

Mata Kuliah : Dasar-dasar Pemrograman(Praktek)
Kode Mata Kuliah : KBTI4104
Jumlah SKS : 4 SKS
Nama Dosen : Ani Rahmani
Minggu ke : 6
Tanggal : 23 Oktober 2015
Jadwal : Jumat (08.10 – 11.30)

Mengulang Loop dan Array

Pada minggu ini, diulang kembali materi tentang loop dan array agar mahasiswa semakin mengerti.

Loop terbagi menjadi 3 macam, yaitu For Do, While Do, dan Repeat Until

Struktur For Do : pengulangan tanpa kondisi, jumlah pengulangan sudah diketahui sebelum eksekusi

Algoritma

Begin

```
| i : integer  
| for i  $\leftarrow$  1 to 5 do  
|   | write (layar) "Hello Word"  
| endfor
```

End

Bahasa C

```
#include <stdio.h>  
int main(){  
    int i;  
    for(i=1;i<=5;i++){  
        printf("Hello World");  
    }  
    return 0;
```

```
}
```

Dalam Algoritma tersebut akan ditampilkan kata “Hello World” sebanyak 5 kali. Sebelum memasuki loop, variable i diisi 1, lalu proses dalam loop dijalankan. setelah proses terakhir dalam loop dijalankan, i ditambah 1, lalu proses dalam loop dijalankan kembali, terus begitu sampai i bernilai 5, proses dalam loop dijalankan, lalu proses keluar dari loop.

Struktur While Do : pengulangan dengan kondisi. Jumlah pengulangan tidak diketahui. Sebelum eksekusi program. Yang dapat ditentukan hanyalah kondisi berhenti

Algoritma

Begin

```
| i : Integer
| kabisat : boolean
| i ← 2009
| kabisat ← .False.
| while (kabisat=.False.) do
|   | i ← i +1
|   | if (i mod 4 = 0)
|   |   | then kabisat ← .True.
|   | endif
| endwhile
| write (layar) i
```

End

Bahasa C

```

#include <stdio.h>

#include <stdbool.h>

int main(){

    int i;

    bool kabisat;

    i = 2009;

    kabisat = false

    while(kabisat == false){

        i = i + 1;

        if(i % 4 == 0){

            kabisat=true;

        }

    }

    printf("%d", i);

    return 0;

}

```

Dalam Algoritma tersebut, akan ditampillkan tahun kabisat selanjutnya setelah tahun 2009. Sebelum memasuki loop, kondisi akan dicek terlebih dahulu, jika kondisi bernilai .True., maka loop akan diproses, jika bernilai .False., maka loop tidak akan diproses. Setelah loop diproses dan proses terakhir dalam loop dijalankan, kondisi akan dicek kembali, bila kondisi masih bernilai .True., maka loop diproses kembali, tapi bila kondisi bernilai .False., maka proses akan keluar dari loop. Loop akan terus berulang selama kondisi bernilai .True., sampai kondisi bernilai .False. loop akan berhenti.

Yang perlu diperhatikan pada loop **While Do** ialah kondisi variable sebelum memasuki loop, kondisi yg dibutuhkan dalam loop, dan kondisi yang menyebabkan loop berakhir. Bila di dalam loop tidak ada assignment yang menyebabkan loop akan berakhir, maka akan terjadi *infinite loop*.

Struktur Repeat Until : sama seperti bentuk while do. Tetapi kondisi dicek setelah proses terakhir dalam loop dijalankan

Algoritma

Begin

```
| i : integer
| find : Boolean
| i ← 3
| find ← .False.
| Repeat
|   | i ← i + 1
|   | if (i mod 5 = 0)
|   |   | then find ← .True.
|   |   endif
| Until (find=.True.)
| write (layar) i
```

End

Dalam Bahasa C tidak terdapat struktur loop **Repeat Until**. Di Bahasa C terdapat loop **Do While** yang pengecekan kondisinya berada di akhir loop. Loop **Do While** akan terus berulang selama kondisinya bernilai .True.

Bahasa C

```
#include <stdio.h>

#include <stdbool.h>
```

```

int main(){

    int i;

    bool find;

    i = 3;

    find=false;

    do{

        i=i+1;

        if(i%5==0){

            find=true;

        }

    }while(find==false)

    printf("%d",i);

    return 0;

}

```

Algoritma diatas akan menampilkan bilangan kelipatan 5 setelah angka 3. Loop pertama akan diproses bagaimanapun kondisi awal variabelnya. Setelah proses terakhir dalam loop dijalankan, kondisi dicek. Bila kondisi bernilai .False. loop akan diproses kembali, bila bernilai .True. proses akan keluar dari loop. Loop akan terus berulang selama kondisi bernilai .False., sampai kondisi bernilai .True. loop berhenti.

Sama seperti Struktur While, yang perlu diperhatikan dalam loop **Repeat Until** adalah kondisi yang dibutuhkan dalam loop, dan kondisi yang menyebabkan loop berakhir. Bila di dalam loop tidak ada assignment yang menyebabkan loop akan berakhir, maka akan terjadi *infinite loop*.

Array 1 Dimensi

Array 1 Dimensi digunakan untuk mempermudah pengolahan data sejenis. Dibandingkan dengan tanpa Array, data lebih sulit diolah, cepat dieksekusi, dan tidak mudah crash.

A [1] ← index array
↑
nama array

Contoh :

a. Tanpa Array 1 Dimensi

Tentukan nilai maksimal dari 3 data!

Begin

```
| A,B,C,Maks : integer
| A ← 5
| B ← 4
| C ← 6
| if (A>B)
| | then if (A>C)
| | | then Maks ← A
| | | else Maks ← C
| | endif
| | else if (B>C)
| | | then Maks ← B
| | | else Maks ← C
| | endif
| endif
| write (layar) Maks
```

End

b. Dengan Array 1 Dimensi

A

5	4	6
---	---	---

Index Array A → **1 2 3**

Tentukan Maksimal dari Array A!

Algoritma

Begin

```
| i,Maks : integer
|  Maks ← A[1]
|  for i ← 2 to 3 do
|    | if (Maks<A[ i ])
|    |   then  Maks ← A[ i ]
|    |   endif
|  endfor
|  write (layar) Maks
```

End

Bahasa C

```
#include <stdio.h>
```

```
int main(){
```

```
    int Maks,i;
```

```
Maks=A[0];  
for(i=1;i<3;i++){  
    if(Maks<A[i]){  
        Maks=A[i];  
    }  
}  
printf("%d",Maks);  
return 0;  
}
```

Index array dalam bahasa C dimulai dari angka 0.

Dibandingkan dengan pengolahan data tanpa array, pengolahan data dengan array lebih efisien, karena bisa diakses menggunakan looping.