**LAPORAN PRAKTIKUM REKURSIF**

**STRUKTUR DATA**

****

Dosen pengampu :

**Lutfi Hakim S.Pd, M.Kom**

Disusun oleh:

Nama : Nikmatul Wafhiroh

NIM : 362458302032

Kelas : 1C TRPL

**PROGRAM STUDI**

**DIV TEKNOLOGI REKAYASA PERANGKAT LUNAK**

**POLITEKNIK NEGERI BANYUWANGI**

**2025**

1. KOMPETENSI DASAR
2. Memahami mengenai konsep rekursif
3. Mampu memecahkan permasalahan dengan konsep rekursif
4. DASAR TEORI

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. Rekursif sering kali tidak bisa berhenti sehingga memori akan terpakai habis dan program bisa hang.

1. TUJUAN
2. Rekursi seringkali membuat kode lebih singkat dan lebih mudah dibaca dibandingkan dengan solusi iteratif.
3. Rekursi memungkinkan kita untuk memecah masalah besar menjadi bagian-bagian yang lebih kecil dan lebih mudah dipecahkan.
4. Dalam beberapa kasus, menggunakan rekursi dapat menghindari penggunaan loop yang rumit dan mengurangi kemungkinan kesalahan.
5. 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!

JAWABAN

1. Rekursif adalah suatu metode dalam pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan submasalah dari suatu masalah yang lebih besar. Rekursi sering digunakan untuk menyelesaikan masalah yang dapat dipecah menjadi masalah yang lebih kecil dengan cara yang sama.
2. Berikut adalah contoh fungsi dalam Python untuk menghitung nilai faktorial:

def faktorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* faktorial(n - 1)

1. Tuliskan fungsi untuk menampilkan nilai fibonacci dari deret Fibonacci

Berikut adalah contoh fungsi dalam Dart untuk menampilkan nilai Fibonacci:

def fibonacci(n):

if n <= 0:

return []

elif n == 1:

return [0]

elif n == 2:

return [0, 1]

else:

fib = fibonacci(n - 1)

fib.append(fib[-1] + fib[-2])

return fib

1. Rekursif tail (tail recursion) adalah jenis rekursi di mana pemanggilan fungsi rekursif adalah operasi terakhir yang dilakukan dalam fungsi tersebut. Dalam tail recursion, tidak ada operasi tambahan setelah pemanggilan fungsi rekursif. Ini memungkinkan compiler atau interpreter untuk mengoptimasinya dan menghindari penggunaan tumpukan tambahan, sehingga lebih efisien dalam penggunaan memori.
2. Tuliskan fungsi untuk menghitung deret fibonacci menggunakan tail rekursif

Berikut adalah contoh fungsi dalam Dart untuk menghitung deret Fibonacci menggunakan tail recursion:

int fibonacciTailRecursive(int n, [int a = 0, int b = 1]) {

  if (n == 0) {

    return a;

  } else if (n == 1) {

    return b;

  } else {

    return fibonacciTailRecursive(n - 1, b, a + b);

  }

}

void main() {

  int n = 10;

  print('Fibonacci ke-$n adalah: ${fibonacciTailRecursive(n)}');

}

1. PERCOBAAN

**Percobaan 1 :** Fungsi rekursif untuk menghitung nilai faktorial

import 'dart:io';

int faktorial(int x) {

 if (x == 1) {

 return x;

 } else {

 return x \* faktorial(x - 1);

 }

}

  void main() {

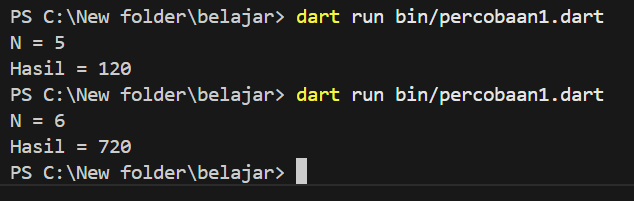
 stdout.write("N = ");

  int n = int.parse(stdin.readLineSync()!);

 print("Hasil = ${faktorial(n)}");

