REKURSIF

TUGAS PRAKTIKUM PERTEMUAN KE-3
MATA KULIAH STRUKTUR DATA

Oleh:

MOH. JEVON ATTAILLAH

NIM: 362458302035

PROGRAM STUDI DIPLOMA IV

TEKNOLOGI REKAYASA PERANGKAT LUNAK

POLITEKNIK NEGERI BANYUWANGI

2024

Daftar Isi

A. Tujuan Praktikum

1. Mahasiswa dapat memahami sintask-sintaks rekursif
2. Mahasiswa dapat mengetahui kegunaan dari rekursif
3. Mahasiswa dapat mengetahui cara kerja rekursif

B. Teori Singkat

Rekursif adalah teknik pemograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah, biasanya dengan membagi masalah yang kompleks menjadi sub-masalah yang lebih kecil dan mudah diselesaikan. Fungsi rekursif terdiri dari dua bagian penting: basis dan rekurens.

Basis merupakan kondisi dasar yang menentukan kapan fungsi harus berhenti memanggil dirinya sendiri, sehingga mencegah perulangan tanpa henti dan teerjadinya stuck overflow. Sementara itu, rekurens adalah bagian di mana fungsi memanggil dirinya kembali dengan parameter yang telah dimodifikasi agar mendekati kondisi dasar. Contohnya, perhitungan faktorial didefinisikan sebagai n! = (n-1) dengan 0! =1 sebagai basis, sehingga setiap pemanggilan fungsi mengurangi nilain hingga mencapai kondisi dasar yang sudah ditentukan.

C. Tugas Pendahuluan

1. Apa yang dimaksud dengan rekursif?

   Rekursif adalah suatu proses atau prosedur dari fungsi yang memanggil dirinya sendiri secara berulang-ulang.
2. Tuliskan fungsi untuk menghitung nilai faktorial.

```
import 'dart:io';

int faktorial (int x) {
    if (x == 1) {
return x;
} else {
return x * faktorial(x-1);
}
}
void main() {
stdout.write("N = ");
int n = int.parse(stdin.readLineSync()!);
print("Hasil = $faktorial (n) }");
}
```

3. Tuliskan fungsi untuk menampilkan nilai fibonacci dari deret fibonacci.

```
int fibbon(int x) {
  if (x == 0 || x == 1) { // Kondisi dasar
 return x;
} else {
 return fibbon(x - 1) + fibbon(x - 2); // Rekurens
}
}

void main() {
int n = 10;
```

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

4. Apa yang dimaksud dengan rekursif tail?

   Rekursif Tail adalah proses rekursif dengan pemanggilan rekursif di akhir method dan tidak memiliki aktivitas selama fase balik.

5. Tuliskan fungsi untuk menghitung deret fibonacci menggunakan tail rekursif!

```
void tail(int i) {
if (i>0) {
print('$i ');
tail(i-1);
}
}
```

D. Percobaan

Percobaan 1 : Fungsi rekursif untuk menghitung nilai factorial

```
p1.dart > main
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  }
   Run | Debug
10 void main() {
11   stdout.write("N = ");
12   int n = int.parse(stdin.readLineSync()!);
13   print("Hasil = ${faktorial(n)}");
14 }
```

```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p1.dart
N = 2
Hasil = 2
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p1.dart
N = 5
Hasil = 120
```

Percobaan 2 : Fungsi rekursif untuk menampilkan deret Fibonacci

```
p2.dart > main
1  int fibbon(int x) {
2    if (x <= 0 || x <= 1) {
3      return x;
4    } else {
5      return fibbon(x - 2) + fibbon(x - 1);
6    }
7  }
8
   Run | Debug
9  void main() {
10   int n = 5;
11   for (int i = 0; i < n; i++) {
12     print("f$i = ${fibbon(i)}");
13   }
14 }
```

```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p2.dart
f0 = 0
f1 = 1
f2 = 1
f3 = 2
f4 = 3
f5 = 5
f6 = 8
f7 = 13
f8 = 21
f9 = 34
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p2.dart
f0 = 0
f1 = 1
f2 = 1
f3 = 2
f4 = 3
```

