MODUL 1

REKURSIF

A. KOMPETENSI DASAR

1. Memahami mengenai konsep rekursif
2. Mampu memecahkan permasalahan dengan konsep rekursif

B. DASAR TEORI

Rekursif adalah suatu proses atau prosedur dari fungsi yang memanggil dirinya sendiri secara berulang-ulang. Karena proses dalam Rekursif ini terjadi secara berulang-ulang maka harus ada kondisi yang membatasi pengulangan persebut, jika tidak maka proses tidak akan pernah berhenti sampai memori yang digunakan untuk menampung proses tersebut tidak dapat menampung lagi/penuh.

Kelebihan Fungsi Rekursif adalah program menjadi lebih singkat. Pada beberapa kasus, lebih mudah menggunakan fungsi rekursif, contohnya: pangkat, factorial, dan fibonacci, dan beberapa proses deret lainnya. Fungsi rekursif lebih efisien dan cepat dibandingkan proses secara iteratif. Kekurangan Fungsi Rekursif adalah memakan memori lebih besar, karena setiap bagian dari dirinya dipanggil, akan membutuhkan sejumlah ruang memori untuk penyimpanan. Rekursif sering kali tidak bisa berhenti sehingga memori akan terpakai habis dan program bisa hang.

Contoh penerapan Rekursif :

a. Faktorial dari bilangan bulat positif n didefinisikan sebagai berikut.

   n! = n x (n-1)! Untuk n > 1

   0! = 1 Untuk n = 0 atau n = 1

   Secara pemrograman dapat ditulis sebagai:

   Faktorial (0) = 1 (1)

   Faktorial (N) = N*Faktorial (N-1) (2)

   Persamaan (2) di atas adalah contoh hubungan rekurens (*recurrence relation*), yang berarti bahwa nilai suatu fungsi dengan argumen tertentu bisa dihitung dari fungsi yang sama dengan argumen yang lebih kecil. Persamaan (1) tidak bersifat rekursif, disebut nilai awal atau basis. Setiap fungsi rekursif paling sedikit mempunyai satu nilai awal, jika tidak fungsi tersebut tidak bisa dihitung secara eksplisit.

b. Bilangan Fibbonacci didefinisikan sebagai berikut

   1 1 2 3 5 8 13 21 34 55 89 …

dari barisan tersebut dapat dilihat bahwa bilangan ke-N (N>2) dalam barisan dapat dicari dari dua bilangan sebelumnya yang terdekat dengan bilangan N, yaitu bilangan ke-(N-1) dan bilangan ke-(N-2), sehingga dapat dirumuskan sebagai :

Fibbonacci (1) = 1 (1)

Fibbonacci (2) = 1 (2)

Fibbonacci (N) = Fibbonacci (N-1) + Fibbonacci (N-2) (3)

Dengan persamaan (1) dan (2) adalah basis dan persamaan (3) adalah rekurensnya. Dalam beberapa situasi, pemecahan secara rekursif maupun secara iteratif mempunyai keuntungan dan kekurangan masing-masing. Cukup sulit untuk menentukan mana yang paling sederhana, paling jelas, paling efisien dan paling mudah dibanding yang lain. Boleh dikatakan pemilihan cara iterative maupun rekursif merupakan kesenangan seorang programmer dan tergantung konteks permasalahan yang akan dipecahkan sesuai dengan kesanggupan yang bersangkutan.

