LAPORAN

DOSEN PENGAMPU: LUTFI HAKIM

DISUSUN OLEH:

DEVI KHALIMATUS SAKDIA

NIM: 362458302081

KELAS: 1C TEKNOLOGI REKAYASA PERANGKAT LUNAK

PROGRAM STUDI TEKNOLOGI REKAYASA PERANGKAT LUNAK (TRPL)

JURUSAN BISNIS DAN INFORMATIKA

POLITEKNIK NEGERI BANYUWANGI

TAHUN 2025/2026

Tujuan Praktikum

Tujuan dari praktikum ini adalah:

1. Memahami konsep dari fungsi rekursif
2. Mampu mengimplementasikan fungsi rekursif dalam beberapa kasus
3. Mampu membedakan penyelesaian masalah menggunakan fungsi rekursif dan fungsi interaktif
4. Mampu mengimplementasikan fungsi rekursif tail

Teori Singkat

Rekursif adalah suatu proses atau prosedur dari fungsi yang memanggil dirinya sendiri secara berulangulang. 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.

Tugas pendahuluan

Jawablah pertanyaan berikut

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!

Jawaban:

1. Cara menyelesaikan masalah dengan memecahnya menjadi masalah yang lebih kecil dan serupa, di mana solusi untuk masalah yang lebih besar bergantung pada solusi masalah yang lebih kecil tersebut. Ini melibatkan fungsi yang memanggil dirinya sendiri sampai kondisi tertentu terpenuhi.
2. Ada beberapa cara mengitung nilai faktorial yaitu rekursif dan Interatif
   Penjelasan factorial rekursif

- Fungsi faktorialRekursif menerima sebuah bilangan bulat n sebagai input.
- Jika n sama dengan 0, fungsi mengembalikan 1 (kasus dasar).
- Jika tidak, fungsi mengembalikan n dikalikan dengan hasil pemanggilan faktorialRekursif dengan n - 1.

Penjelasan factorial interatif

- ungsi faktorialIteratif juga menerima sebuah bilangan bulat n sebagai input.
- Sebuah variabel hasil diinisialisasi dengan 1.
- Loop for digunakan untuk mengalikan hasil dengan setiap bilangan bulat dari 1 hingga n.
- Hasil akhir dikembalikan.

3. Fungsi untuk menampilkan nilai

- Fungsi fibonacciIteratif juga menerima sebuah bilangan bulat n sebagai input.
- Jika n kurang dari atau sama dengan 1, fungsi mengembalikan n.
- Variabel a dan b diinisialisasi dengan 0 dan 1, masing-masing.
- Loop for digunakan untuk menghitung nilai Fibonacci secara berurutan.
- Nilai akhir dikembalikan.

4. Rekursif tail (tail recursion) adalah bentuk khusus dari rekursi di mana pemanggilan rekursif adalah operasi terakhir yang dilakukan dalam sebuah fungsi. Ini berarti bahwa hasil dari pemanggilan rekursif langsung dikembalikan tanpa perlu melakukan operasi tambahan.

5. Fungsi fibonacci_tail (atau fibonacciTail) menggunakan tiga parameter:

- n: Bilangan Fibonacci yang ingin dihitung.
- a: Nilai Fibonacci sebelumnya (diinisialisasi dengan 0).
- b: Nilai Fibonacci saat ini (diinisialisasi dengan 1).

➢ Kasus dasar:

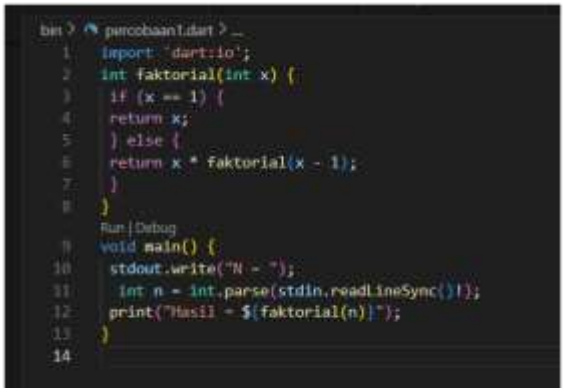
- Jika n adalah 0, fungsi mengembalikan a.
- Jika n adalah 1, fungsi mengembalikan b.