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

```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p3.dart
Masukkan bilangan nya : 3
Bilangan Prima
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p3.dart
Masukkan bilangan nya : 4
Bukan Bilangan Prima
```

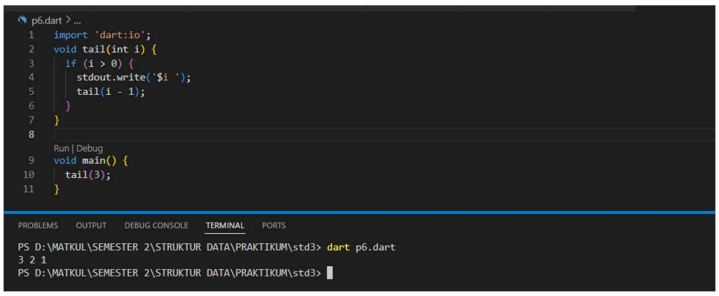
Percobaan 4 : Fungsi rekursi untuk menampilkan kombinasi 2 karakter

```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p4.dart
aa ab ba bb
```

Percobaan 5 : Fungsi rekursi untuk menghitung pangkat

```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p5.dart
Bilangan x pangkat y :
Bilangan x : 2
Bilangan y : 5
2 dipangkatkan 5 = 32
```

Percobaan 6 : Fungsi tail rekursif untuk menampilkan i



```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p6.dart
3 2 1
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3>
```

Percobaan 7 : Fungsi tail rekursif untuk menghitung faktorial

```dart
int factAux(int n, int result) {
  if (n == 1) {
    return result;
  }
  return factAux(n - 1, n * result);
}

int fact(int n) {
  return factAux(n, 1);
}

Run | Debug
void main() {
  int result = fact(5);
  print('Faktorial: $result');
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p7.dart
Faktorial: 120
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3>
```

Percobaan 8 : Fungsi tail rekursif untuk menghitung Fibonacci

```dart
int fibAux(int n, int next, int result) {
if (n == 0) {
return result;
}
return fibAux(n - 1, next + result, next);
}
int fib(int n) {
return fibAux(n, 1, 0);
}
Run | Debug
void main() {
int result = fib(5);
print('Deret Fibonacci: $result');
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart p8.dart
Deret Fibonacci: 5
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3>
```

E. Latihan

Latihan 1

```dart
import 'dart:io';

int segitiga(int row, int col) {
  if (col == 0 || col == row) {
    return 1;
  } else {
    return segitiga(row - 1, col - 1) + segitiga(row - 1, col);
  }
}

Run | Debug
void main() {
  int n = 5; // jumlah baris segitiga Pascal

  for (int i = 0; i < n; i++) {
    stdout.write(' ' * (n - i)); // spasi untuk membentuk piramida
    for (int j = 0; j <= i; j++) {
      stdout.write('${segitiga(i, j)} ');
    }
    print('');
  }
}
```

```
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart l1.dart
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
1 5 10 10 5 1
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3>
```

Analisa

```
int segitiga(int row, int col) {
  if (col == 0 || col == row) {
    return 1;
  } else {
    return segitiga(row - 1, col - 1) + segitiga(row - 1, col);
  }
}
```

Kode ini digunakan untuk mencetak segitiga pascal dalam bentuk piramida. Segitiga pascal adalah sebuah pola angka di mana setiap angka di dalam baris (selain angka 1 di ujung) merupakan hasil penjumlahan dari dua angka yang berada tepat diatasnya

Fungsi segitiga(row, col) ini bertugas menghitung angka yang ada di baris ke-row dan kolom ke-col dari segitiga pascal

Parameter

- row : Baris segitiga (dimulai dari 0)
- col : Kolom baris tersebut (juga dimulai dari 0)

```
  if (col == 0 || col == row) {
    return 1;
```

Logika dasar kondisi tepi yakni jika col == 0 atau col == row, artinya kita berada di ujung segitiga. Pada kasus kedua itu nilai yang harus dikebalikan adalah 1

```
for (int i = 0; i < n; i++) {
  stdout.write(' ' * (n - i));
```