Dalam prakteknya, terdapat dua metode rekursif yaitu Tail recursive dan Nontail recursive. Nontail recursive adalah proses rekursif yang sudah kita bahas sebelumnya. Rekursif Tail adalah proses rekursif dengan pemanggilan rekursif di akhir method dan tidak memiliki aktivitas selama fase balik. Method rekursif yang bukan tail recursive disebut non-tail recursive. Contoh dari tail rekursif adalah :

```
1  void tail(int i) {
2     if (i>0) {
3        print('$i ');
4     tail(i-1);
5     }
6  }
```

Yang bukan merupakan tail rekursif adalah:

```
1  int fact(int x){
2     if (x == 0)
3        return 1;
4     else
5        return x*fact(x-1);
6  }
```

Bukan tail recursive karena pada saat kembali dari recursive call `x*fact(x-1)`, masih terdapat operasi perkalian.

```
1  void nonProg(int i) {
2     if (i>0) {
3        nonProg(i-1);
4        print('$i ');
5        nonProg(i-1);
6     }
7  }
```

Bukan tail rekursif, karena dibaris awal terdapat recursive call. Pada tail recursive, recursive call menjadi statement akhir, dan tidak ada recursive call diatasnya.

Proses mengubah dari Non Tail ke Tail Rekursif

Method non-tail recursive dapat diubah menjadi tail-recursive method dengan menambahkan parameter tambahan untuk menampung hasil.

```
method fact(int n)       fact_aux(int n, int result)
```

Teknik yang digunakan biasanya membuat fungsi tambahan (method fact(int n))

yang memanggil method tail recursive.

```
1  int fact_aux(int n, int result) {
2     if (n == 1)
3        return result;
4     return fact_aux(n - 1, n * result)
5  }
6  int fact(n) {
7     return fact_aux(n, 1);
8  }
```

Method tail-recursive Fibonacci diimplementasikan menggunakan dua parameter bantuan untuk menampung hasil.

```
1  int fib_aux (int n, int next, int result) {
2     if (n == 0)
3        return result;
4     return fib_aux(n - 1, next + result, next);
5  }
6  To calculate fib(n), call fib_aux(n,1,0)
```

C. TUGAS PENDAHULUAN

Jawablah pertanyaan berikut ii:

1. Apa yang dimaksud dengan rekursif?
2. Tuliskan fungsi untuk menghitung nilai faktorial
3. Tuliskan fungsi untuk menampilkan nilai fibonacci dari dere fibonacci
4. Apa yang dimaksud dengan rekursif tail?
5. Tuliskan fungsi untuk menghitung deret fibonacci menggunakan tail rekursif!

D. PERCOBAAN

Percobaan 1 : Fungsi rekursif untuk menghitung nilai faktorial

```
1  import 'dart:io';
2
3  int faktorial(int x) {
4    if (x == 1) {
5      return x;
6    } else {
7      return x * faktorial(x - 1);
8    }
9  }
10 void main() {
11   stdout.write("N = ");
```

```
  int n = int.parse(stdin.readLineSync()!);
  print("Hasil = ${faktorial(n)}");
}
```

Percobaan 2 : Fungsi rekursif untuk menampilkan deret fibonacci

```
int fibbon(int x) {
  if (x <= 0 || x <= 1) {
    return x;
  } else {
    return fibbon(x - 2) + fibbon(x - 1);
  }
}

void main() {
  int n = 10;
  for (int i = 0; i < n; i++) {
    print("f$i = ${fibbon(i)}");
  }
}
```

Percobaan 3 : Fungsi rekursif untuk menentukan bilangan prima atau bukan prima

```
import 'dart:io';

int ambilNilaiRekursif(int number, int index) {
  if (index == 1) {
    return 1;
  } else if (number % index == 0) {
    return 1 + ambilNilaiRekursif(number, --index);
  } else {
    return 0 + ambilNilaiRekursif(number, --index);
  }
}

bool cekBilanganPrima(int num) {
  if (num > 1) {
    return (ambilNilaiRekursif(num, num) == 2);
  } else {
    return false;
  }
}

void main() {
  stdout.write("Masukkan bilangan nya : ");
  int num = int.parse(stdin.readLineSync()!);

  if (cekBilanganPrima(num)) {
    print("Bilangan Prima");
  } else {
    print("Bukan Bilangan Prima");
  }
}
```

