BAB II
IMPLEMENTASI ALGORITME

STRUKTUR ALGORITME

Algoritme yang disusun oleh seseorang tidak bersifat unik, tetapi jawaban dari suatu persoalan yang dilakukan oleh algoritme bersifat unik. Sebagai contoh, algoritme untuk menghitung nilai 9^5 seperti pada LATIHAN 1 nomor 3 adalah sebagai berikut:

ALGORITME 3a. MENGHITUNG 9^5.
1. hitung hasil ← 9 x 9 x 9 x 9 x 9.
2. cetak hasil.

Nilai 9 dipangkatkan dengan 5 pada ALGORITME 3a dihitung sekaligus, yaitu nilai 9 dikalikan sebanyak lima kali. Nilai ini juga sama bila ditulis secara bertahap seperti yang tercantum pada ALGORITME 3b di bawah ini.

ALGORITME 3b. MENGHITUNG 9^5.
1. hasil ← 9.
2. hasil ← hasil x 9.
3. hasil ← hasil x 9.
4. hasil ← hasil x 9.
5. hasil ← hasil x 9.
6. cetak hasil.

ALGORITME 3b juga akan menghasilkan hasil perhitungan yang sama dengan ALGORITME 3a. Perhatikan instruksi pada langkah 2 hingga 5 berikut:

hasil ← hasil x 9.

Nilai 'hasil' pada ruas sebelah kanan disebut R-value, sedangkan pada ruas kiri disebut 'L-value'. Komputer mempunyai aturan bahwa yang diolah pada R-value adalah isi memori pada alamat tersebut, sedangkan L-value adalah alamat itu sendiri. Dengan demikian, walaupun peubahnya sama, namun kedua nilai tersebut sangat berbeda. Jadi, pernyataan:

hasil ← 9

berarti memori pada alamat 'hasil' diisi dengan nilai '9'; dan pernyataan:

hasil ← hasil x 9

berarti isi memori pada alamat 'hasil' (yaitu 9) dikalikan dengan nilai '9' dan hasilnya disimpan pada alamat memori 'hasil'.

Instruksi pada langkah 2 hingga 5 adalah sama, sehingga jika menyusun algoritme untuk menghitung 9^{100}, maka instruksi yang sama tersebut harus ditulis sebanyak 99 kali. Hal ini kurang efisien sehingga dapat disempurnakan seperti ALGORITME 4 berikut ini yang menghitung nilai perpangkatan secara umum, yaitu 9^n (n>0):

ALGORITME 4. MENGHITUNG 9^n.
1. baca n.
2. hasil ← 9; counter ← 1.
3. selama counter < n lakukan:
   - hasil ← hasil * 9
   - counter ← counter + 1
4. cetak hasil.

Pernyataan pada langkah 3 berarti bahwa kedua baris instruksi tersebut harus dilakukan oleh komputer secara berulang-ulang selama kondisi 'counter < n' terpenuhi. Jika kondisi tersebut tidak terpenuhi, maka algoritme langsung pergi ke langkah 4, yaitu 'cetak hasil'. Proses pengulangan seperti ini sangat sering dilakukan dalam menyusun suatu algoritme maupun program komputer sehingga perlu pemahaman yang mendalam. Jenis-jenis pengulangan akan dibahas pada materi struktur pemrograman.

FASE IMPLEMENTASI

Setelah algoritme disusun, maka langkah selanjutnya untuk menjawab suatu persoalan dengan komputer adalah menulis program komputer dengan menggunakan bahasa pemrograman tertentu sehingga dapat diproses oleh komputer. Bahasa pemrograman komputer terbagi menjadi tiga kelompok besar, yaitu bahasa mesin, bahasa tingkat rendah atau sering disebut sebagai bahasa rakitan (*assembly*), dan bahasa tingkat tinggi. Berikut ini adalah karakteristik dari masing-masing kelompok bahasa pemrograman komputer:

Bahasa mesin (*machine language*):
- berupa *micro-instruction* atau *hardwire*
- prosesnya sangat cepat
- sangat sulit dipahami manusia
- program sangat panjang
- sangat tergantung pada arsitektur mesin komputer
- tidak perlu penterjemah
- built-in pabrik

Bahasa tingkat rendah atau bahasa rakitan (*assembly language*):
- berupa macro-instruction (*assembly*)
- proses lebih cepat
- relatif sulit dipahami dan program lebih panjang
- tergantung pada arsitektur mesin komputer
- perlu penterjemah yang disebut dengan *assembler*
- contoh: Macro Assembler (MASM), Turbo Assembler (TASM)

Bahasa tingkat tinggi (*high level language*):

- menyerupai struktur bahasa manusia (bahasa Inggris)
- mudah dipahami
- tidak tergantung pada arsitektur mesin komputer
- perlu penterjemah berupa *interpreter* atau *compiler*
- contoh: BASIC, PASCAL, C, JAVA, dan lain-lain.

Berikut adalah contoh untuk mengimplementasikan ALGORITME 4 dengan menggunakan bahasa pemrograman C.

```
#include <stdio.h>
main() {
    int hasil=9, counter=1, n;                          /* langkah 2 */
    printf("\nTuliskan nilai pangkat : ");  scanf("%d", &n);   /* langkah 1 */
    while (counter<n} {                                 /* langkah 3 */
        hasil *= 9;
        counter++;
    }
    printf("\nHasilnya adalah : %d", hasil);            /* langkah 4 */
    return 0;
}
```

Penjelasan tentang program C ini akan disajikan pada bab berikutnya. Dalam hal ini hanya ingin ditunjukkan bahwa setiap baris instruksi pemrograman yang ditulis sesuai dengan disain algoritme yang telah disusun, atau merupakan terjemahan dari instruksi algoritmenya.

LATIHAN 2.

Buatlah algoritme untuk menjawab setiap persoalan berikut:

1. Mencari jawaban dari sebuah persamaan kuadrat dengan menggunakan rumus abc.
2. Menghitung jumlah dan rata-rata dari n buah bilangan bulat positip.
3. Mendapatkan bilangan terkecil dan terbesar dari n buah bilangan riil.
4. Menentukan pembagi persekutuan terbesar dari dua buah bilangan bulat positip.
5. Menghitung banyaknya bilangan yang lebih kecil, sama dengan, dan lebih besar dari a di antara n buah bilangan riil.