PRAKTIKUM 6
PENGULANGAN PROSES 2

A. Tujuan

1. Menjelaskan loop di dalam loop (nested loop) dan contoh kasusnya
2. Menjelaskan penggunaan pernyataan break
3. Menjelaskan penggunaan pernyataan continue
4. Menjelaskan penggunaan pernyataan goto
5. Menjelaskan penggunaan exit() untuk menghentikan eksekusi program dan contoh kasusnya

B. DASAR TEORI

Loop didalam Loop

Dalam suatu *loop* bisa terkandung *loop* yang lain. *Loop* yang terletak di dalam *loop* biasa disebut dengan *loop* di dalam *loop* (nested *loop*). Salah satu contoh nested *loop* misalnya pada permasalahan untuk membuat tabel perkalian:

	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

Implementasi dalam program selengkapnya adalah sebagai berikut :

```
/* File program : tblkali.c
 Loop for bersarang untuk membuat tabel perkalian */

#include <stdio.h>

#define MAKS 8

main()
{
  int baris, kolom, 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 */
  }
}
```

Bagian yang terletak dalam bingkai di depan dapat dapat diperoleh melalui

```
for (baris = 1; baris <= MAKS; baris++)
{
   hasil_kali = baris * kolom;
   printf ("%2d", hasil_kali);
}
```

dengan **MAKS** didefinisikan bernilai 8. Bagian *loop* yang terdalam :

```
for (kolom  = 1; kolom <= MAKS; kolom++)
{
   hasil_kali = baris * kolom;
   printf ("%2d", hasil_kali);
}
```

digunakan untuk mencetak suatu deret hasil perkalian dalam satu baris. Untuk berpindah ke baris berikutnya, pernyataan yang digunakan yaitu

```
printf("\n");
```

Adapun pencetakan untuk semua baris dikendalikan melalui

```
for (baris = 1; baris <= MAKS; baris++)
```

Pernyataan di atas mempunyai arti "dari baris ke-1 sampai dengan baris ke-MAKS".

Pernyataan break

Pernyataan break sesungguhnya telah diperkenalkan pada pernyataan *switch*. Pernyataan ini berfungsi untuk keluar dari *loop for*, *do-while* dan *while*. Sedangkan pada *switch* yaitu untuk menuju ke akhir (keluar dari) struktur *switch*. Sebagai contoh dapat dilihat pada gambar 7.1. Kalau pernyataan *break* dijalankan maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir tubuh *loop for*.

```
for ( ; ; )
{
       .
       .
  if ( …… )
    break;
       .
       .
}                    /* akhir tubuh loop for */

puts("\nSelesai…");
```

Gambar 7.1 Ilustrasi pengaruh break

Pada contoh potongan program berikut, pembacaan dan penampilan terhadap tombol yang ditekan akan berakhir kalau tombol yang ditekan adalah ENTER ('\n'). Pernyataan yang digunakan untuk keperluan ini :

```
if (kar == '\n')
    break;   /* keluar dari loop for */
```

Yang menyatakan "Jika tombol yang ditekan berupa ENTER, maka keluarlah dari *loop for*". Untuk lebih jelasnya, perhatikan program di bawah ini.

```
/* File program : tamat.c
Pemakaian break untuk keluar dari looping */

#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");
}
```

Contoh eksekusi :

```
Ketik sembarang kalimat dan akhiri dengan ENTER :

Menulis apa saja
Selesai
```

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.

Pernyataan Continue

Pernyataan *continue* digunakan untuk mengarahkan eksekusi ke iterasi (proses) berikutnya pada *loop* yang sama. Pada *do-while* dan *while*, pernyataan *continue* menyebabkan eksekusi menuju ke kondisi pengujian pengulangan, seperti yang dilukiskan pada Gambar 7.2. Pada *loop for*, pernyataan contunue menyebabkan bagian penaik variabel pengendali *loop* dikerjakan (ungkapan3 pada struktur *for*) dan kondisi untuk keluar dari *loop for* (ungkapan2 pada struktur *for*) diuji kembali.

Program ini digunakan untuk memasukkan data harus diulangi dan hal ini dikendalikan dengan *continue*. Untuk mengakhiri pemasukan data, data yang dimasukkan harus bernilai kurang dari 0. Perlu diketahui kondisi bernilai 1.