Percobaan 4 : Fungsi rekursi untuk menampilkan kombinasi 2 karakter

```
void charCombination(String a, int n) {
  if (n == 0) {
    stdout.write('$a ');
```

```
4     } else {
5       for (int i = 97; i < 99; i++) {
6         charCombination(a + String.fromCharCode(i), n - 1);
7       }
8     }
9   }
10
11  void main() {
12    charCombination("", 2);
    }
```

Percobaan 5 : Fungsi rekursi untuk menghitung pangkat

```
1   import 'dart:io';
2
3   int pangkatrekursif(int x, int y) {
4     if (y == 0) {
5       return 1;
6     } else {
7       return x * pangkatrekursif(x, y - 1);
8     }
9   }
10
11  void main() {
12    stdout.write("Bilangan x pangkat y : \n");
13    stdout.write("Bilangan x : ");
14    int x = int.parse(stdin.readLineSync()!);
15
16    stdout.write("Bilangan y : ");
17    int y = int.parse(stdin.readLineSync()!);
18
19    print('$x dipangkatkan $y = ${pangkatrekursif(x, y)}');
    }
```

Percobaan 5 : Fungsi tail rekursif untuk menampilkan i

```
1   void tail(int i) {
2     if (i > 0) {
3       stdout.write('$i ');
4       tail(i - 1);
5     }
6   }
```

Percobaan 6 : Fungsi tail rekursif untuk menghitung faktorial

```
1   int factAux(int n, int result) {
2     if (n == 1) {
3       return result;
4     }
5     return factAux(n - 1, n * result);
6   }
7
9   int fact(int n) {
10    return factAux(n, 1);
11  }
12
13  void main() {
14    int result = fact(5);
15    print('Faktorial: $result');
```

```
16 }
```

Percobaan 8 : Fungsi tail rekursif untuk menghitung fibonacci

```
1  int fibAux(int n, int next, int result) {
2    if (n == 0) {
3      return result;
4    }
5    return fibAux(n - 1, next + result, next);
6  }
7
8  int fib(int n) {
9    return fibAux(n, 1, 0);
10 }
11
12 void main() {
13   int result = fib(5);
14   print('Deret Fibonacci: $result');
15 }
```

E. LATIHAN

1. Buatlah program rekursif untuk menghitung segitiga Pascal !

   F1 1

   F2 1 1

   F3 1 2 1

   F4 1 3 3 1

   F5 1 4 6 4 1

   F6 1 5 10 10 5 1

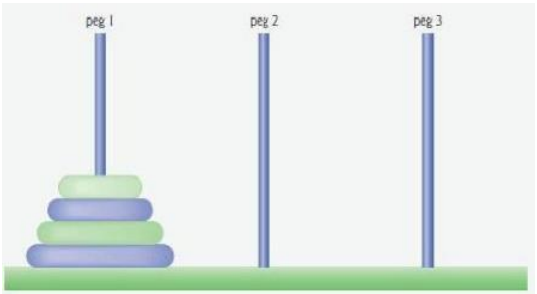
2. Buatlah program secara rekursif, masukkan jumlah N karakter dan cetak dalam semua kombinasi !

   ```
   Jumlah karakter = 3
   aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc caa cab cac cba cbb cbc cca ccb ccc BUILD SUCCESSFUL (total time: 1 second)
   ```

3. Buat program BinarySearch dengan Rekursif ! (data tentukan sendiri):

   ```
   Data : 2,5,8,10,14,32, 35, 41, 67, 88, 90, 101, 109
   Data yang dicari : 10
   Data 10 berada pada indek ke – 3
   ```

4. Buatlah program rekursif untuk memecahkan permasalahan Menara Hanoi !



Gambar 6.1. Menara Hanoi

Program ini merupakan program untuk menampilkan pergerakkan menara hanoi, yang merujuk pada class menaraHanoi. Secara umum algoritma menara hanoi, adalah memindahkan sub menara hanoi dengan n – 1 pin dari n pin ke tiang perantara. Lalu memindahkan pin ke n ke tiang tujuan, lalu memindahkan sub menara hanoi dengan n – 1 pin yang ada di tiang perantara, ke tiang tujuan. StopCase nya jika n == 1.

