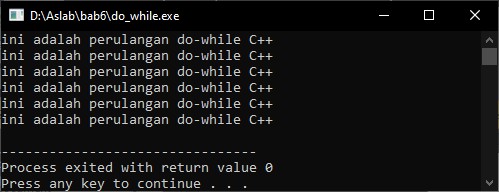
e. Proses ini berlanjut hingga saat **kondisi** bernilai **false**

f. Saat **kondisi** yang di cek bernilai **false**, perulangan berakhir

Contoh :



Tampilan :



Analisa :

Pada program diatas dimulai dengan variabel P = 0, kemudian program akan mengerjakan perintah output, lalu akan melakukan perubahan setelah itu dicek pada kondisi. Saat ini P = 1 maka akan melakukan perintah output kembali, ini akan dilakukan terus menerus sampai kondisi tidak terpenuhi lagi.



#taugaksih

1. Continue

**Jump Statement Continue**

Berbeda dengan break yang di gunakan untuk menghentikan proses loop, continue digunakan untuk melanjutkan proses loop

continue;

Contoh Program :

#include <iostream>

using namespace std;

int main(){

int a;

for(a=1;a<=10;a++){

if(a==2||a==5||a==7){

continue;

}

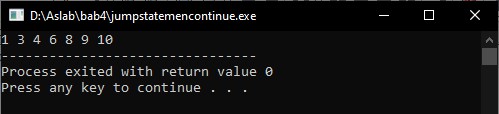
cout<<a<<" ";

}

return 0;

}

Tampilan Program :



Analisa :

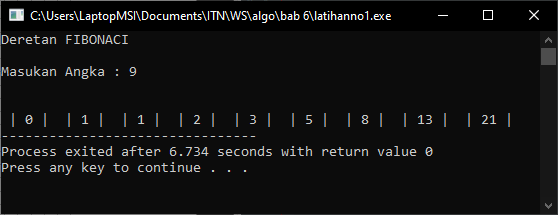
Continue digunakan untuk melanjutkan sebuah looping, seperti contoh perulangan diatas, apabila nilai a adalah 2,5,7, maka nilai tersebut akan di lewati sehingga data yang ditampilakan 1,3,4,6,8,9,10

**LATIHAN**

1. Buatlah program Barisan ***Fibonaci***, dengan panjang barisan Fibonaci merupakan Inputan User, seperti contoh program dibawah ini:

SYARAT KETENTUAN :

* Praktikum Kelas B,C,E : Menggunakan While;
* Praktikum Sesi A,D,F : Menggunakan Do-While;
* Atau sebaliknya tergantung Keinginan Pengajar.



1. Buatlah program dengan hasil seperti dibawah ini:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a. | 22 | 17 | 12 | 0 | -5 | -10 | -15 | -20 | |  |  |
| b. | F | I | L | O | R |  |  |  |  | |  |  |
| c. | Y |  |  |  |  |  |  |  |  | |  |  |
|  | T |  |  |  |  |  |  |  |  | |  |  |
|  | O |  |  |  |  |  |  |  |  | |  |  |
|  | J |  |  |  |  |  |  |  |  | |  |  |
|  | E |  |  |  |  |  |  |  |  | |  |  |
| d. | 4 | d | 5 | e | 7 | g | 10 | j | 14 | | n |  |
| e. | -17 | -11 | -5 | 7 | 13 |  |  |  |  | |  |  |
| f. | K | 4 | I | 5 | G | 7 | E | 10 | C | | 14 |  |