MODUL 3: CONTROL STATEMENT: REPETITION

LAB 3.1: TUJUAN PRAKTIKUM

Learning outcome untuk praktikum pemrograman dasar pada modul 3 ini adalah:

LO 2. Mahasiswa mampu mengimplementasikan dan menganalisis konsep dari pernyataan kontrol pemrograman.

Secara khusus, tujuan dari praktikum pada modul 3 ini adalah:

1. Memahami konsep dan kegunaan kendali program iteratif (pengulangan) dalam pemrograman.
2. Mampu menerapkan penggunaan kendali program iteratif dalam bahasa C++.

LAB 3.2: DASAR TEORI

Repetition structure digunakan untuk melakukan pengulangan (iterasi/repetisi) terhadap suatu pernyataan atau suatu blok pernyataan. Kendali program iteratif sangat berguna apabila terdapat blok pernyataan yang harus dikerjakan lebih dari satu kali, karena pemrogram cukup menulis blok pernyataan tersebut satu kali saja, dan program tersebut dengan sendirinya akan mengulangi pengerjaannya sesuai persyaratan yang ditentukan.

Terdapat dua jenis perulangan, yaitu *counter-controlled* dan *event-controlled*. Perulangan yang kita sudah tahu jumlah perulangan yang diperlukan dinamakan dengan *counter-controlled repetition* (perulangan yang dikendalikan oleh pencacah/penghitung). Namun seandainya kita tidak tahu persis berapa jumlah perulangan yang diperlukan, maka kita dapat mengendalikan jalannya perulangan berdasarkan munculnya kejadian tertentu yang disebut dengan perulangan yang dikendalikan oleh kejadian tertentu.

Empat komponen yang harus ada dalam suatu perulangan yang termasuk di dalam jenis *counter -controlled repetition* adalah sebagai berikut.

1. Variabel kontrol atau sering disebut juga dengan variabel **pencacah**. Nama variabel pencacah yang umum digunakan dalam praktek pemrograman misalnya ***i, j, k***.
2. Nilai awal dari variabel pencacah. Dalam program harus ada pemberian nilai awal kepada variabel pencacah, misalnya ***i=0*.**
3. Kenaikan atau penurunan, variabel pencacah dimodifikasi setiap kali melalui perulangan.
4. Sebuah kondisi yang menguji nilai akhir dari variabel pencacah (yaitu, apakah looping harus dilanjutkan).

Kendali program pengulangan dalam bahasa C++ dapat dilakukan dengan 3 pernyataan, yaitu pernyataan **While**, pernyataan **Do…While**, dan pernyataan **For**.

HINT

Pernyataan While dan Do...WHile, adalah pernyataan pengulangan yang mempunyai kemampuan untuk menyaring (filter) laju/jalannya eksekusi program (mirip dengan pernyataan if).

PERNYATAAN WHILE

Pernyataan while merupakan pernyataan pengulangan di mana pengulangan blok pernyataan akan dilakukan selama kondisi syaratnya dipenuhi. Adapun sintaks dari pernyataan while adalah sebagai berikut:

```
while(kondisi)
{
  .....        // blok pernyataan
  .....
}
```

PERNYATAAN DO...WHILE

Pernyataan do...while memiliki fungsi yang hampir sama dengan perintah while, tetapi dalam pernyataan do...while ini, blok pernyataan dilakukan terlebih dahulu sebelum pengecekan kondisi syarat. Dengan demikian, blok pernyataan akan dikerjakan paling sedikit satu kali. Sintaks dari pernyataan do...while ditunjukkan di bawah ini:

```
do
{
        .....        //blok pernyataan
        .....
} while(kondisi)
```

PERNYATAAN FOR

Pernyataan for adalah pernyataan pengulangan yang meminta pendeklarasian nilai awal variabel kondisi, kondisi yang dipenuhi, dan peningkatan (increments) dari variabel kondisi. Sintaks pernyataan for adalah sebagai berikut:

```
for(inisialisasi; kondisi; pengatur variabel)
{
        .....        //blok pernyataan
        .....
}
```

KOMBINASI KENDALI PROGRAM SELEKSI DAN REPETISI

Pada kenyataannya akan sangat jarang kita memerlukan hanya satu jenis kendali program dalam sebuah program komputer. Seringkali kita akan membutuhkan lebih dari satu jenis kendali program, sebagai contoh seleksi dan repetisi. Oleh karena itu, kita harus berlatih menggunakan kedua jenis kendali program tersebut dalam percobaan berikut ini.