➢ Kasus rekursif:

- Fungsi memanggil dirinya sendiri dengan n-1, b sebagai nilai a baru, dan a+b sebagai nilai b baru.
- Dengan cara ini, hasil perhitungan dibawa dalam parameter fungsi, sehingga pemanggilan rekursif adalah operasi terakhir.

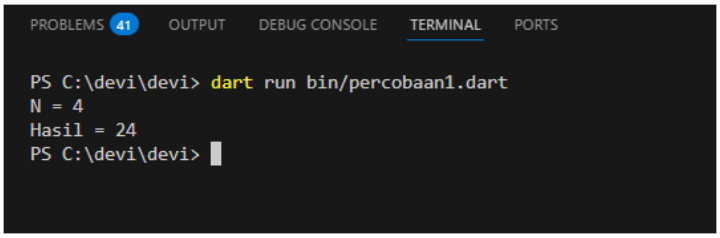
Percobaan

1. Fungsi rekursif untuk menghitung nilai faktorial



```dart
import 'dart:io';
int faktorial(int x) {
 if (x == 1) {
 return x;
 } else {
 return x * faktorial(x - 1);
 }
}
Run | Debug
void main() {
 stdout.write("N = ");
  int n = int.parse(stdin.readLineSync()!);
 print("Hasil = ${faktorial(n)}");
}
```

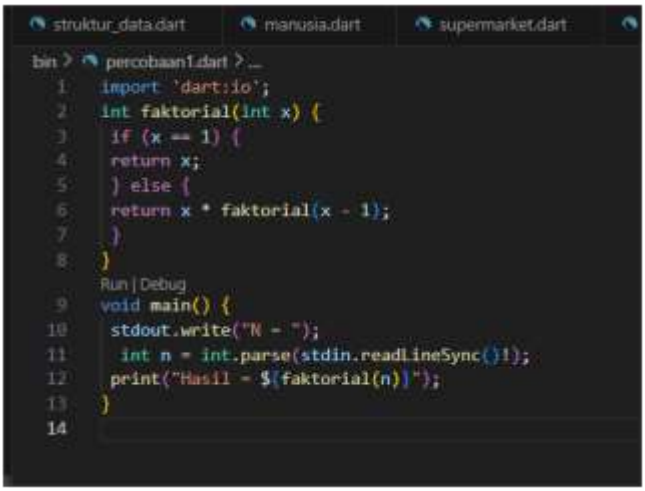
Hasil:



```
PROBLEMS 41  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\devi\devi> dart run bin/percobaan1.dart
N = 4
Hasil = 24
PS C:\devi\devi>
```

2. Fungsi rekursif untuk menampilkan deret Fibonacci



```dart
import 'dart:io';
int faktorial(int x) {
 if (x == 1) {
 return x;
 } else {
 return x * faktorial(x - 1);
 }
}
Run | Debug
void main() {
 stdout.write("N = ");
  int n = int.parse(stdin.readLineSync()!);
 print("Hasil = ${faktorial(n)}");
}
```

Hasil :

```
PS C:\devi\devi> dart run bin/percobaan2.dart
f0 = 0
f1 = 1
f2 = 1
f3 = 2
f4 = 3
f5 = 5
f6 = 8
f7 = 13
f8 = 21
f9 = 34
PS C:\devi\devi>
```

3. Fungsi rekursif untuk menentukan bilangan prima atau bukan prima

```dart
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;
 }
}
Run | Debug
void main() {
 stdout.write("Masukkan bilangan nya : ");
 int num = int.parse(stdin.readLineSync()!);
 if (cekBilanganPrima(num)) {
 print("Bilangan Prima");
 } else {
 print("Bukan Bilangan Prima");
 }
}
```