 }

**Hasil pada terminalnya**

****

**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;

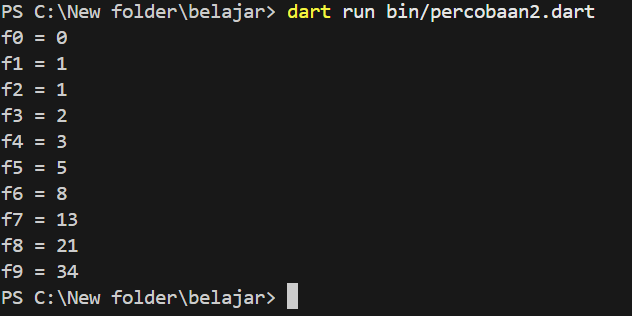
 for (int i = 0; i < n; i++) {

 print("f$i = ${fibbon(i)}");

 }

}

**Hasil pada terminalnya**



**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");

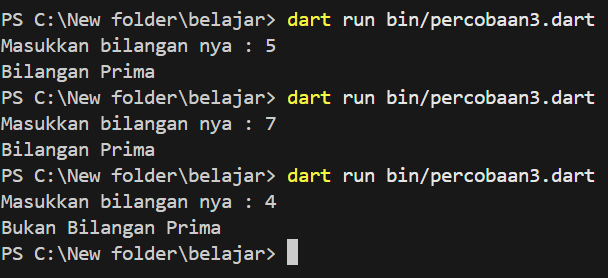
 } else {

 print("Bukan Bilangan Prima");

 }

}

**Hasil pada terminalnya**

****

**Percobaan 4 :** Fungsi rekursi untuk menampilkan kombinasi 2 karakter

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);

}

**Hasil pada terminalnya**

****

**Percobaan 5 :** Fungsi rekursi untuk menghitung pangkat

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 : ");

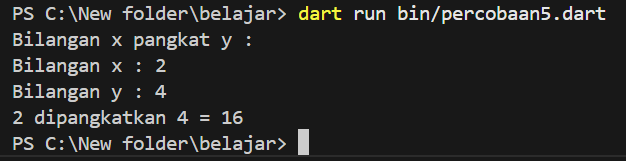
 int x = int.parse(stdin.readLineSync()!);

 stdout.write("Bilangan y : ");

 int y = int.parse(stdin.readLineSync()!);

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

}

**Hasil pada terminalnya**

**Percobaan 6 :** Fungsi tail rekursif untuk menampilkan I

 if (i > 0) {

 stdout.write('$i ');

 tail(i - 1);

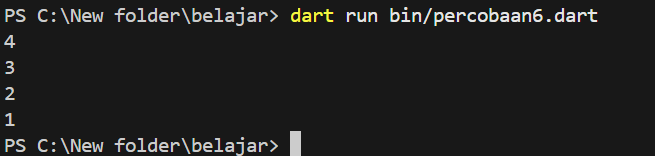
 }

}

void main() {

  tail(4); // Misalkan kita memanggil tail dengan nilai

}

**Hasil pada terminalnya**

**Percobaan 7 :** Fungsi tail rekursif untuk menghitung faktorial

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');

 }

**Hasil pada terminalnya**

**Percobaan 8 :** : Fungsi tail rekursif untuk menghitung Fibonacci

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() {

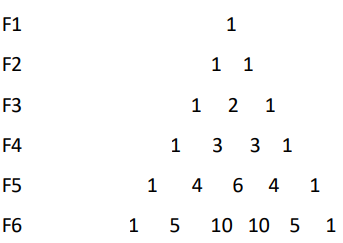
 int result = fib(5);

 print('Deret Fibonacci: $result');

}

**Hasil pada terminalnya**

****

1. LATIHAN
2.  Buatlah program rekursif untuk menghitung segitiga Pascal !

Contoh penggunaan rekursif untuk menghitung segitiga pascal

void main() {

  int numRows = 6;  // Menentukan jumlah baris segitiga Pascal

  List<List<int>> triangle = generatePascalTriangle(numRows);

  printPascalTriangle(triangle);