Di sini, untuk setiap baris (dari i = 0 sampai i = 5), kita mencetak sejumlah spasi di awal. Jumlah spasi ini membuat segitiga terlihat **terpusat** seperti piramida.

```
    for (int j = 0; j <= i; j++) {
      stdout.write('${segitiga(i, j)} ');
    }
    print('');
  }
}
```

Untuk setiap baris, kita melakukan perulangan untuk setiap kolom dari j = 0 hingga j = i. Di setiap posisi, kita memanggil fungsi segitiga(i, j) untuk mendapatkan angka yang tepat dan mencetaknya dengan spasi di belakangnya agar angkanya tidak menempel. Setelah selesai dengan satu baris, kita pindah ke baris berikutnya dengan print('').

Latihan 2

```
12.dart > main
1  import 'dart:io';
2
3  void kombinasi(String a, int n) {
4    if (n == 0) {
5      stdout.write("$a ");
6    } else {
7      for (int i = 97; i <= 99; i++) { // 97 adalah a 98=b 99=c
8        kombinasi(a + String.fromCharCode(i), n - 1);
9      }
10   }
11 }
12
   Run | Debug
13 void main() {
14   int n = 3;
15   kombinasi("", n);
16 }
```

```
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3> dart 12.dart
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
PS D:\MATKUL\SEMESTER 2\STRUKTUR DATA\PRAKTIKUM\std3>
```

Analisa

Perulangan ini berjalan dari nilai i = 97 hingga i = 99. Angka-angka ini adalah kode ASCII. Perulangan ini digunakan untuk memilih satu karakter dari 'a', 'b', dan 'c'.

```
(a + String.fromCharCode(i), n - 1);
```

Fungsi String.fromCharCode(i) mengubah kode angka menjadi karakter. Misalnya, ketika i adalah 97, String.fromCharCode(97) menghasilkan 'a'. Parameter a adalah string yang telah terbentuk dari kombinasi karakter sebelumnya. Dengan menambahkan karakter baru ke a, kita secara bertahap membangun kombinasi huruf.

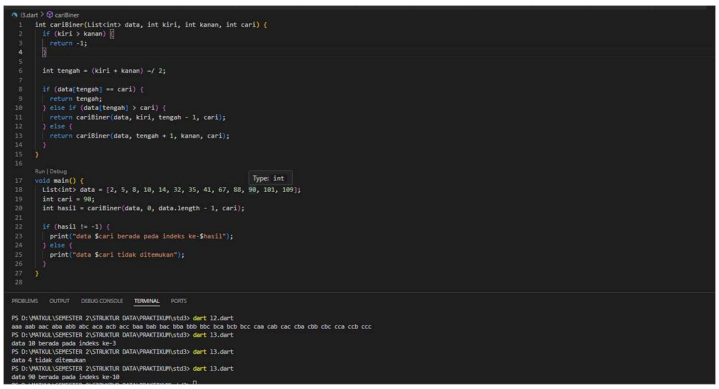
kombinasi(a + String.fromCharCode(i), n - 1);

Setelah menambahkan satu huruf, fungsi kombinasi dipanggil lagi dengan:

- String yang baru (hasil penggabungan a dengan karakter yang baru ditambahkan)
- Nilai n dikurangi 1 (karena satu karakter sudah ditambahkan)

n menunjukkan berapa banyak karakter lagi yang harus ditambahkan untuk mencapai panjang kombinasi yang diinginkan. Jika kita mulai dengan n = 3, maka setelah menambahkan satu huruf, sisa yang harus ditambahkan menjadi 2.

Latihan 3



Analisa

Kode menerapkan algoritma Binary search secara rekursif. Pencarian bineer digunakan untuk mencarti sebuah nilai atau target di dalam sebuah daftar yang sudah diurutkan. Pada kode ini, kita mencari nilai tertentu dalam daftar angka.