Hasil :

```
PROBLEMS 41  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\devi\devi> dart run bin/percobaan3.dart
Masukkan bilangan nya : 15
Bukan Bilangan Prima
PS C:\devi\devi>
```

4. : Fungsi rekursi untuk menampilkan kombinasi 2 karakter

```dart
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);
     }
   }
}
Run | Debug
void main() {
 charCombination("", 2);
}
```

5. Fungsi rekursi untuk menghitung pangkat

```dart
import 'dart:io';
int pangkatrekursif(int x, int y) {
 if (y == 0) {
 return 1;
 } else {
 return x * pangkatrekursif(x, y - 1);
 }
}
Run | Debug
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 = ${pangkatrekursif(x, y)}');
}
void tail(int i) {
 if (i > 0) {
 stdout.write('$i ');
 tail(i - 1);
 }
}
```

Hasil :

```
PS C:\devi\devi> dart run bin/percobaan5.dart
Bilangan x pangkat y :
Bilangan x : 15
Bilangan y : 3
15 dipangkatkan 3 = 3375
PS C:\devi\devi>
```

6. Fungsi tail rekursif untuk menghitung factorial

```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');
}
```

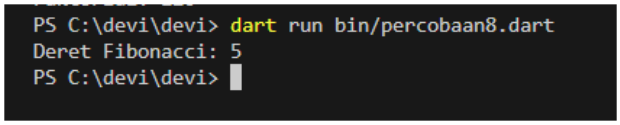
Hasil:

```
PROBLEMS 40  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\devi\devi> dart run bin/percobaan6.dart
Faktorial: 120
PS C:\devi\devi>
```

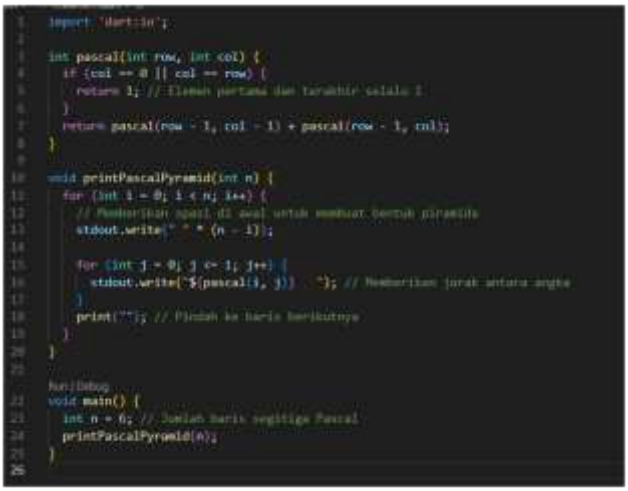
7. Fungsi tail rekursif untuk menghitung Fibonacci

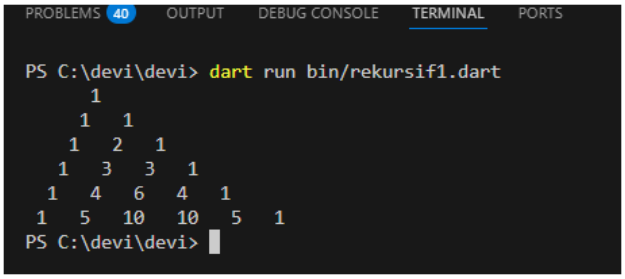
Hasil:



Latian dan Pembahasan

1. Hasil





Codingan:

import 'dart:io';

```
int pascal(int row, int col) {
  if (col == 0 || col == row) {
    return 1; // Elemen pertama dan terakhir selalu 1
  }
  return pascal(row - 1, col - 1) + pascal(row - 1, col);
}

void printPascalPyramid(int n) {
  for (int i = 0; i < n; i++) {
    // Memberikan spasi di awal untuk membuat bentuk piramida
    stdout.write(" " * (n - i));

    for (int j = 0; j <= i; j++) {
      stdout.write("${pascal(i, j)}    "); // Memberikan jarak antara angka
    }
    print(""); // Pindah ke baris berikutnya
  }
}

void main() {
  int n = 6; // Jumlah baris segitiga Pascal
  printPascalPyramid(n);
}
```

