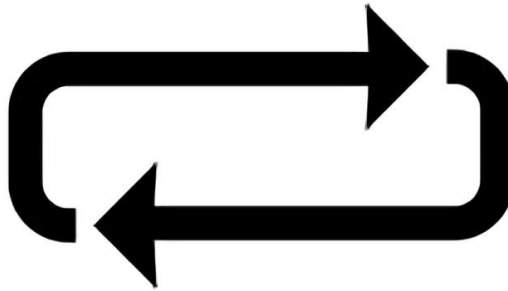


## Modul 5

### Perulangan I ( Loop I )



#### A. Tujuan

1. Memberikan pemahaman kepada para praktikan mengenai konsep perulangan atau loop,
2. Memberikan gambaran kepada para praktikan mengenai cara mengimplementasikan konsep perulangan pada sebuah program komputer dalam bahasa C.

#### B. Pendahuluan

Perulangan atau umumnya disebut dengan *Loop*, adalah sebuah teknik yang memungkinkan program untuk menjalankan serangkaian perintah secara berulang sampai suatu kondisi tertentu telah berhasil dicapai. Hal ini memungkinkan kita untuk menulis kode program yang lebih efisien dan lebih mudah dibaca karena tidak ada penulisan perintah-perintah yang sama berulang kali.

```
for(i = 0; i < 50; i++){  
    printf("DasPro");  
}
```

Gambar A.1 Contoh Perulangan

Salah satu contoh implementasi perulangan dapat dilihat dari Gambar A.1, dimana program akan menampilkan output kata “DasPro” sebanyak 50 kali. Dengan demikian, kode program akan menjadi lebih efisien daripada menulis `printf("DasPro");` sebanyak 50 kali secara manual.

## C. Kelompok Perulangan

### 1. Counted Loop (Perulangan Terhitung)

Perulangan yang sudah pasti dan sudah ditentukan sebelumnya berapa jumlah perulangan yang akan dijalankan oleh program.

### 2. Uncounted Loop (Perulangan Tidak Terhitung)

Perulangan yang akan terus berulang sampai kondisi tertentu terpenuhi sehingga tidak diketahui pasti seberapa banyak perulangan yang akan dijalankan program dari awal.

## D. Komponen Perulangan

### 1. Inisialisasi : Penentuan awal variabel kontrol.

Variabel kontrol adalah variabel yang digunakan sebagai pengatur/pengendali perulangan dan harus mulai dari suatu nilai tertentu.

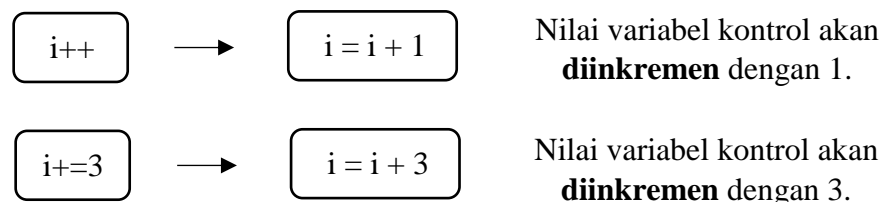
### 2. Kondisi : Ekspresi Boolean (Kondisi terhadap variabel kontrol berupa **true** / **false**)

Sebagai contoh, loop akan terus berjalan apabila kondisi masih true dan berhenti ketika kondisi bernilai false.

### 3. Inkremen/Dekremen : Peningkatan / Pengurangan nilai variabel kontrol secara bertahap. Pada setiap perulangan, variabel kontrol akan diinkremen maupun didekremen sesuai kebutuhan.

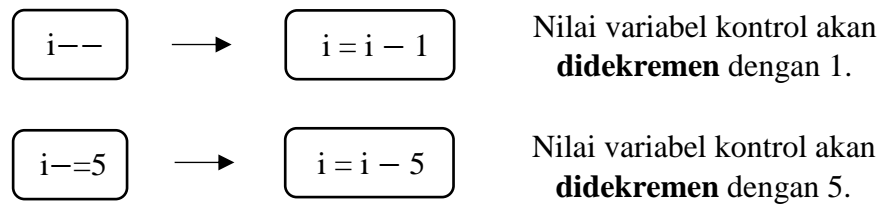
- **Inkremen** : Proses penambahan nilai pada variabel kontrol. Nantinya nilai ini akan terus bertambah selama perulangan masih berjalan.