LAB 3.3: LATIHAN

PERCOBAAN 1 : PERULANGAN "FOR"

Perhatikan program di bawah ini, kemudian salin (ketik ulang, jangan copy-paste), *compile*, dan eksekusi program tersebut.

```
#include<iostream>
using namespace std;

int main()
{
   int i, batas;

   cout << "Masukkan jumlah pengulangan : ";
   cin >> batas;

   for (int i = 0; i < batas; i++) {
      cout << i+1;
   }
         return 0;
}
```

1. **Apakah jumlah total perulangan yang dilakukan sama dengan nilai yang dimasukkan di awal?**
2. **Bagian manakah yang perlu diubah apabila kita menginginkan angka dicetak dimulai dari -1?**

PERCOBAAN 2 : PERULANGAN "WHILE"

Perhatikan program di bawah ini, kemudian salin (ketik ulang, jangan copy-paste), *compile*, dan eksekusi program tersebut.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
   int angka;
   cout << "Masukkan angka max : ";
   cin >> angka;

   while (angka > 0) {
      cout << angka << endl;
      angka++;
   }

   return 0;
}
```

1. **Apakah jumlah total perulangan yang dilakukan sama dengan nilai yang dimasukkan di awal?**
2. **Bagian manakah yang perlu diubah apabila kita tidak menginginkan program berjalan secara**

terus menerus tanpa henti?

PERCOBAAN 3: PERULANGAN "DO-WHILE"

Perhatikan program di bawah ini, kemudian salin (ketik ulang, jangan copy-paste), *compile*, dan eksekusi program tersebut. Jika terdapat kesalahan pada program, perbaiki sehingga dapat sukses terkompilasi, tereksekusi, dan memberikan luaran yang benar.

```
#include <iostream>

int main() {
   int n= 0;
   int sum = 0;

   do {
      sum += number;

      cout << "Enter a number: ";
      cin >> n;
   }
   while (number >= 0);

   cout << "\nThe sum is " << sum << endl;

   return 0;
}
```

PERCOBAAN 4: NESTED LOOP DENGAN DO…WHILE

Program di bawah ini merupakan sebuah contoh nested *loop* dengan menggunakan Do..While. *Nested-loop* adalah *loop* bertingkat, artinya terdapat *loop* di dalam *loop*. **Lengkapilah** program tersebut, kemudian *compile*, dan eksekusi program tersebut. Jika terdapat kesalahan pada program, **perbaiki** sehingga dapat sukses terkompilasi, tereksekusi, dan **memberikan luaran yang benar seperti yang ditampilkan pada kolom sebelah kanan**.

```
do {
        int j = 1;
          do {
            cout<< i <<"\n";
              j++;
          } while (j <= 3) ;
          i++;
       } while (i <= 3) ;
```

Output yang diharapkan:

```
1   1
1   2
1   3
2   1
2   2
2   3
3   1
3   2
3   3
```

PERCOBAAN 5: NESTED LOOP DENGAN FOR

Modifikasi program anda pada Percobaan 4 supaya menjadi menggunakan pernyataan FOR.

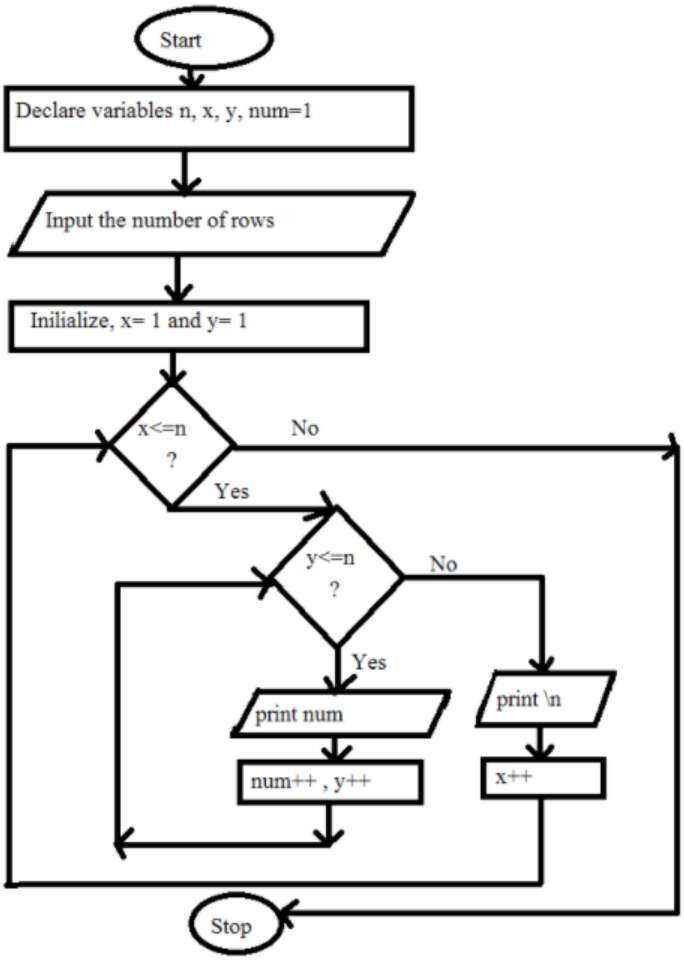
LAB 3.4: CEK POIN 1

Berikut adalah potongan program yang berfungsi untuk menampilkan kelipatan 4 mulai dari 0 sampai dengan 100. Terdapat kesalahan pada potongan kode tersebut, perbaiki dan lengkapi supaya sukses terkompilasi, tereksekusi, dan memberikan hasil yang benar.