Penjelasannya :

Fungsi pascal(baris, kolom):

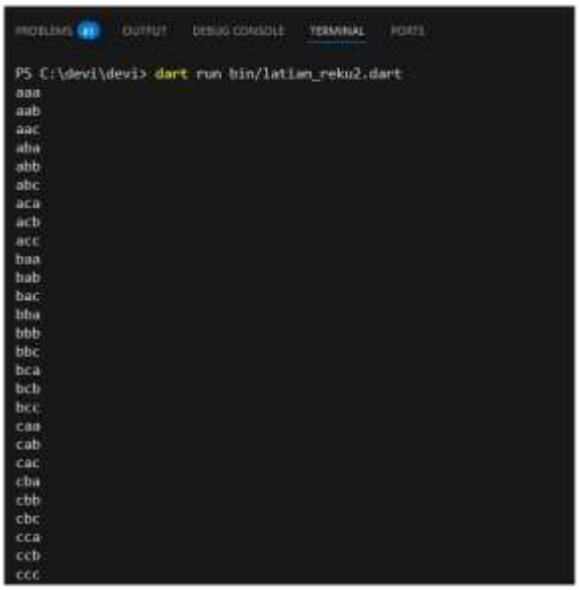
- Fungsi ini menghitung nilai elemen pada segitiga Pascal menggunakan rekursi.
- Kasus dasar: Jika kolom adalah 0 atau kolom sama dengan baris, nilai elemen adalah 1.
- Kasus rekursif: Jika tidak, nilai elemen adalah jumlah dari elemen di atasnya (baris-1, kolom-1) dan elemen di atasnya di sebelah kanan (baris-1, kolom).

Fungsi cetak_segitiga_pascal(tinggi):

- Fungsi ini mencetak segitiga Pascal dengan tinggi yang ditentukan.
- Menggunakan perulangan bersarang untuk mencetak setiap baris dan kolom.
- Memanggil fungsi pascal() untuk mendapatkan nilai setiap elemen.

2. Hasil

Hasilnya:



Codingan:

```
void cetakKombinasi(String prefix, int n, List<String> karakter) {
  if (n == 0) {
    print(prefix);
    return;
  }
```

```
  for (int i = 0; i < karakter.length; i++) {
    cetakKombinasi(prefix + karakter[i], n - 1, karakter);
  }
}

void main() {
  int jumlahKarakter = 3;
  List<String> karakter = ['a', 'b', 'c']; // Daftar karakter yang akan digunakan

  cetakKombinasi('', jumlahKarakter, karakter);
}
```

Penjelasan:

Fungsi kombinasi_karakter(n, karakter=""):

- Fungsi ini adalah fungsi rekursif yang menghasilkan kombinasi karakter.
- **Basis kasus**: Jika panjang karakter sudah sama dengan n, artinya kombinasi sudah lengkap, maka cetak kombinasi tersebut dan fungsi berhenti.
- **Kasus rekursif**: Jika panjang karakter masih kurang dari n, maka lakukan perulangan untuk setiap karakter yang mungkin (dalam contoh ini, "abc").
- Untuk setiap karakter, panggil fungsi kombinasi_karakter lagi dengan menambahkan karakter tersebut ke karakter yang sedang dibangun.

Contoh penggunaan:

- jumlah_karakter menentukan panjang kombinasi yang diinginkan.
- kombinasi_karakter(jumlah_karakter) memulai proses rekursif.
- print("\nBUILD SUCCESSFUL") mencetak pesan "BUILD SUCCESSFUL" setelah semua kombinasi dicetak.