Jumlah disk : 3

Langkah-langkah nya adalah dengan :

a. Pindahkan disc 1 dari pasak A ke pasak C
b. Pindahkan disc 2 dari pasak A ke pasak B
c. Pindahkan disc 1 dari pasak C ke pasak B
d. Pindahkan disc 3 dari pasak A ke pasak C
e. Pindahkan disc 1 dari pasak B ke pasak A
f. Pindahkan disc 2 dari pasak B ke pasak C
g. Pindahkan disc 1 dari pasak A ke pasak C

5. Jelaskan proses rekursif untuk program dibawah ini !

```
1  void decToBin(int num) {
2    if (num > 0) {
3      decToBin(num ~/ 2);
4      stdout.write('${num % 2}');
5    }
6  }
```

6. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil `test("01101", 4)`!

```
1  int test(String s, int last) {
2    if (last < 0) {
3      return 0;
4    }
5    if (s[last] == "0") {
6      return 2 * test(s, last - 1);
7    }
8    return 1 + 2 * test(s, last - 1);
9  }
```

7. Jelaskan proses rekursif untuk program dibawah ini !

```
bool search(List<int> x, int size, int n) {
  if (size > 0) {
    if (x[size - 1] == n) {
      return true;
    } else {
      return search(x, size - 1, n);
    }
  }
  return false;
}
```

8. Jelaskan proses rekursif untuk program dibawah ini !

```
bool binarySearch(List<int> x, int start, int end, int n) {
  if (end < start) return false;
  int mid = (start + end) ~/ 2;

  if (x[mid] == n) {
    return true;
  } else {
    if (x[mid] < n) {
      return binarySearch(x, mid + 1, end, n);
    } else {
      return binarySearch(x, start, mid - 1, n);
    }
  }
}
```

9. Jelaskan proses rekursif untuk program dibawah ini !

```
int f(int n) {
  if (n == 0) return 0;
  if (n == 1) return 1;
  if (n == 2) return 1;
  return 2 * f(n - 2) + f(n - 3);
}
```

10. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil `square(5)`, `cube(5)`, `cube(123)`?

```
int square(int n) {
  if (n == 0) return 0;
  return square(n - 1) + 2 * n - 1;
}

int cube(int n) {
  if (n == 0) return 0;
  return cube(n - 1) + 3 * (square(n)) - 3 * n + 1;
}
```

11. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini !

```
void decToBin(int num) {
  if (num > 0) {
    decToBin(num ~/ 2);
    stdout.write('${num % 2}');
```

```
5     }
6  }
```

12. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini !

```
1  bool gcdlike(int p, int q) {
2    if (q == 0) {
3      return (p == 1);
4    }
5    return gcdlike(q, p % q);
6  }
```

13. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini !

```
1  int f(int n) {
2    if (n == 0) return 0;
3    if (n == 1) return 1;
4    if (n == 2) return 1;
5    return 2 * f(n - 2) + f(n - 3);
6  }
```

14. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini dengan memanggil `square(5), cube(5), cube(123)?`

```
1  int square(int n) {
2    if (n == 0) return 0;
3    return square(n - 1) + 2 * n - 1;
4  }
```

F. LAPORAN RESMI

Kerjakan hasil percobaan (D) dan latihan (E) di atas dan tambahkan Analisa.

Dikumpulkan H-1 sebelum perkuliahan minggu selanjutnya jam 23.59.

Format Laporan Praktikum: Judul, tujuan, tugas pendahuluan, Percobaan, Latihan dan Pembahasan, Kesimpulan, Referensi