Contoh :



- **Dekremen** : Proses pengurangan nilai pada variabel kontrol. Maka, nilai dari variabel kontrol akan berkurang selama perulangan masih berjalan.

Contoh :



4. **Statement** : Perintah yang akan diloop dalam suatu perulangan.
5. **Terminasi** : Sebuah perintah yang dapat mengakhiri suatu perulangan, atau menyatakan akhir dari suatu perulangan (**bersifat opsional**).

## E. Jenis Perulangan

### 1. For Loop (Counted Loop)

Perulangan “For” merupakan perulangan yang termasuk dalam **counted loop**, hal ini karena jumlah perulangan yang akan dijalankan program sudah ditentukan dari awal. Pertama-tama, inisialisasikan sebuah nilai terlebih dahulu, kemudian dilanjutkan dengan menulis kondisi perulangan. Lalu, tentukan apakah nilai inisialisasi akan diinkremen atau didekremen. Langkah terakhir adalah mengisi statement dengan aksi yang akan diulang-ulang oleh program dan diakhiri dengan terminasi (**opsional**).

Struktur For Loop :

```
for(inisialisasi; kondisi; inkremen/dekremen){
    statement
}
terminasi
```

Gambar E.1 Struktur For Loop

Contoh Code :

```
1      2      3
for(i = 0; i < 5; i++){
    printf("\n Perulangan yang ke-%d (i = %d)", i+1, i); 4
}
printf("\n\n Perulangan Selesai [!]); 5
```

Gambar E.2 Contoh For Loop

Alur Program :

- 1) [ 1 – 2 – 4 – 3 ] : Cek kondisi  $0 < 5$  (**True**), lalu inkremen variabel  $i$ .
- 2) [ 2 – 4 – 3 ] : Cek kondisi  $1 < 5$  (**True**), lalu inkremen variabel  $i$ .
- 3) [ 2 – 4 – 3 ] : Cek kondisi  $2 < 5$  (**True**), lalu inkremen variabel  $i$ .
- 4) [ 2 – 4 – 3 ] : Cek kondisi  $3 < 5$  (**True**), lalu inkremen variabel  $i$ .
- 5) [ 2 – 4 – 3 ] : Cek kondisi  $4 < 5$  (**True**), lalu inkremen variabel  $i$ .
- 6) [ 2 – 5 ] : Cek kondisi  $5 < 5$  (**False**), maka loop berhenti.

Output Program :

```
Perulangan yang ke-1 (i = 0)
Perulangan yang ke-2 (i = 1)
Perulangan yang ke-3 (i = 2)
Perulangan yang ke-4 (i = 3)
Perulangan yang ke-5 (i = 4)

Perulangan Selesai [!]
```

Gambar E.3 Output Program For Loop

## 2. While Loop (Uncounted Loop)

Perulangan “While” termasuk ke dalam **uncounted loop**. Hal ini karena belum diketahui berapa kali program akan menjalankan perulangan. Pada perulangan while, nilai inisialisasi / variabel kontrol akan dicek terlebih dahulu. Jika variabel kontrol tidak memenuhi kondisi (False), maka program akan langsung keluar dari loop tersebut. Dengan kata lain, selama variabel kontrol memenuhi kondisi (True), perulangan akan terus menerus dijalankan oleh program.

Struktur While Loop :

```
inisialisasi
while(kondisi){

    statement;
    statement;

    inkremen/dekremen;
}
terminasi;
```

Gambar E.4 Struktur While Loop

Contoh Code :

```
int userInput = 5; 1
printf("Hitung mundur dari : ");

while(userInput > 0){ 2
    printf(" %d ", userInput); 4

    userInput--; 3
}
printf("While loop selesai..."); 5
```

Gambar E.5 Contoh While Loop

Alur Program :

- 1) [ 1 – 2 – 4 – 3 ] : Cek kondisi  $5 > 0$  (**True**), lalu dekremen variabel *userInput*.
- 2) [ 2 – 4 – 3 ] : Cek kondisi  $4 > 0$  (**True**), lalu dekremen variabel *userInput*.
- 3) [ 2 – 4 – 3 ] : Cek kondisi  $3 > 0$  (**True**), lalu dekremen variabel *userInput*.
- 4) [ 2 – 4 – 3 ] : Cek kondisi  $2 > 0$  (**True**), lalu dekremen variabel *userInput*.
- 5) [ 2 – 4 – 3 ] : Cek kondisi  $1 > 0$  (**True**), lalu dekremen variabel *userInput*.
- 6) [ 2 – 5 ] : Cek kondisi  $0 > 0$  (**False**), maka loop berhenti.