}

List<List<int>> generatePascalTriangle(int rows) {

  List<List<int>> triangle = [];

  for (int i = 0; i < rows; i++) {

    List<int> row = [];

    for (int j = 0; j <= i; j++) {

      // Menghitung nilai dengan rekursi

      row.add(pascalTriangle(i, j));

    }

    triangle.add(row);

  }

  return triangle;

}

int pascalTriangle(int n, int k) {

  // Kondisi dasar: jika k adalah 0 atau k sama dengan n, nilai adalah 1

  if (k == 0 || k == n) {

    return 1;

  }

  // Rekursi: menghitung nilai dengan menjumlahkan dua nilai dari baris sebelumnya

  return pascalTriangle(n - 1, k - 1) + pascalTriangle(n - 1, k);

}

void printPascalTriangle(List<List<int>> triangle) {

  for (int i = 0; i < triangle.length; i++) {

    // Mencetak label F1, F2, dst.

    print('F${i + 1}: ${triangle[i].join(' ')}');

  }

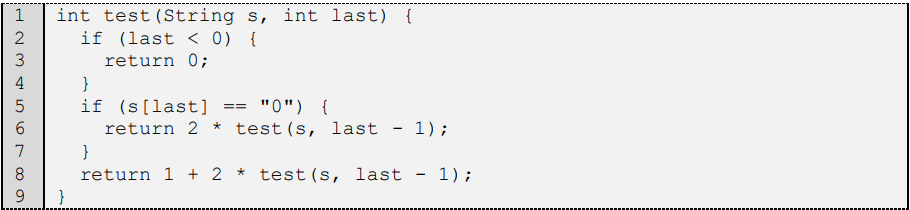
}

**Hasil pada terminalnya**



**Pembahasan soal**

Program ini menghitung nilai Segitiga Pascal dengan fungsi rekursif pascalTriangle, yang menerima dua parameter: row (baris) dan col (kolom). Jika col 0 atau sama dengan row, fungsi mengembalikan 1. Jika tidak, fungsi akan memanggil dirinya sendiri dua kali untuk mendapatkan nilai dari dua baris sebelumnya. Di fungsi main, ditentukan jumlah baris yang ingin ditampilkan, dan dua loop digunakan untuk mencetak nilai-nilai Segitiga Pascal. Program ini dapat menghasilkan hingga 5 baris, dan Anda bisa mengubah nilai rows untuk menampilkan lebih banyak baris

1. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil test(“01101”, 4)!

Proses penggunaan rekursif memanggil test(“01101,4)

int test(String s, int last) {

 if (last < 0) {

    return 0; // Kondisi dasar: jika last kurang dari 0, kembalikan

  }

  if (s[last] == "0") {

    return 2 \* test(s, last - 1); // Jika karakter saat ini adalah '0'

  }

  return 1 + 2 \* test(s, last - 1); // Jika karakter saat ini adalah '1'

}

void main() {

  String input = "01101";

  int result = test(input, 4); // Memanggil dengan indeks terakhir 4

  print('Hasil: $result');

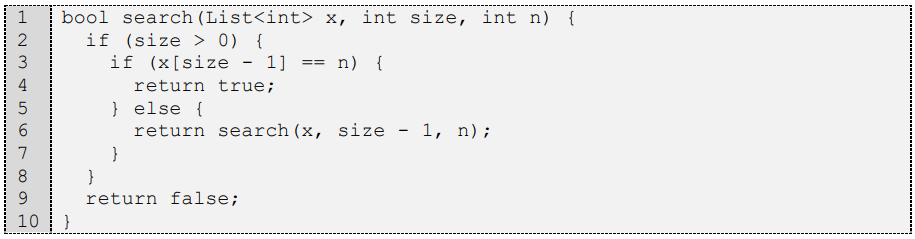
}

**Hasil pada terminalnya**



**Pembahasan soal**

Fungsi test menghitung nilai dari string "01101" berdasarkan karakter '0' dan '1'. Saat dipanggil dengan indeks 4, fungsi memeriksa karakter dari indeks 4 hingga 0. Jika karakter adalah '0', hasil berikutnya dikalikan 2. Jika karakter adalah '1', 1 ditambahkan ke hasil berikutnya dan juga dikalikan 2. Proses ini berlanjut hingga indeks kurang dari 0, yang mengembalikan 0. Setelah semua perhitungan, hasil untuk test("01101", 4) adalah 13, menunjukkan bahwa karakter '1' memberikan nilai lebih tinggi daripada '0'.