3.Hasil

```dart
int binarySearchRekursif(List<int> data, int target, int kiri, int kanan) {
  if (kiri > kanan) {
    return -1; // Target tidak ditemukan
  }

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

  if (data[tengah] == target) {
    return tengah; // Target ditemukan pada indeks tengah
  } else if (data[tengah] < target) {
    return binarySearchRekursif(data, target, tengah + 1, kanan); // Cari di bagian kanan
  } else {
    return binarySearchRekursif(data, target, kiri, tengah - 1); // Cari di bagian kiri
  }
}

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

  if (hasil != -1) {
    print('Data $target berada pada indeks ke-$hasil');
  } else {
    print('Data $target tidak ditemukan');
  }
}
```

Hasil:

```
PROBLEMS 41   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\devi\devi> dart run bin/rekursif3.dart
Data 10 berada pada indeks ke-3
PS C:\devi\devi>
```

Codingan :

```
int binarySearchRekursif(List<int> data, int target, int kiri, int kanan) {
  if (kiri > kanan) {
    return -1; // Target tidak ditemukan
  }

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

  if (data[tengah] == target) {
    return tengah; // Target ditemukan pada indeks tengah
  } else if (data[tengah] < target) {
    return binarySearchRekursif(data, target, tengah + 1, kanan); // Cari di bagian kanan
  } else {
    return binarySearchRekursif(data, target, kiri, tengah - 1); // Cari di bagian kiri
  }
}
```

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

  if (hasil != -1) {
    print('Data $target berada pada indeks ke-$hasil');
  } else {
    print('Data $target tidak ditemukan');
  }
}
```

Penjelasan:

Fungsi binarySearchRekursif(List<int> data, int target, int kiri, int kanan):

- Fungsi ini adalah implementasi rekursif dari Binary Search.
- data: List data yang sudah terurut.
- target: Data yang ingin dicari.
- kiri: Indeks awal pencarian.
- kanan: Indeks akhir pencarian.
- Kasus dasar: Jika kiri > kanan, artinya data tidak ditemukan, maka kembalikan -1.
- Hitung indeks tengah.
- Jika data[tengah] == target, artinya data ditemukan, maka kembalikan tengah.
- Jika data[tengah] < target, artinya data yang dicari berada di bagian kanan, maka panggil fungsi binarySearchRekursif secara rekursif dengan kiri = tengah + 1.
- Jika data[tengah] > target, artinya data yang dicari berada di bagian kiri, maka panggil fungsi binarySearchRekursif secara rekursif dengan kanan = tengah - 1.

Fungsi main():

- Fungsi main() adalah titik masuk program Dart.
- data: List data yang sudah terurut.
- target: Data yang ingin dicari.
- Panggil fungsi binarySearchRekursif untuk mencari data.
- Cetak hasil pencarian.

Kesimpulan

Dari praktikum ini adalah fungsi rekursif adalah fungsi yang memanggil dirinya sendiri secara berulang-ulang. Fungsi ini terdiri dari dua bagian penting:

- Kasus dasar: Kondisi yang menghentikan pemanggilan rekursif.
- Kasus rekursif: Bagian di mana fungsi memanggil dirinya sendiri.

Keuntungan dari fungsi rekursif meliputi:

- Kode yang lebih singkat dan mudah dibaca, terutama untuk masalah yang memiliki struktur rekursif alami.
- Kemampuan untuk memecahkan masalah kompleks menjadi masalah yang lebih kecil dan serupa.

Namun, fungsi rekursif juga memiliki kekurangan:

- Penggunaan memori yang lebih besar karena setiap pemanggilan fungsi membutuhkan ruang memori tambahan.
- Risiko stack overflow jika rekursi terlalu dalam.

Fungsi rekursif dapat diimplementasikan dalam berbagai kasus, seperti perhitungan faktorial, deret Fibonacci, pencarian biner, dan pembangkitan kombinasi. Pemahaman yang baik tentang konsep rekursi dan kemampuan untuk mengidentifikasi kasus dasar dan kasus rekursif sangat penting untuk mengimplementasikan fungsi rekursif dengan benar.

Selain itu, terdapat variasi dari rekursi yaitu rekursi ekor (tail recursion) di mana panggilan rekursif adalah operasi terakhir yang dilakukan dalam fungsi. Rekursi ekor dapat dioptimalkan oleh beberapa kompiler untuk mengurangi penggunaan memori.