**LAPORAN PRAKTIKUM**

**STRUKTUR DATA**

**REKURSIF**

****

**DOSEN PENGAMPU ;**

**LUTFI HAKIM, S. PD, M. T**

**DISUSUN OLEH :**

**MUHAMMAD NURIL ANWARI**

**362458302106**

**POLITEKNIK NEGERI BANYUWANGI**

**TEKNOLOGI REKAYASA PERANGKAT LUNAK**

**TAHUN 2025**

**BAB I**

**PENDAHULUAN**

**TUJUAN PRAKTIKUM**

Tujuan praktikum rekursif umumnya meliputi beberapa hal berikut:

1. **Memahami Konsep Rekursi:** Memahami bagaimana fungsi rekursif bekerja, yaitu fungsi yang memanggil dirinya sendiri untuk menyelesaikan suatu masalah. Praktikum ini bertujuan agar mahasiswa atau peserta dapat mengenali pola rekursif dan bagaimana penerapannya pada berbagai masalah.
2. **Meningkatkan Kemampuan Pemecahan Masalah:** Rekursi adalah salah satu teknik pemrograman yang sangat berguna untuk memecahkan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil. Praktikum bertujuan untuk melatih peserta dalam mengidentifikasi masalah yang cocok diselesaikan menggunakan rekursi.
3. **Memahami Prinsip Divide and Conquer:** Banyak masalah dapat diselesaikan dengan pendekatan "divide and conquer" menggunakan rekursi, di mana masalah besar dibagi menjadi sub-masalah yang lebih kecil. Tujuannya adalah untuk memahami dan menerapkan teknik ini dalam pemrograman.
4. **Memperkenalkan Basis Kasus dan Langkah Rekursif:** Salah satu hal penting dalam rekursi adalah adanya "basis kasus" yang menghentikan rekursi, dan langkah rekursif yang memecah masalah menjadi bagian-bagian lebih kecil. Praktikum ini bertujuan untuk memahami bagaimana mendefinisikan kedua bagian ini.
5. **Mengembangkan Pemahaman tentang Efisiensi dan Batasan Rekursi:** Dengan melakukan praktikum rekursif, peserta dapat mempelajari kelebihan dan kekurangan teknik ini, termasuk masalah terkait dengan penggunaan memori, waktu eksekusi, dan batas kedalaman rekursi (stack overflow).
6. **Menggunakan Rekursi dalam Berbagai Algoritma:** Dalam praktikum ini, peserta dapat mempelajari penerapan rekursi dalam algoritma populer seperti pencarian biner, algoritma sorting (seperti quicksort dan mergesort), serta solusi masalah matematika atau pemrograman dinamis yang memanfaatkan rekursi.

**TEORI SINGKAT**

**Rekursif** adalah proses di mana suatu fungsi memanggil dirinya sendiri secara langsung atau tidak langsung. Dengan menggunakan algoritma rekursif, masalah tertentu dapat diselesaikan dengan cukup mudah. Fungsi rekursif juga diartikan sebagai fungsi yang memanggil dirinya sendiri secara berulang dalam proses pengolahan data atau pemanggilan fungsi. Pada beberapa kasus fungsi rekursif bisa lebih mudah dipahami dan sederhana daripada dengan solusi iteratif.

**BAB II**

**TUGAS PRAKTIKUM**

**TUGAS PENDAHULUAN**

1. Apa yang dimaksud dengan Rekursif?

Rekursif adalah konsep dalam pemrograman, di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil dari masalah awal hingga mencapai kondisi dasar. Kondisi dasar ini adalah titik di mana rekursi berhenti, dan solusi akhir dikembalikan. Dalam pemrograman, konsep ini sangat berguna untuk memecahkan masalah yang dapat dibagi menjadi submasalah yang lebih sederhana dengan pola yang sama.