1. ****Jelaskan proses rekursif untuk program dibawah ini !

Proses penggunaan rekursif pada program diatas

* Proses yang ditemukan di dalam daftar

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;

}

void main() {

  List<int> numbers = [6, 7, 8, 9, 10];

  int valueToFind = 8;

  bool found = search(numbers, numbers.length, valueToFind);

  if (found) {

    print('$valueToFind ditemukan dalam daftar.');

  } else {

    print('$valueToFind tidak ditemukan dalam daftar.');

  }

}

bool find(List<int> x, int size, int n) {

  if (size > 0) {

    if (x[size - 1] == n) {

      return true;

    } else {

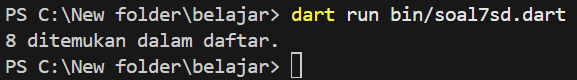
      return search(x, size - 1, n);

    }

  }

  return false;

}

**** **Hasil pada terminalnya**

**Penjelasannya**

* **Daftar**: numbers adalah daftar yang berisi angka 6 sampai 10
* **Nilai yang Dicari**: valueToFind adalah nilai yang ingin dicari, yaitu 8
* **Memanggil Fungsi:** Fungsi search dipanggil dengan daftar, ukuran daftar, dan nilai yang dicari.
* **Hasil**: Jika nilai ditemukan, program mencetak pesan bahwa nilai ditemukan seperti diatas 8 ditemukan dalam daftar.
* Proses yang tidak ditemukan pada daftar

void main() {

  List<int> numbers = [6, 7, 8, 9, 10];

  int valueToFind = 5;

  bool found = search(numbers, numbers.length, valueToFind);

  if (found) {

    print('$valueToFind ditemukan dalam daftar.');

  } else {

    print('$valueToFind tidak ditemukan dalam daftar.');

  }

}

bool find(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;

}

** Hasil pada terminalnya**

**Penjelasan**

Proses tersebut sama seperti yg diatas, dimana daftar number nya 6 sampai 10, yang membedakan hanya nilai yg dicari. angka 5 tidak terdaftar pada number sehinggan pada saat mengeksekusi pada bagian terminal nilai 5 tidak ditemukan

1. 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

Proses membuat program BinarySearch dengan rekursif dimana program ini akan mencari 10 angka dalam data yang telah ditentukan

void main() {

  List<int> data = [2, 5, 8, 10, 14, 32, 35, 41, 67, 88, 90, 101, 109];

  int target = 10;

  int result = binarySearch(data, target, 0, data.length - 1);

  if (result != -1) {

    print('Data $target berada pada indeks ke - $result');

  } else {

    print('Data $target tidak ditemukan');

  }

}

int binarySearch(List<int> arr, int target, int left, int right) {

  if (left > right) {

    return -1; // Target tidak ditemukan

  }

  int mid = left + (right - left) ~/ 2;