Fungsi cari biner memiliki 4 parameter antara lain :

1. data : Daftar angka yang sudah terurut.
2. kiri : Indeks paling kiri (awal) dari bagian daftar yang sedang dicari.
3. kanan : Indeks paling kanan (awal) dari bagian daftar yang sedang dicari.
4. cari : Nilai yang ingin dicari dalam daftar.

```
if (kiri > kanan) {
  return -1;
}
```

Jika nilai indeks kiri lebih besar dari kanan, artinya seluruh bagian daftar sudah dicari dan nilai target tidak ditemukan. Maka, fungsi mengembalikan -1.

```
int tengah = (kiri + kanan) ~/ 2;
```

Operator ~/ melakukan pembagian bulat. Dengan cara ini, kita mendapatkan indeks tengah dari bagian daftar yang sedang dipertimbangkan. Misalnya, jika kiri = 0 dan kanan = 12, maka tengah akan menjadi 6. Jika pembagian tidak menghasilkan bilangan bulat, operator ~/ pada Dart akan menghilangkan bagian desimalnya (atau dengan kata lain, melakukan pembulatan ke bawah). Misalnya, jika kiri = 3 dan kanan = 4, maka : $\frac{3+4}{2} = 3.5$, operator ~/ menghasilkan 3 karena ia hanya mengabil bagian bilangan bulatnya.

```
if (data[tengah] == cari) {
  return tengah;
```

Jika nilai pada indeks tengah sama dengan nilai yang dicari (cari), maka fungsi langsung mengembalikan indeks tengah karena target telah ditemukan.

```
} else if (data[tengah] > cari) {
  return cariBiner(data, kiri, tengah - 1, cari);
```

Jika nilai di indeks tengah lebih besar dari target, maka target harus berada di **separuh kiri** dari daftar (karena daftar sudah diurutkan). Maka, kita panggil kembali fungsi cariBiner dengan batas kanan baru yaitu tengah - 1.

```
} else {
  return cariBiner(data, tengah + 1, kanan, cari);
```

Jika nilai di indeks tengah lebih kecil dari target, maka target berada di **separuh kanan**. Jadi, kita memanggil fungsi cariBiner lagi dengan batas kiri baru yaitu tengah + 1.

F. Kesimpulan

Dari praktikum ini, dapat disimpulkan bahwa rekursif merupakan teknik pemrograman yang memungkinkan suatu fungsi untuk memanggil dirinya sendiri secara berulang guna menyelesaikan masalah yang kompleks dengan membaginya menjadi sub-masalah yang lebih

kecil. penerapan rekursif sangat berguna dalam menyelesaikan permasalahan seperti faktorial, fibonacci, dan binary search, di mana pendekatan ini dapat membuat kode lebih sederhana dan elegan. namun, kelemahan rekursif adalah penggunaan memori yang lebih besar dibandingkan iteratif, terutama jika tidak dikelola dengan baik, yang dapat menyebabkan stack overflow. oleh karena itu, penting untuk memahami konsep dasar rekursif seperti kondisi basis, rekurens, serta perbedaan antara tail-recursive dan non-tail-recursive agar dapat mengoptimalkan penggunaannya dalam pemrograman.

G. Referensi

- lutfi hakim. modul struktur data 01 - rekursif. politeknik negeri banyuwangi, 2024.
- yuliana, praktikum 7 - rekursif 1. politeknik elektronik negeri surabaya (pens), 2015. tersedia di: https://yuliana.lecturer.pens.ac.id/Struktur%20Data/PRAKTIKUM%202015/Praktikum%207%20-%20Rekursif%201.pdf.
- kumparan. segitiga pascal: pengertian, pola bilangan, dan cara menghitung. tersedia di: https://kumparan.com/kabar-harian/segitiga-pascal-pengertian-pola-bilangan-dan-cara-menghitung-1wtHwD50ioE.
- martha dwi. segitiga pascal dalam bahasa dart. tersedia di: https://github.com/Martha-Dwi/rekursif/blob/main/segitigaPascal.dart.
- chatgpt. openai, 2025.