Fungsi rekursi adalah fungsi yang secara eksplisit memanggil dirinya sendiri di dalam definisinya. Kunci utama dari hal ini adalah adanya basis kasus (*base case*) yang menentukan kapan fungsi harus berhenti memanggil dirinya sendiri. Tanpa basis kasus, sebuah fungsi rekursi akan terus memanggil dirinya tanpa henti dan menyebabkan *stack overflow*, yang bisa menghabiskan memori komputer.

Fungsi rekursi biasanya jauh lebih elegan dan mudah dibaca dibandingkan dengan solusi iteratif yang memerlukan pengulangan melalui *loop*. Namun, terdapat pula beberapa kelemahannya, seperti penggunaan memori yang lebih banyak dibandingkan dengan iterasi, karena setiap pemanggilan fungsi memerlukan tempat penyimpanan di memori.

1. Tuliskan fungsi untuk menghitung nilai factorial.

int faktorial(int n) {

  // Basis kasus: faktorial dari 0 adalah 1

  if (n == 0) return 1;

  // Langkah rekursif: n \* faktorial(n - 1)

  return n \* faktorial(n - 1);

}

void main() {

  int hasil = faktorial(5);

  print(hasil); // Output: 120

}

1. Tuliskan fungsi untuk menampilkan nilai fibonacci dari dere Fibonacci.

int fibonacci(int n) {

  // Basis kasus: fibonacci(0) = 0 dan fibonacci(1) = 1

  if (n == 0) return 0;

  if (n == 1) return 1;

  // Langkah rekursif: fibonacci(n) = fibonacci(n - 1) + fibonacci(n - 2)

  return fibonacci(n - 1) + fibonacci(n - 2);

}

void main() {

  int n = 6; // Menampilkan nilai Fibonacci ke-6

  print(fibonacci(n)); // Output: 8

}

1. Apa yang dimaksud dengan rekursif tail?

**Rekursif tail** (tail recursion) adalah jenis rekursi di mana pemanggilan fungsi rekursif terjadi pada bagian akhir dari fungsi tersebut, yaitu **sebelum** fungsi mengembalikan nilai. Dengan kata lain, hasil dari pemanggilan rekursif langsung dikembalikan tanpa ada operasi lebih lanjut setelahnya. Keuntungan utama dari rekursif tail adalah bahwa kompiler atau interpreter dapat mengoptimalkan penggunaan memori, karena tidak perlu menyimpan jejak pemanggilan fungsi sebelumnya dalam stack. Ini disebut **tail call optimization (TCO)**, yang memungkinkan rekursi menjadi lebih efisien dan mengurangi kemungkinan stack overflow.

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

int fibonacciTail(int n, int a, int b) {

  // Basis kasus: jika n == 0, kembalikan a (nilai Fibonacci ke-n)

  if (n == 0) return a;

  // Langkah rekursif tail: panggil fibonacciTail dengan parameter baru

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

}

int fibonacci(int n) {

  // Mulai dengan a = 0 dan b = 1 (nilai Fibonacci pertama dan kedua)

  return fibonacciTail(n, 0, 1);

}

void main() {

  print(fibonacci(6)); // Output: 8 (Fibonacci ke-6)

}

**PERCOBAAN**

**Percobaan 1** Fungsi rekursif untuk menghitung nilai factorial.

import 'dart:io';

int faktorial(int x) {

 if (x == 1) {

 return x;

 } else {

 return x \* faktorial(x - 1);

 }

}

void main() {

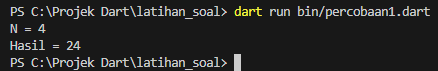
 stdout.write("N = ");

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

 print("Hasil = ${faktorial(n)}");

}

**Hasil**

****

**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**

****

**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");

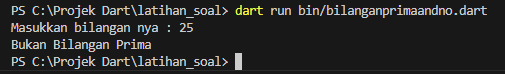
 } else {

 print("Bukan Bilangan Prima");

 }

}

**Hasil**

****

**Pecobaan 4** Fungsi rekursi untuk menampilkan kombinasi 2 karakter

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);

