BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

LECTURE NOTES

COMP6599 – Algorithm and Programming

Minggu 4

Sesi 5

Program Control: Selection & Repetition

People
Innovation
Excellence

LEARNING OUTCOMES

Learning Outcomes (Hasil Pembelajaran) :

Setelah menyelesaikan pembelajaran ini mahasiswa akan mampu :

1. LO 2 : Menerapkan sintaks-sintaks dan fungsi-fungsi bahasa pemrograman C dalam pemecahan masalah.
2. LO 3 : Membuat program dengan menggunakan bahasa C dalam pemecahan masalah.

OUTLINE MATERI (Sub-Topic): Program Control: Selection & Repetition

1. Pemilihan (*Selection*)
2. Perulangan (*Repetition*)
3. Break Vs. Continue
4. Go to dan Label
5. Error Type

Program Control: Selection & Repetition

1. Pemilihan (*Selection*)

   Stuktur kendali pemilihan digunakan untuk memilih langkah-langkah mana yang akan dikerjakan terlebih dahulu berdasarkan hasil pemeriksaan suatu kondisi.

   Bahasa C menyediakan 5 langkah dalam pemilihan yaitu :

   - *if*

   If digunakan untuk memilih apakah proses tersebut akan dilakukan atau tidak. Berikut bentuk dari sintax *if* yang digunakan dalam bahasa C :

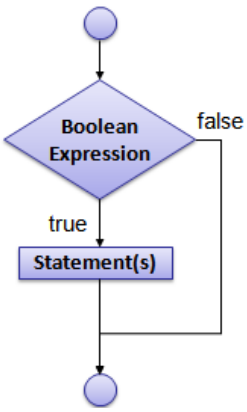
   ```
   if(expression){ statement(s); }
   ```

   Dimana *expression* adalah syarat yang akan dikerjakan, syarat ini memiliki nilai benar (*true*) atau salah (*false*). Bila *expression* tersebut bernilai benar (*true*), statement di dalam blok if tersebut akan dijalankan. Bila nilai expression bernilai salah (*false*), maka program akan berakhir. Berikut merupakan contoh program if pada bahasa C:

   ```
   if(r>=0)
   {
       area = r*r*pi;
       printf("Area dari r : %d adalah %f",r, area);
   }
   ```

   Pada contoh potongan program diatas, misalkan nilai dari radius adalah 5. Program akan mengeksekusi perintah yang ada di blok *if* tersebut bila syarat terpenuhi. Pertama-tama akan dilakukan pengecekan apakah nilai radius saat ini bernilai lebih dari atau sama dengan 0? Jawabannya adalah benar, 5 >= 0, maka program akan menghitung nilai dari area, dan kemudian akan dicetak nilai dari radius dan area tersebut. Bila tidak, maka tidak ada statement yang akan dieksekusi(dijalankan).

   Secara *flowChart* bentuk dari *if* adalah :



- *if-else*

If-else digunakan untuk memilih satu dari 2 proses yang akan dieksekusi. Berikut bentuk dari sintax *if-else* yang digunakan dalam bahasa C :

```
if (expression){
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```

Statement untuk kondisi yang benar diletakkan didalam blok *if* yang dibatasi dengan tanda kurung kurawal buka dan tutup. Kemudian untuk statement kondisi yang salah diletakkan pada *else* blok yang diawali dan diakhiri dengan kurung kurawal buka dan tutup juga.

Berikut contoh program dari penggunaan *if-else* pada bahasa C:

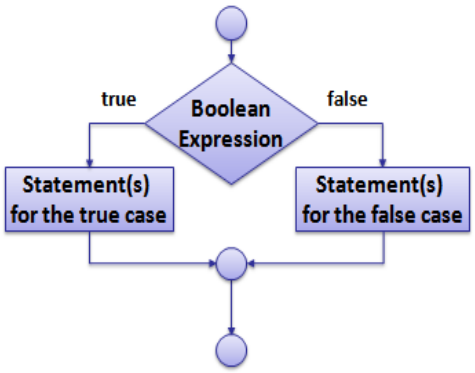
```c
#include <stdio.h>