H. Salinan Kode

Percobaan 1

```
import 'dart:io';


int faktorial (int x) {
if (x == 1) {
 return x;
} else {
 return x * faktorial(x - 1);
}
}
void main() {
 stdout.write("N = ");
 int n = int.parse(stdin.readLineSync()!);
 print("Hasil = ${faktorial(n)}");
}
```

Percobaan 2

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

void main() {
  int n = 5;
  for (int i = 0; i < n; i++) {
    print("f$i = ${fibbon(i)}");
  }
}
```

Percobaan 3

```
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

```
import 'dart:io';
void charCombination(String a, int n) {
  if (n == 0) {
    stdout.write('$a ');
  } else {
    for (int i = 97; i < 99; i++) {
      charCombination(a + String.fromCharCode(i), n - 1);
    }
  }
}

void main() {
  charCombination("", 2);
}
```

Percobaan 5

```
import 'dart:io';
```

```
int pangkatrekursif(int x, int y) {
  if (y == 0) {
    return 1;
  } else {
    return x * pangkatrekursif(x, y - 1);
  }
}

void main() {
  stdout.write("Bilangan x pangkat y : \n");
  stdout.write("Bilangan x : ");
  int x = int.parse(stdin.readLineSync()!);

  stdout.write("Bilangan y : ");
  int y = int.parse(stdin.readLineSync()!);

  print('$x dipangkatkan $y = ${pangkatrekursif(x, y)}');
}
```

Percobaan 6

```
import 'dart:io';
void tail(int i) {
  if (i > 0) {
    stdout.write('$i ');
    tail(i - 1);
  }
}

void main() {
  tail(3);
}
```

Percobaan 7

```
int factAux(int n, int result) {
  if (n == 1) {
    return result;
  }
  return factAux(n - 1, n * result);
}

int fact(int n) {
  return factAux(n, 1);
}

void main() {
  int result = fact(5);
  print('Faktorial: $result');
}
```

Percobaan 8

```
int fibAux(int n, int next, int result) {
if (n == 0) {
return result;
}
return fibAux(n - 1, next + result, next);
}
int fib(int n) {
return fibAux(n, 1, 0);
}
void main() {
int result = fib(5);
print('Deret Fibonacci: $result');
}
```

Latihan 1

```
import 'dart:io';
```

```
int segitiga(int row, int col) {
  if (col == 0 || col == row) {
    return 1;
  } else {
    return segitiga(row - 1, col - 1) + segitiga(row - 1, col);
  }
}

void main() {
  int n = 6;
  for (int i = 0; i < n; i++) {
    stdout.write(' ' * (n - i)); // spasi untuk membentuk piramida
    for (int j = 0; j <= i; j++) {
      stdout.write('${segitiga(i, j)} ');
    }
    print('');
  }
}
```

Latihan 2

```
import 'dart:io';

void kombinasi(String a, int n) {
  if (n == 0) {
    stdout.write("$a ");
  } else {
    for (int i = 97; i <= 99; i++) { // 97 adalah a 98=b 99=c
      kombinasi(a + String.fromCharCode(i), n - 1);
    }
  }
}

void main() {
```

```
  int n = 3;
  kombinasi("", n);
}
```

Latihan 3

```
int cariBiner(List<int> data, int kiri, int kanan, int cari) {
  if (kiri > kanan) {
    return -1;
  }

  int tengah = (kiri + kanan) ~/ 2;

  if (data[tengah] == cari) {
    return tengah;
  } else if (data[tengah] > cari) {
    return cariBiner(data, kiri, tengah - 1, cari);
  } else {
    return cariBiner(data, tengah + 1, kanan, cari);
  }
}

void main() {
  List<int> data = [2, 5, 8, 10, 14, 32, 35, 41, 67, 88, 90, 101, 109];
  int cari = 90;
  int hasil = cariBiner(data, 0, data.length - 1, cari);

  if (hasil != -1) {
    print("data $cari berada pada indeks ke-$hasil");
  } else {
    print("data $cari tidak ditemukan");
  }
}
```