```
for mult4 = 0;
mult4 < 100;
mult4 += 4;
cout<<mult4;
```

LAB 3.5: CEK POIN 2

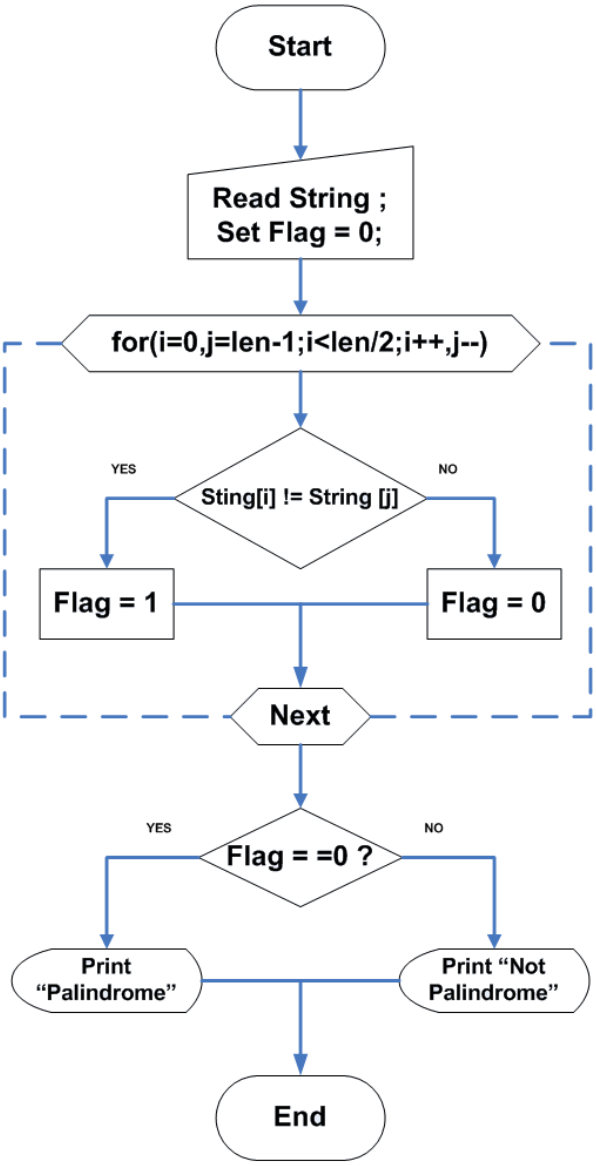
Pelajari *flowchart* berikut ini. Buatlah kode/program berdasarkan flowchart tersebut. Kemudian *compile* dan eksekusi program anda. Apabila terdapat *syntax error* (kesalahan), perbaiki sampai program anda dapat sukses terkompilasi dan dapat dieksekusi.



LAB 3.6: CEK POIN 3

Flowchart di halaman selanjutnya adalah flowchart sebuah program pendeteksi apakah suatu kata merupakan palindrom atau bukan. Iterasi pengulangan akan dilakukan dari indeks awal kata hingga tengah kata (n-jumlah kata / 2). Apabila huruf yang berada di indeks tertentu sama dengan indeks akhir dikurang indeks tersebut dikurang 1, maka kata tersebut adalah sebuah palindrome. Tugas Anda adalah membuat source code program dari flow tersebut.

Namakan program yang sedang Anda buat dengan nama : palindrome.cpp.



LAB 3.7: REFERENSI

Di bawah ini adalah referensi online dari UNIT III. Apabila mengalami kesulitan dalam memahami materi, silahkan buka referensi di bawah ini :

1. http://www.cprogramming.com/tutorial/lesson3.html
2. https://www.w3schools.com/cpp/cpp_while_loop.asp

LAB 3.8: SEKILAS MATERI UNIT BERIKUTNYA

Materi praktikum pemrograman dasar unit selanjutnya akan lebih banyak membahas beberapa hal, terkait dengan algoritma repetisi. Beberapa hal yang patut Anda ketahui, antara lain

1. Pendefinisian Pointer dan Array
2. Jenis--jenis penggunaan pointer
3. Merancang program dari flowchart dan pseudocode.