  if (arr[mid] == target) {

    return mid; // Target ditemukan

  } else if (arr[mid] > target) {

    return binarySearch(arr, target, left, mid - 1); // Cari di sebelah kiri

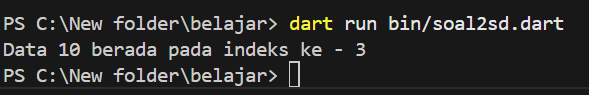
  } else {

    return binarySearch(arr, target, mid + 1, right); // Cari di sebelah kanan

  }

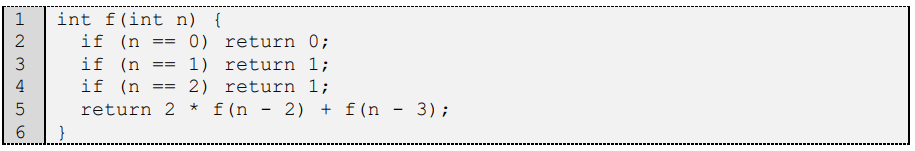
}

**Hasil pada terminalnya**

****

**Penjelasan**

Program ini menggunakan metode Binary Search untuk mencari nilai dalam daftar [2, 5, 8, 10, 14, 32, 35, 41, 67, 88, 90, 101, 109]. Fungsi binarySearch menerima array, target, serta indeks kiri dan kanan untuk pencarian rekursif. Pencarian dilakukan dengan memeriksa elemen di tengah (mid). Jika elemen tersebut adalah target, fungsi mengembalikan indeksnya. Jika lebih besar, pencarian berlanjut ke kiri; jika lebih kecil, ke kanan. Program ini mencetak indeks dari nilai yang dicari atau menyatakan bahwa nilai tersebut tidak ditemukan.

1. Jelaskan proses rekursif untuk program dibawah ini !

Berikut adalah proses program rekursif diatas

void main() {

  // Contoh penggunaan

  for (int n = 0; n <= 10; n++) {

    print('f($n) = ${f(n)}');

  }

}

int f(int n) {

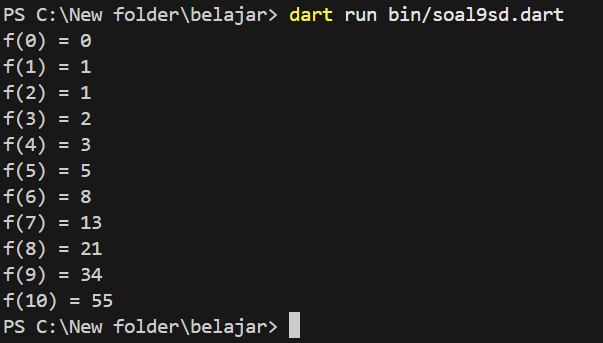
  if (n == 0) return 0;  // Basis: n = 0

  if (n == 1) return 1;  // Basis: n = 1

  if (n == 2) return 1;  // Basis: n = 2

  return 2 \* f(n - 2) + f(n - 3); // Rekursi untuk n > 2

}

**Hasil pada terminalnya**

**Penjelasan**

Untuk menggunakan fungsi f(n), pertama-tama, definisikan fungsi tersebut untuk menghitung nilai berdasarkan input. Selanjutnya, dalam fungsi main, lakukan loop dari 0 hingga 10 untuk memanggil f(n) dan menampilkan hasilnya. Program akan mencetak nilai f(n) untuk setiap nilai n yang diberikan, memungkinkan Anda menghitung nilai sesuai dengan aturan rekursif yang ditentukan.

KESIMPULAN

Kesimpulan dari program-program yang dibahas di atas adalah bahwa semua contoh tersebut menunjukkan cara menggunakan rekursi dalam pemrograman. Segitiga Pascal menghitung nilai dengan menjumlahkan dua elemen di atasnya dan dapat dengan mudah diubah untuk menambah baris. Pencarian dalam string menghitung nilai berdasarkan karakter dalam string biner, di mana setiap karakter berkontribusi pada hasil.

Pencarian di daftar dilakukan secara efisien dengan mencatat langkah-langkah yang diambil, sedangkan binary search mencari nilai dalam daftar terurut dengan membagi pencarian menjadi dua bagian. Fungsi umum juga menghitung nilai berdasarkan kondisi tertentu dan dapat mencetak hasil untuk berbagai nilai. Secara keseluruhan, rekursi membantu menyelesaikan masalah dengan cara yang sederhana dan efektif, serta meningkatkan pemahaman kita tentang algoritma dan pemrograman.

REFERENSI

* <https://www.kodeco.com/books/data-structures-algorithms-indart/v1.0/chapters/3-basic-data-structures-in-dart>
* https://www.geeksforgeeks.org/dart-recursion/
* https://www.javatpoint.com/dart-recursion