Gambar 7.2 Pengaruh *continue* pada *while* dan *do-while*

Menyatakan bahwa kondisi selalu dianggap benar. Untuk keluar dari *loop*, pernyataan yang digunakan berupa *break*.

Pengaruh *continue* pada *loop for* diperlihatkan pada dibawah ini. Program ini dipakai untuk menampilkan bilangan ganjil yang terletak antara 7 sampai dengan 25, kecuali 15.

```
/* File program : ganjil.c
menampilkan bilangan ganjil antara 7 - 25 kecuali 15 */

#include <stdio.h>

main()
{
    int x;

    for (x = 7; x <= 25; x += 2)
    {
      if  (x == 15)
         continue;
      printf("%4d", x);
    }
    printf("\n");
}
```

Contoh eksekusi :

```
  7   9  11  13  17  19  21  23  25
```

Pada program di atas, untuk menghindari agar nilai 15 tidak ditampilkan ke layar, pernyataan yang digunakan berupa

```
if ( x == 15)
     continue;
```

Artinya, jika kondisi `x == 15` bernilai benar, pernyataan *continue* menyebabkan pernyataan sisanya yaitu

```
printf("%d",x);
```

diabaikan dan eksekusi diarahkan kepada ungkapan :

```
x += 2
```

dan kemudian menguji kondisi :

```
x <= 25
```

Pada program di atas, pernyataan :

```
for (x = 7; x <= 25; x += 2)
{
  if  (x == 15)
     continue;
  printf("%4d", x);
}
```

dapat ditulis dalam bentuk lain sebagai berikut :

```
for (x = 7; x <= 25; x += 2)
    if (x != 15)
       printf("%4d", x);
```

Pernyataan *goto*

Pernyataan *goto* merupakan intruksi untuk mengarahkan eksekusi ke pernyataan yang diawali dengan suatu label. Label sendiri berupa suatu pengenal (*identifier*) yang diikuti dengan tanda titik dua (:)

Contoh pemakaian *goto* ditujukan pada program dibawah ini:

Pernyataan

```
goto cetak;
```

Mengisyaratkan agar eksekusi dilanjutkan ke pernyataan yang diawali dengan label

```
cetak:
```

Pernyataan

```
if (++pencacah <= 10)
     goto cetak;
```

Mempunyai arti :

- Naikkan nilai **pencacah** sebesar 1
- Kemudian, jika **pencacah** kurang dari atau sama dengan 10 maka eksekusi menuju ke label **cetak.**

Penerapan *goto* biasanya dilakukan pada *loop* di dalam *loop* (*nested loop*), dengan tujuan memudahkan untuk keluar dari *loop* terdalam menuju ke pernyataan yang terletak di luar *loop* terluar.

Menggunakan *exit ( )* Untuk Menghentikan Eksekusi Program.

Suatu eksekusi program dapat dihentikan (secara normal) melalui pemanggilan fungsi *exit( )*. Hal ini biasa dilakukan, jika di dalam suatu eksekusi terdapat suatu kondisi yang tak dikehendaki. 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. Sedangkan untuk menunjukkan kesalahan, nilai yang diberikan pada argumen fungsi diisi dengan nilai bukan-nol. Pada contoh program berikut, eksekusi program akan dihentikan hanya jika tombol 'X' ditekan

```
/* File program : keluar.c
Pemakaian exit() untuk menghentikan eksekusi program */

#include <stdio.h>
#include <stdlib.h>

main()
{
  char kar;

  printf("Tekanlah X untuk menghentikan program.\n");
```

```
  for ( ; ;)
  {
     while ((kar = getchar()) == 'X')
        exit(0);
  }
}
```

C. TUGAS PENDAHULUAN

1. Buatlah program untuk mencetak huruf tertentu hingga membentuk sebuah matriks sebanyak m x n dimana m menyatakan banyak baris dan n menyatakan banyak kolom.

