Pengenalan Rekursif

Oleh:

Bagoes Maulana, M.Kom.

bagoesmaulana85

Tentang Rekursif

Struktur data rekursif merupakan fungsi yang memanggil dirinya sendiri. Fungsi ini akan terus berjalan sampai kondisi berhenti terpenuhi.

Dalam sebuah fungsi rekursif, programmer harus menentukan kondisi harus berhenti dari sebuah proses rekursi agar tidak terjadi infinite loop (perulangan yang tak terbatas).

Ilustrasi Rekursif

Diberi permasalahan memotong roti tawar tipis-tipis sampai habis

Kita dapat memberikan algoritma solusinya yang terdiri dari langkah berikut ini:

1. Jika roti sudah habis atau potongannya sudah paling tipis maka pemotongan roti selesai.
2. Jika roti masih bisa dipotong, maka potong tipis dari tepi roti tersebut, lalu **lakukan langkah 1 dan 2** untuk sisa potongannya.

Dari contoh di atas, kita dapat melihat bahwa langkah 2 memanggil langkah 1 dan dirinya sendiri (langkah 2) dengan kata-kata "lakukan langkah 1 dan 2". Inilah yang disebut sebagai rekursif.

Try ME!

Berdasarkan definisi yang telah dijelaskan pada slide sebelumnya, rekursif adalah fungsi yang memanggil dirinya sendiri.

Ketiklah kode disamping dan beri nama file **sd45rekursif.cpp**

Pada kode tersebut, fungsi **`rekursif()`** memanggil dirinya sendiri di dalam fungsi tersebut (lihat baris ke-7).

Setelah itu, silakan kalian coba jalankan kodenya. Apa yang terjadi? Silakan kirim jawaban kalian pada poll di SIPDA Pertemuan 4-5.

```
1   #include <iostream>
2
3   using namespace std;
4
5   void rekursif() {
6       cout << "Ini adalah rekursif\n";
7       rekursif();
8   }
9
10  int main () {
11      rekursif();
12
13      return 0;
14  }
```

Edit ME!

Silakan kalian edit kode pada file **sd45rekursif.cpp** sesuai dengan kode di samping ini.

Pada kode yang sudah diperbaiki ini, fungsi **`rekursif()`** memanggil dirinya sendiri di dalam fungsi tersebut (lihat baris ke-10).

Hanya saja, sebelum dia memanggil dirinya sendiri, kita memberikan batasan agar dirinya tidak mengulang-ngulang tanpa berhenti yaitu dengan memberikan kode **`if (n > 0)`** dan **`n--`**

Dengan memberi if (n > 0) kita memberi syarat rekursif dapat dijalankan apabilan nilai n lebih besar dari 0. Fungsi n-- adalah untuk mengurangi nilai n dengan 1 sampai dia bernilai 0. Ketika nilai n = 0, maka rekursif tidak akan berjalan lagi.

Jika dijalankan, bagaimana kira-kira hasilnya?

```cpp
#include <iostream>

using namespace std;

void rekursif(int n) {
    if (n > 0)
    {
        cout << "Ini adalah rekursif\n";
        n--; // decrement: variabel n dikurangi 1 angka
        rekursif(n); // rekursif fungsi
    }
}

int main() {
    rekursif(10); // kita isi nilai n = 5
    return 0;
}
```

Konsep Rekursif Pada Faktorial (1)

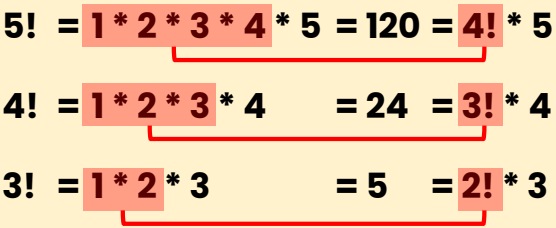
Kalian masih ingat topik matematika tentang **faktorial**?
Iya, **faktorial adalah** hasil perkalian menaik dan biasanya diberi simbol tanda seru " **!** ".

Sebagai contoh, 3 faktorial ditulis dengan 3! yang nilainya merupakan hasil perkalian dari 1 * 2 * 3 = 6.

Contoh lainnya, 5 faktorial ditulis dengan 5!, nilainya merupakan hasil perkalian dari 1 * 2 * 3 * 4 * 5 = 120.

Konsep Rekursif Pada Faktorial (2)

Mari kita lihat faktorial bilangan di bawah ini:



Dari faktorial di samping ini, kita bisa melihat bahwa

hasil faktorial diperoleh dari perkalian bilangan terakhir dengan faktorial sebelumnya.

Ketika kita diminta untuk membuat program menghitung faktorial, maka kita dapat menggunakan konsep rekursif dalam penyelesaiannya.

Formulanya: ***faktorial(n) = faktorial(n-1) * n***

Contoh Program Rekursif Pada Faktorial

Buatlah file baru dengan nama **sd45faktorial.cpp** dan ketikkan kode di samping.