Output Program :

```
Hitung mundur dari : 5 4 3 2 1
While loop selesai...
```

Gambar E.6 Output Program While Loop

### 3. Do While Loop (Uncounted Loop)

Sama halnya dengan perulangan while, perulangan “Do While” juga termasuk dalam **uncounted loop** untuk alasan yang sama. Jumlah perulangan yang akan dijalankan oleh program sama sekali tidak diketahui dari awal. Perulangan do while akan menjalankan statement terlebih dahulu lalu mengecek kondisi di setiap akhir perulangan. Jika kondisi tidak terpenuhi (False), maka program akan keluar dari loop tetapi jika kondisi terpenuhi (True) maka program akan terus menjalankan perulangan. Perlu diingat bahwa apapun hasil pengecekan kondisi pada do while loop, program akan tetap menjalankan statement/code minimal 1 kali!

Struktur Do While Loop :

```
inisialisasi
do{
    statement;
    statement;

    inkremen/dekremen;
}while(kondisi);
terminasi
```

Gambar E.7 Struktur Do While Loop

Contoh Code :

```
int userInput = 7; 1
do{
    printf("\nDo while"); 4

    userInput-=3; 3
}while(userInput > 0); 2
printf("\nDo While loop selesai..."); 5
```

Gambar E.8 Contoh Do While Loop

Alur Program :

1) [ 1 – 4 – 3 – 2 ]

: Menjalankan *printf*, dekremen variabel *userInput*, lalu cek kondisi  $4 > 0$  (**True**).

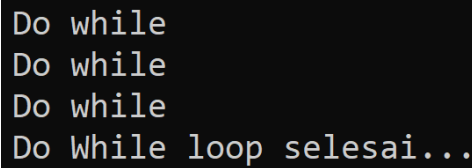
2) [ 4 – 3 – 2 ]

: Menjalankan *printf*, dekremen variabel *userInput*, lalu cek kondisi  $1 > 0$  (**True**).

3) [ 4 – 3 – 2 – 5 ]

: Menjalankan *printf*, dekremen variabel *userInput*, cek kondisi  $-2 > 0$  (**False**), kemudian loop berhenti.

Output Program :



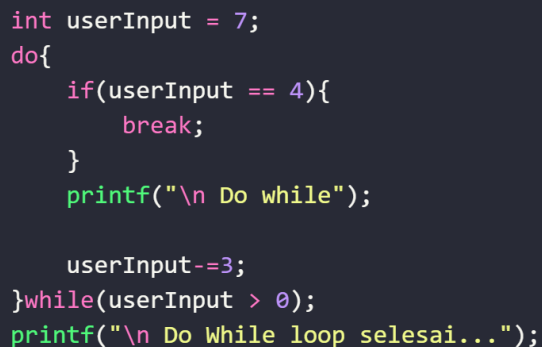
```
Do while
Do while
Do while
Do While loop selesai...
```

Gambar E.9 Output Program Do While Loop

## F. QnA (Question and Answer)

1. Apakah loop bisa dihentikan secara paksa ?

Jawaban : Tentu saja bisa! Dengan sintaks ***break***, program akan langsung keluar dari loop secara paksa. Di bawah ini merupakan contoh penghentian loop ketika inputan user bernilai 4.



```
int userInput = 7;
do{
    if(userInput == 4){
        break;
    }
    printf("\n Do while");

    userInput-=3;
}while(userInput > 0);
printf("\n Do While loop selesai...");
```

Gambar F.1 Contoh Penggunaan "Break"

Dengan menggunakan sintaks ***break*** pada gambar F.1, program akan otomatis berhenti ketika inputan dari user / variabel kontrol bernilai 4. Dengan demikian, output program akan terlihat seperti :

```
Do while  
Do While loop selesai...
```

Gambar F.2 Output Program dengan  
"Break"

Program hanya menampilkan "Do while" sebanyak 1 kali karena pada saat inputan user / variabel kontrol bernilai 4, program akan keluar dari perulangan secara paksa.