int main()
{

    int angka;
    scanf("%d", &angka);
    if(angka<0 || angka>100)
        printf("Wrong Input");
    else

        printf("Input : %d", angka);

    getchar();
    return 0;
}
```

Pada program tersebut meminta inputan yang akan ditampung pada variabel angka. Fungsi seleksi dilakukan dimana terdapat dua buah kondisi yang dihubungkan dengan logical operator OR "||", yang berarti bahwa bila salah satu kondisi terpenuhi atau benar maka *expression* tersebut akan bernilai benar. Andaikan **angka** bernilai 200, maka program akan mengecek apakah **angka** < 0? Jawabannya adalah salah (*false)*, kemudian program akan mengecek kembali kondisi berikutnya apakah **angka** > 100? 200 bernilai lebih dari 100, maka kondisinya adalah benar. Karena relasi yang ada adalah OR, maka nilai akhir dari syarat ini adalah benar, sehingga program akan mencetak *"Wrong input"*.

Tetapi sebaliknya bila kondisi **angka** yang dimasukkan adalah 50, maka kondisi pertama dan kedua bernilai salah, dan syarat tidak terpenuhi, maka program akan mencetak *"Input : "* yang dilanjutkan dengan nilai dari **angka** tersebut.

Bila melihat kembali pada program yang ada, pada blok statement *if* ataupun *else* tidak terdapat tanda kurung kurawal buka dan tutup "{" "}" yang biasanya digunakan untuk menandakan batasan blok. Hal ini dapat dilakukan jika hanya ada satu statement yang akan dieksekusi pada blok tersebut. Bila terdapat lebih dari satu statement yang ingin dilakukan pada blok statement if, maka perlu digunakan tanda kurung kurawal buka dan tutup "{" "}" sebagai penanda.

Secara flow Chart bentuk dari *if-else* adalah :



- nested if

Nested if merupakan jenis variasi yang digunakan bila memiliki lebih dari satu syarat dan syarat tersebut saling berkaitan. Yang dimasud dengan *nested* adalah di dalam if atau *if-else* statement terdapat *if* atau *if-else* statement lainnya. Tidak ada batasan maximum untuk *if* di dalam *if* tersebut. Berikut adalah contoh *nested if*:

```
if (score >= 90)
    grade = 'A';
else
    if (score >= 80)
        grade = 'B';
    else
        if (score >= 70)
            grade = 'C';
        else
            if (score >= 60)
                grade = 'D';
            else
                grade = 'F';
```