   Tampilan:

   Masukkan Jumlah baris:3

   Masukkan jumlah kolom:6

   A A A A A A

   A A A A A A

   A A A A A A

2. Dengan menggunakan nested loop, buatlah tampilan seperti gambar dibawah ini!

   Tampilan:

   * * * _ _ _ _

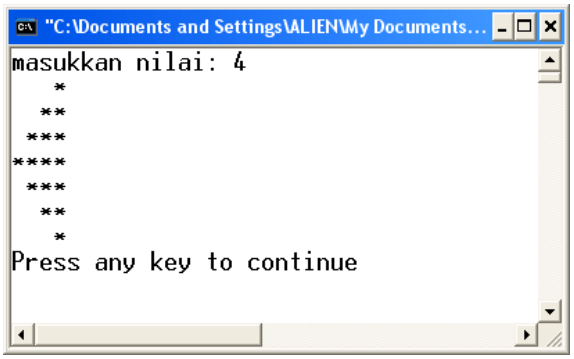
   * * * _ _ _ _

   _ _ _ _ _ _ _

   _ _ _ _ _ _ _

D. PERCOBAAN

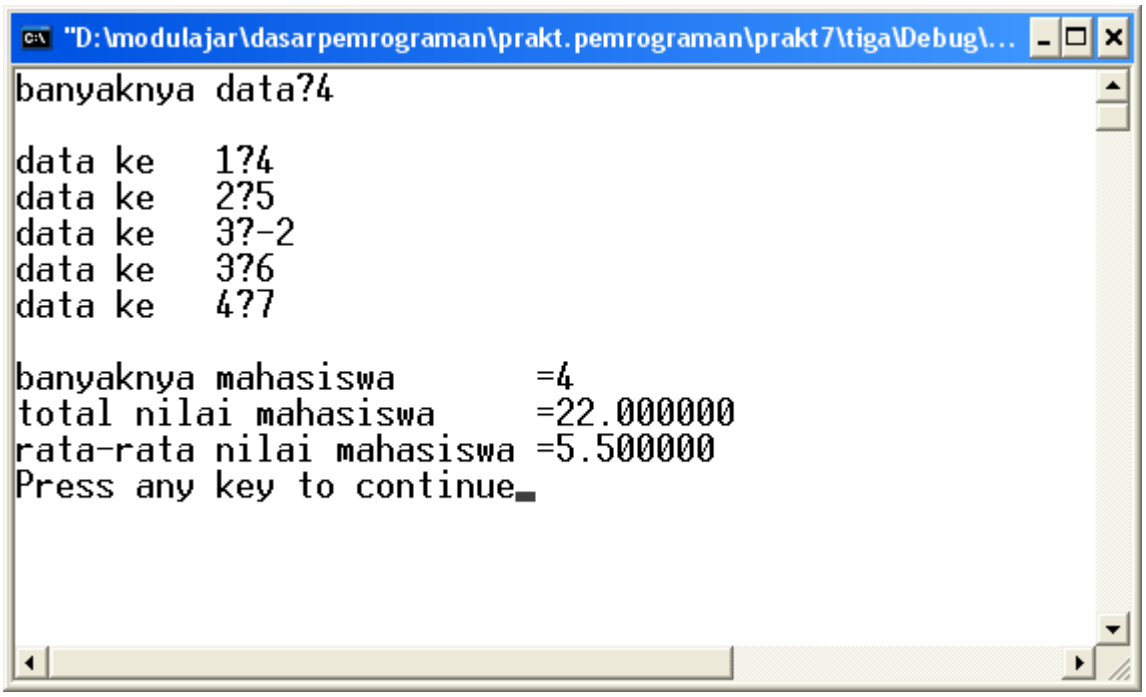
1. Dengan menggunakan nested loop, buat tampilan di layar monitor seperti yang terlihat dibawah ini bila diinputkan nilai n=4



2. Dengan menggunakan nested loop, buatlah tampilan seperti gambar dibawah ini:

```
Tampilan:
1   2   3   4   5
2   4   6   8   10
3   6   9   12  15
```

3. Dengan menggunakan pernyataan *continue*, buatlah program untuk menghitung total dan rata-rata nilai mahasiswa. (Jika ditemui nilai mahasiswa yang dimasukkan sebagai data berupa nilai negaif, maka proses perulangan untuk memasukkan nilai mahasiswa ini akan diulangi kembali. Proses untuk mengembalikan ke awal perulangan kembali dapat dilakukan dengan pernyataan *continue*)



4. Dengan menggunakan pernyataan *goto* , buatlah program untuk menyeleksi nilai dari variabel B (Jika nilai B adalah sama dengan nilai 0, maka proses program akan melompat ke bagian yang ditunjukkan oleh label yang bernama `Tak_berhingga`)

E. LAPORAN RESMI

1. Buatlah desain flowchart dari semua percobaan yang telah dilakukan..

2. Dengan menggunakan nested loop, buatlah program untuk menghitung total nilai dari M bilangan prima yang pertama.