2. Apakah inisialisasi / variabel kontrol yang ada di dalam perulangan harus menggunakan *i* ?

Jawaban : Tidak, pemilihan penamaan variabel kontrol sangat bebas dan sepenuhnya terserah kita. Walau demikian, diusahakan nama tersebut tidak membingungkan untuk mempermudah kita dalam membuat program dan juga memudahkan orang lain dalam memahami code yang telah kita buat.



## GUIDED

Buatlah sebuah program yang berisikan 4 menu, masing-masing dengan fungsi dan tujuannya sendiri. Berikut ini adalah menu-menu yang perlu dibuat :

1. Membuat program untuk menginputkan nama dan umur, lalu ditampilkan.
2. Menghitung nilai total dan rata-rata dari angka inputan user
3. Menampilkan angka acak sebanyak 5 kali dari angka 0 - 100
4. Menghitung deret aritmatika dengan batas berdasarkan inputan user

### #Note

1. Komentar tidak perlu ditulis (hanya sebagai penjelasan)
2. Menu 1-3 sudah tersedia dalam guided.
3. **Buatlah** kode jawaban dari menu 4 sesuai soal.

Output Kode Menu 4:

```
<< GUIDED LOOP 1 >>
[1]. Input Data Diri
[2]. Hitung Rata-Rata
[3]. Tampil 5 Angka Random
[4]. Hitung Deret Aritmatika [TUGAS GUIDED]
[0]. Keluar Program

>>> 4

Masukkan Batas Angka : 10

Jumlah akumulatif dari angka 1 - 10 = 55
```

Gambar G.1 Output Program pada Menu 4

Di awal program, user akan diminta untuk menginputkan nilai yang akan digunakan sebagai batas akhir dari deret. Berdasarkan gambar G.1, kita ambil angka 10 sebagai batas akhir deret dan angka 1 yang akan selalu menjadi nilai awal deret. Lalu program akan menghitung deret aritmatika dari angka 1 sampai 10.

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55$$

Dengan demikian, program akan menampilkan angka 55 sebagai hasil akumulatif dari angka 1 sampai angka 10.

(**Dilarang** menggunakan rumus deret aritmatika untuk menemukan hasil akhir, tetapi gunakanlah konsep perulangan yang sudah diajarkan sebelumnya)

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h> //library penting dalam menggunakan fungsi rand()

typedef char string[50];

int main() {

    int menu, umur, banyakAngka;
    int i, angkaRandom;
    float nilai, total, rataRata;

    string nama;

    srand(time(NULL)); //berfungsi untuk menghasilkan serangkaian bilangan bulat pseudo-acak

    do{
        system("cls"); // ini adalah perintah clear screen
        printf("\n\t << GUIDED LOOP 1 >>");
        printf("\n [1]. Input Data Diri");
        printf("\n [2]. Hitung Rata-Rata");
        printf("\n [3]. Tampil 5 Angka Random");
        printf("\n [4]. Hitung Deret Aritmatika [TUGAS GUIDED]");
        printf("\n [0]. Keluar Program");
        printf("\n\n >>> ");scanf("%d", &menu);

        switch(menu){
            case 1:
                do{
                    printf(" Masukkan nama : ");fflush(stdin);gets(nama);
                }while(strlen(nama) == 0 || strcmp(nama, "-")==0); // meminta inputan ulang jika inputan
                                                                    // kosong atau sama dengan '-'

                do{
                    printf("\n Masukkan umur : ");scanf("%d", &umur);
                }while(umur < 1); //meminta inputan ulang jika angka yang diinputkan kurang dari 1

                printf("\n\n\t [ Data Diri ]");
                printf("\n Nama : %s", nama);
                printf("\n Umur : %d", umur);
                break;

            case 2:
                printf("\n Masukkan banyak data yang akan dihitung : "); scanf("%d", &banyakAngka);