```
if (score >= 90)
    grade = 'A';
else if (score >= 80)
    grade = 'B';
else if (score >= 70)
    grade = 'C';
else if (score >= 60)
    grade = 'D';
else
    grade = 'F';
```

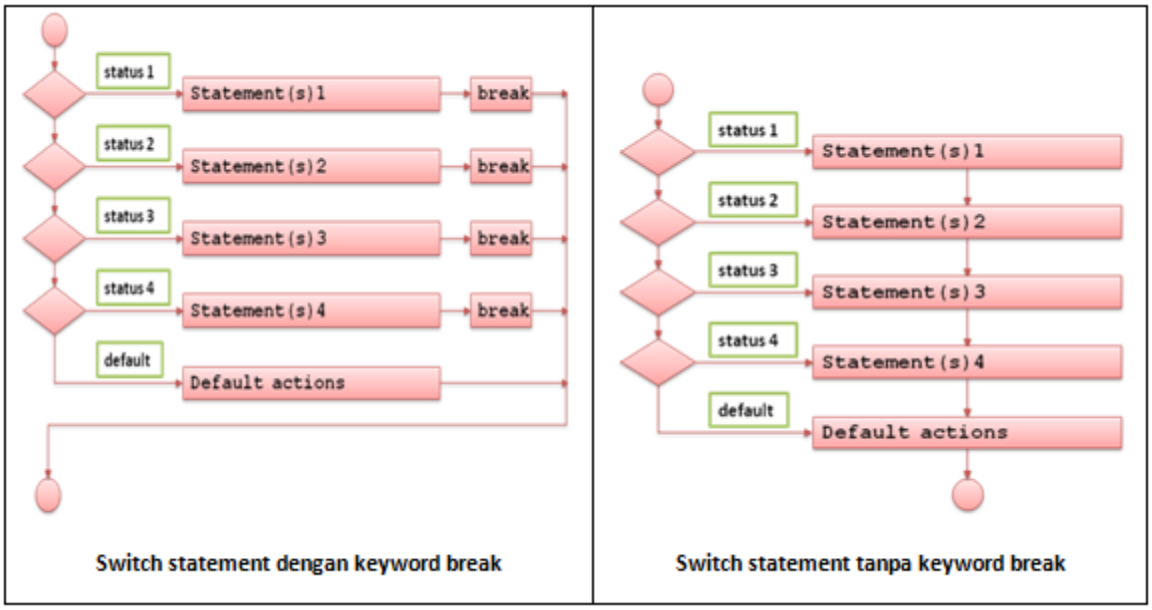
Potongan coding diatas adalah contoh penyeleksian dalam memberikan grade berdasarkan dari nilai yang ada. Bila kondisi tidak dipenuhi maka didalam blok *else* terdapat kondisi lainnya yang dicek kembali, dan seterusnya. Ini adalah salah satu contoh sederhana dari *nested if*. Potongan coding dikolom sebelah kiri dan kanan adalah sama, hanya lebih disarankan untuk mengikuti penulisan coding di kolom sebelah kanan.

- switch-case

Switch – case adalah salah satu cara seleksi yang digunakan untuk memilih satu dari sejumlah alternatif. Seleksi yang dilakukan pada `switch` statement berdasarkan pada sebuah status. Format penulisan statement `switch` itu sendiri dapat dilihat dibawah ini :

```
switch (switch-expression)
{
case value1:    statement(s)1;
     break;
case value2:    statement(s)2;
     break;
…
case valueN:    statement(s)N;
     break;
default:  statement(s)-for-default;
}
```

switch-expression hanya boleh menggunakan sebuah nilai dengan tipe karakter(*char*), *byte*, *short* atau *int*. Kemudian untuk nilai *value1*, *value2*, *valueN* harus mempunyai tipe data yang sama seperti tipe data yang ada di *switch-expression*. Statement-statement pada setiap pilihan (*case*) akan dieksekusi sampai program membaca *keyword* `break` (**akan dijelaskan lebih rinci di topik selanjutnya**), atau hingga *switch* statement telah berakhir. *Keyword break* dan *default case* bersifat optional, boleh digunakan boleh juga tidak. *Case* statement tersebut juga dieksekusi secara berurut (*sequential*).



Gunakan *keyword* break bila memang diperlukan. Karena bila tidak ada *keyword break*, misalnya status 2 terpenuhi, maka statement 2 dan seterusnya akan dieksekusi juga selama blok *switch* belum berakhir. Untuk lebih kelas dapat dilihat pada *flow chart* diatas.

Dibawah ini adalah contoh program sederhana yang menerapkan *switch* statement dalam melakukan seleksi. Seleksi dilakukan terhadap hasil input user berupa bilangan integer yang disimpan ke dalam variable number. Yang menjadi *switch expression* adalah nilai sisa dari number dibagi dua (modulus). Statement di case 0 akan dijalankan bila sisa bagi bernilai 0, dan statement di case 1 akan dijalankan bila sisa bagi tersebut bernilai 1. Di dalam setiap statement *case* juga diberikan *keyword break.* Program sederhana ini merupakan contoh dari penentuan apakah sebuah angka merupakan bilangan ganjil atau genap.