Pada baris ke-5 s.d. 13, kita membuat fungsi rekursif bernama faktorial yang dapat menerima parameter angka **n** bertipe integer. Baris ke-6 s.d. 8 membuat batas bawah fungsi tersebut untuk dapat dijalankan, yaitu ketika bilangan yang diinput adalah 0 atau 1, maka melalui **return n**, maka fungsi faktorial hanya menghasilkan 0 atau 1.

Ketika nilai n lebih besar dari 1, maka kode pada baris ke-9 s.d. 12 dikerjakan, yaitu memproses perhitungan faktorialnya. Di baris ke-11 kita lihat, fungsi faktorial memanggil dirinya sendiri.

Silakan ketik dan coba kode programnya dan bagaimana hasilnya.

```cpp
#include <iostream>

using namespace std;

int faktorial (int n) {
    if (n <= 1) {
        cout << n;
        return n;
    } else {
        cout << n << " * ";
        return faktorial(n-1) * n;
    }
}

int main()
{
    int angka, hasil;

    cout << "Program Menghitung Faktorial\n";
    cout << "==========================\n";
    cout << "Faktorial dari     : ";
    cin >> angka;
    cout << angka << "! = ";
    hasil = faktorial(angka);
    cout << "\nHasil faktorialnya : " << hasil << endl;

    return 0;
}
```

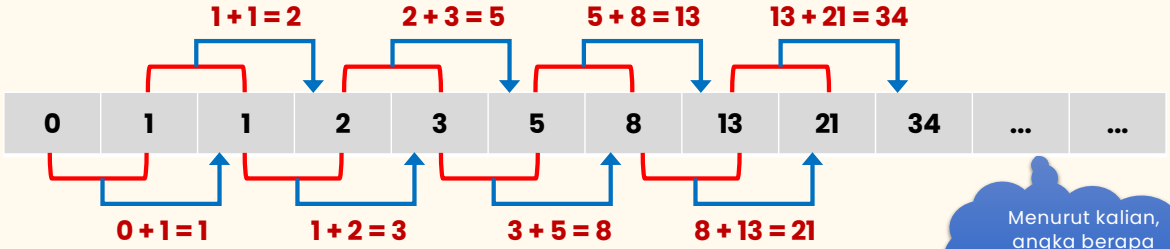
Konsep Rekursif Pada Fibonacci (1)

Berikutnya, kita akan menggunakan fungsi rekursif pada deret Fibonacci. Fibonacci adalah deret angka yang dikembangkan oleh seorang ahli matematika Italia bernama Leonardo Fibonacci pada abad ke-13.

Deret angka Fibonacci dimulai dengan angka 0 dan 1, dan setiap angka berikutnya dihitung dengan menjumlahkan dua angka sebelumnya dalam deret tersebut.

Konsep Rekursif Pada Fibonacci (2)

Perhatikan deret Fibonacci berikut ini:



Dari pola di atas, dapat diperoleh suatu rumus yaitu:

Fn = F(n-1) + F(n-2), dengan n adalah suku bilangan

Pada contoh di atas, suku pertama adalah 0, suku kedua adalah 1, suku ketiga adalah 1, dst.

Dapat kita lihat bahwa rumus Fibonacci tersebut merupakan salah satu contoh implementasi dari konsep rekursif yang memanggil dirinya sendiri di dalam rumus tersebut.

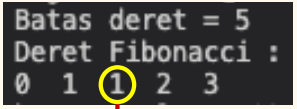
Contoh Program Rekursif Pada Deret Fibonacci

Buatlah file baru dengan nama **sd45fibonacci.cpp** dan ketikkan kode di samping.

Pada baris ke-5 s.d. 11, kita membuat fungsi rekursif bernama fibonacci yang dapat menerima parameter angka **n** bertipe integer. Baris 6 dan 7 akan menentukan langsung nilai suku fibonacci apabila n = 0 maka return 0, sehingga hasilnya juga 0 atau jika n = 1 maka return 1, sehingga hasilnya juga 1. Itulah kenapa bilangan fibonacci suku ke-0 = 0 dan bilangan fibonacci suku ke-1 = 1.

Baris ke-8 ada perintah else, yang artinya jika n bukan 0 atau 1, maka hitung nilai fibonacci sesuai dengan perintah di baris ke-9.

Contohnya, apabila n = 2, maka
= fibonacci (2-1) + fibonacci (2-2)
= fibonacci (1) + fibonacci (0)
= 1 + 0 = 1



Silakan ketik dan coba kode programnya dan bagaimana hasilnya.

```cpp
#include <iostream>

using namespace std;

int fibonacci(int n) {
  if (n == 0 || n == 1) {
    return n;
  } else {
    return fibonacci(n - 1) + fibonacci(n - 2);
  }
}

int main() {
  int batasDeret, i, j = 0;

  cout << "Batas deret = "; cin >> batasDeret;
  cout << "Deret Fibonacci : \n";

  for (i=1; i<=batasDeret; i++) {
    cout << fibonacci(j) << " ";
    j++;
  }

  cout << endl;

  return 0;
}
```

“

```
while (alive)
{
    eat();
    sleep();
    code();
    repeat();
}
```

”