                for(i=0; i<banyakAngka; i++){
                    printf("\n Masukkan angka ke-%d : ", i+1); scanf("%f", &nilai); // meminta inputan sebanyak jumlah angka
                    total = total + nilai; //menghitung akumulasi nilai total
                }

                rataRata = total / banyakAngka; //menghitung rata-rata
                printf("\n Jumlah dari seluruh data adalah %.0f", total);
                printf("\n Rata-Rata dari seluruh data tersebut adalah %.2f", rataRata);
                break;

            case 3:
                for(i = 0; i < 5; i++){
                    angkaRandom = (rand() % (100 - 0 + 1)) + 0; // format (rand() % (angkaMax - angkaMin + 1)) + angkaMin
                    printf("\n Angka Random : %d", angkaRandom);
                }
                break;

            case 4:
                // Buatlah kode untuk menu 4
                break;

            case 0:
                printf("\n\t [NAMA - KELAS - NPM]"); // Contoh: [Pieter Leviano - A - 220711653]
                break;

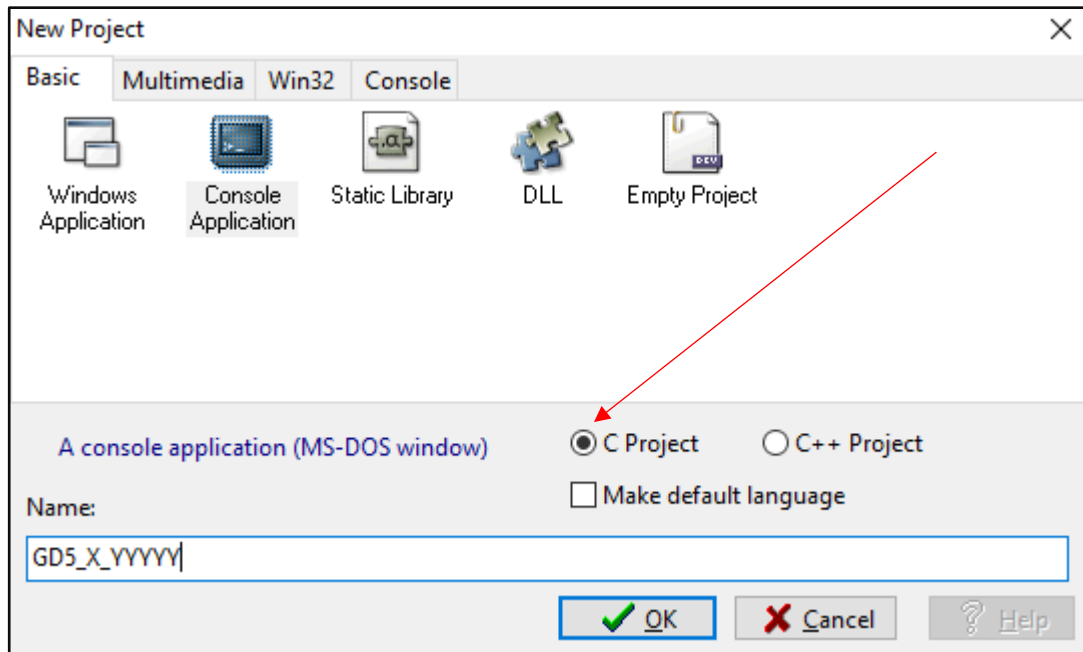
            default:
                printf("\n\t Menu Tidak Ada [!]");
        }
        getch(); /* getch(); adalah fungsi yang digunakan untuk membaca data karakter,
                    jika kita menekan character di keyboard maka tidak memberikan efek apapun */
    }while(menu != 0);

    return 0;
}

```

Gambar G.2 Kode Program Guided Loop 1

## Ketentuan Pengerjaan :



- Pastikan kalian membaca modul sebelum mengerjakan Guided
- Ekstensi program file harus **.c bukan .cpp**
- Guided dikerjakan dalam folder terpisah lalu di-zip

## Format Penamaan :

- **GD5\_X\_YYYYY.zip**
- X = Kelas
- YYYYYY = 5 digit NPM akhir

Jika ada yang mau ditanyakan atau masih bingung, bisa menghubungi saya melalui teams atau whatsapp yaa teman-teman

-Pieter Leviano

Tetap Semangat!