```c
#include <stdio.h>

int main()
{

    int number;
    scanf("%d", &number);
    switch(number%2)
    {
        case 0 : printf("%d is even number",number);
            break;
        case 1 : printf("%d is odd number",number);
            break;
    }
    getchar();
    return 0;
}
```

Berikut adalah contoh *input* dan *output* dari program tersebut:

```
9
9 is odd number
```

```
10
10 is even number
```

- *Conditional expression*

Jenis terakhir dari seleksi statement dapat juga dilakukan dengan bentuk *conditional expression*. Format yang digunakan adalah sebagai berikut:

`Boolean-expression ? expression1 : expression2;`

Lihatlah contoh *if* sederhana yang dituliskan dalam bentuk *conditional expression* berikut ini:

```c
if (x > 0)
    y = 1;
else
    y = -1;
```

Use the conditional Expression →

```c
y = (x >
```

2. Perulangan (*Repetition*)

Struktur kendali perulangan atau iterasi (*repetition)* digunakan untuk melaksanakan satu atau beberapa instruksi secara berulang-ulang. Frekuensi pengulangan dapat ditentukan dalam program atau ditentukan saat program dijalakan.

Dalam iterasi ini instruksi yang dikerjakan berulang-ulang harus memiliki nilai variabel pengendali kondisi yang harus selalu berubah setiap iterasi terjadi agar iterasi dapat berakhir. Pada bahasa C menyediakan 3 instruksi iterasi yaitu :

- *for*

For digunakan untuk mengulang suatu statement yang dideklarasikan berdasarkan suatu kondisi atau syarat tertentu.

Format penggunaan For:

- Jika hanya 1 statement yang dijalankan dalam perulangan

```
for ([initialization];[condition];[increment/decrement])
        <statement>
```

- Jika terdiri lebih dari 1 statement yang dijalankan dalam perulangan

```
for (initialization; condition; increment/decrement)
{
        <block of statements>
}
```

Contoh:

```
for ( int i = 0; i < 5; i++) {
     printf("Value of i is = %d" , i);
     printf("Loop number %d", (i+1));
}
```

Pada potongan program di atas, kedua statement yang terdapat didalam *for* akan dijalankan selama kondisi i < 5 terpenuhi. Dalam hal ini, awalnya nilai i adalah 0 dan ketika proses dilakukan, nilai i akan ditambahkan satu point (*increment* i++), dan proses dilakukan terus.

- *while*

While digunakan untuk mengulang suatu statement selama expression yang diberikan didalamnya bernilai benar, jika bernilai salah, maka statement yang dideklarasikan di dalamnya tidak akan diulang.

```
while (expression) { <block statement> }
```

Contoh :

```
int i = 0;
while (i < 5){
     printf("Value of i is %d", i);
     i++;
}
```

Pada potongan program di atas, program akan melakukan pengecekan terlebih dahulu apakah nilai i < 5, jika benar, maka program akan mengulang kedua statement yang terdapat di dalam *while.*

- *do-while*

Do...While digunakan untuk mengulan suatu statement yang dideklarasikan sesudah keyword Do dan setelah menjalankan statement tersebut, dilakukan pengecekan terhadap expression yang diberikan, apakah bernilai benar atau salah. Jika bernilai benar, maka statement tesebut akan diulang kembali. Jika bernilai salah, maka statement tersebut hanya akan dijalankan satu kali tanpa dilakukan pengulangan.

```
do{ <block statement> }while (expression);
```

Contoh :

```
int i = 0;
do{
     printf("Value of i is %d", i);
     i++;
}while (i < 5);
```

Pada potongan program di atas, program akan menjalankan terlebih dahulu kedua statement di dalam *do* kemudian dilakukan pengecekan apakah nilai i < 5, jika benar, maka program akan kembali mengulang kedua statement tersebut.

3. Break Vs. Continue

Break merupakan instruksi yang menyebabkan proses keluar dari instruksi *for, while, do-while, switch*.

Continue dapat menyebabkan proses mengabaikan (melewati) instruksi selanjutnya dan melanjutkan proses iterasi berikutnya pada instruksi *for, while, do-while*.

Berikut contoh program continue pada bahasa C:

```
#include <stdio.h>
int main() {
        int x;
        for(x=1; x<=10; x++) {
        if (x == 5) continue;
        printf("%d ", x);
        }
        return 0;
}
```

Output : 1 2 3 4 6 7 8 9 10

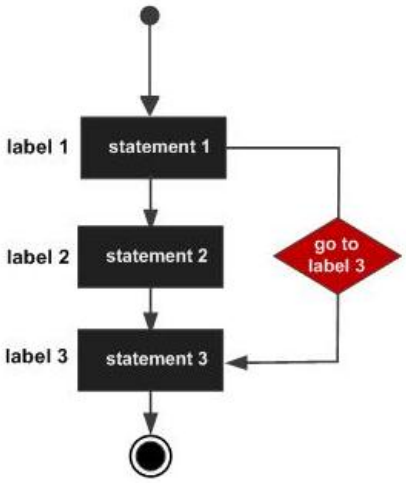
Pada program tersebut terlihat bahwa ketika x bernilai 5 maka proses 5 akan dilewati dan tidak di cetak. Sehingga akan menghasilkan output : 1 2 3 4 6 7 8 9 10.

4. Go to dan Label

   Bahasa C menyediakan cara untuk menyediakan lompatan goto melalui label dengan fungsi yang sama.

   Penggunaan goto ini sangat tidak dianjurkan dalam bahasa pemrograman dikarenakan akan membuat struktur code menjadi tidak teratur, dan membuat sulit untuk melacak aliran kontrol dari sebuah program.

   Berikut bentuk flow chart dari *goto label* pada bahasa C :



Berikut contoh penggunaan goto label pada bahasa C:

```c
#include <stdio.h>

int main ()
{
   /* local variable definition */
   int a = 10;

   /* do loop execution */
   LOOP:do
   {
      if( a == 15)
      {
         /* skip the iteration */
         a = a + 1;
         goto LOOP;
      }
      printf("value of a: %d\n", a);
      a++;
   }while( a < 20 );

getchar();
   return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

5. Error Type

Terdapat beberapa jenis error yang sering terjadi pada umumnya, antara lain:

a. Lupa untuk memberikan tanda kurung batas blok if. Perlu diingat bahwa Anda boleh menghilangkan penulisan "{" " }" sebagai penanda pada *if* statement jika hanya terdapat satu statement pada blok tersebut. Lihat contoh berikut:

```
if(radius >=0 )
    area = radius * radius * PI;
    System.out.println("The area is " + area);
```

Pada potongan program diatas, bila syarat radius >=0 terpenuhi maka program akan menjalankan statement if yaitu area = radius * radius * PI. Kemudian program akan melanjutkan untuk mengeksekusi statement berikutnya (dalam konteks melanjutkan pengeksekusian program yang berada diluar blok *if*. Bila syarat radius >=0 tidak terpenuhi, maka program tidak akan menjalankan perhitungan area, tetapi akan langsung mengeksekusi statement System.out.println, karena statement tersebut tidak termasuk statement didalam blok *if*.

b. Memberikan tanda ";" pada akhir persyaratan *if*. Hal ini akan menyebabkan *error*. Ingat bahwa tanda titik koma ";" digunakan untuk menandakan akhir dari sebuah statement.

```
if(radius >=0 );
{
     area = radius * radius * PI;
     System.out.println("The area is " + area);
}
```

Logic error

c. **Perulangan dalam pengecekan nilai**. Pada contoh dibawah ini terlihat bahwa statement *if* akan dijalankan bila syarat/kondisi bernilai benar/true. Maka secara tersirat *if* akan melakukan pengecekan apakah nilai dari variable yang dicek bernilai benar.

```
if (even == true)
     System.out.println("It is even");
```

Equivalent

```
if (even)
     System.out.println
```

d. **Pernyataan blok else yang ambigu dalam nested if**. Contoh:

```
int i = 1;
int j = 2;
int k = 3;

if(i>j)
     if(i>k)
          System.out.println("A");
else
     System.out.println("B");
```

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
     if (i > k)
          System.out.println("A");
     else
          System.out.println("B");
```

Anda dapat melihat blok else pada kolom sebelah kiri cukup ambigu, apakah blok else tersebut merupakan blok else milik `if(i>j)` atau `if(i>k)`. Seharusnya blok else tersebut ditulis seperti potongan program disebelah kanan. Bila Anda ingin membuat else untuk milik `if(i>j)`, maka Anda perlu memberikan tanda "{ }" untuk membatasi statement blok `if(i>j)` tersebut.

e. Compile time-error : disebabkan oleh kesalahan sintaks.

f. Link-time error : berhasil di compile, namun menyebabkan *link-error*, dikarenakan kesalahan pada saat kompilasi. Biasanya disebabkan karena file *.exe* tidak berhasil dihubungkan sehingga menyebabkan format file *.exe* tidak dapat dibentuk.

g. Run-time error : program berhasil di susun namun terdapat kesalahan pada saat run-time. Biasanya hal ini disebabkan karena operasi numerik seperti pembagian dengan nol, dan lain sebagainya.

h. Logical error : berhasil di eksekusi namun, hasil output tidak sesuai dengan yang diharapkan. Hal ini terjadi dikarenakan kesalahan pada algoritma yang dibentuk.

SIMPULAN

1. Fungsi seleksi digunakan untuk mengaplikasikan alternaive pilihan. Ada beberapa jenis seleksi: *if* statement, *if-else* statement, *nested if* statement, *switch* statement, dan *conditional expression*.

2. Semua macam dari *if* statement merupakan struktur control yang berdasarkan pada *expression*. Bergantung pada nilai *true* atau *false* yang dihasilkan dari expression tersebut (syarat seleksi), seleksi *if* ini akan memilih satu dari dua pilihan.

3. *Switch* statement adalah struktur control yang bergantung pada tipe dari *switch-expression* yang hanya dapat mengaplikasikan tipe *char*, *byte*, *short*, atau *int*.

4. Keyword *break* bersifat pilihan (optional), tetapi biasanya digunakan pada setiap kahir case statement dengan tujuan untuk memberhentikan pengeksekusian program yang tersisa dalam *switch* statement. Jika *break* statement tidak digunakan, maka case

statement yang selanjutkan tetap akan dieksekusi, dimana *switch* statement melakukan pengeksekusian secara *sequential* order.

5. Ada 3 jenis sintaks perulangan yang dapat digunakan, yaitu *while, do-while dan for*.
6. Break merupakan instruksi yang menyebabkan proses keluar dari instruksi *for, while, do-while, switch.*
7. Continue dapat menyebabkan proses mengabaikan (melewati) instruksi selanjutnya dan melanjutkan proses iterasi berikutnya pada instruksi *for, while, do-while*.
8. Bahasa C menyediakan cara untuk menyediakan lompatan goto melalui label dengan fungsi yang sama.
9. Penggunaan goto ini sangat tidak dianjurkan dalam bahasa pemrograman dikarenakan akan membuat struktur code menjadi tidak teratur, dan membuat sulit untuk melacak aliran kontrol dari sebuah program.
10. Beberapa macam tipe error pada bahasa C :
    a. Compile time-error
    b. Link-time error
    c. Run-time error
    d. Logical error

DAFTAR PUSTAKA

1. http://www.tutorialspoint.com/cprogramming/c_goto_statement.htm
2. Paul J. Dietel,Harvey M. Deitel,. 2010. C : how to program. PEAPH. New Jersey. ISBN:978-0-13-705966-9 Chapter 3 & 4
3. Choosing between Alternatives: http://docs.roxen.com/pike/7.0/tutorial/statements/conditions.xml
4. Getting Controls: http://aelinik.free.fr/c/ch10.htm
5. Doing the Same Thing Over and Over: http://aelinik.free.fr/c/ch07.htm
6. Thompson Susabda Ngoen, 2006. Pengantar Algoritma dengan Bahasa C. Salemba Teknika. ISBN : 979-9549-25-6. Bagian 4.