 }

 }

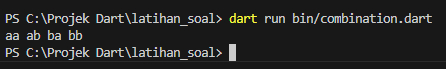
}

void main() {

 charCombination("", 2);

}

**Hasil**

****

**Percobaan 5** Fungsi rekursi untuk mengitung 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()!);

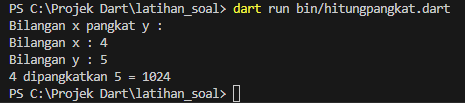
 stdout.write("Bilangan y : ");

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

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

}

**Hasil**

****

**Percobaan 6**  Fungsi tail rekursif untuk menampilkan i

import 'dart:io';

int faktorial(int x) {

if (x == 1) {

return x; // Basis kasus

} else {

return x \* faktorial(x - 1); // Rekursi

}

}

void main() {

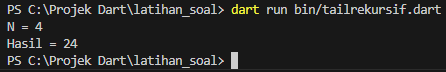
stdout.write("N = ");

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

print("Hasil = ${faktorial(n)}");

}

**Hasil**

****

**Percobaan 7** Fungsi rekursif tail untuk menghitung factorial

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**

**C:\Users\user\Pictures\Screenshots\Screenshot (189).png**

**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**

**C:\Users\user\Pictures\Screenshots\Screenshot (191).png**

**LATIHAN DAN PEMBAHASAN**

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

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

  // Basis kasus: jika left > right, berarti data tidak ditemukan

  if (left > right) {

    return -1;  // Menandakan data tidak ditemukan

  }

  // Hitung tengah

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

  // Jika elemen tengah adalah target, kembalikan indeksnya

  if (arr[mid] == target) {

    return mid;

  }

  // Jika target lebih kecil dari elemen tengah, cari di sebelah kiri

  if (arr[mid] > target) {

    return binarySearch(arr, left, mid - 1, target);

  }

  // Jika target lebih besar dari elemen tengah, cari di sebelah kanan

  return binarySearch(arr, mid + 1, right, target);

}

void main() {

  List<int> data = [2, 5, 15, 23, 37, 44, 52, 73, 88, 95, 110];

  int target = 10;

  // Panggil fungsi binarySearch

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

  // Tampilkan hasil

  if (result != -1) {

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

  } else {

    print('Data $target tidak ditemukan');

  }

}

Program ini bertujuan untuk mencari sebuah posisi angka yang terdaftar menggunakan Bynary Search Rekursif. Yang berguna untuk pencarian dalam database,apk dll

Binary search akan memulai dari tengah dengan cara

* 1. ambil angka dari tengah – cek apakah itu target
  2. jika target(tengah) lebih kecil maka cari di kiri
  3. jika target lebih besar cari di kanan ulang terus sampai ketemu target.

Hal ini sangat efisien dari pada hitung satu persatu dari indeks 0 / linear search. Jika data ada banyak contoh 1000 bisa menggunakan metode binary search mungkin membutuhkan 10 langkah , tapi kalo menggunakan satu persatu / linear search maka ada 1000 langkah

**Fungsi rekursif** (binarySearch(arr, left, right, target) ini untuk mencari posisi suatu angka yang terdaftar / urut menggunakan metode *Binary Search*

**Main function untuk proses input dan output**

Untuk menerima input dari user menggunakan stdin.readLineSync(). Input di ubah menjadi sebuah daftar angka/ (List<int) menggunakan split(',') dan int.tryParse(). Daftar angka di urutkan menggunakan sort() .

**Hasil**

**C:\Users\user\Pictures\Screenshots\Screenshot (193).png**

1. Buatlah program rekursif untuk menghitung segitiga Pascal !

import 'dart:io';

// Fungsi rekursif untuk menghitung nilai di segitiga Pascal

int pascal(int row, int col) {

  if (col == 0 || col == row) {

    return 1;

  }

  return pascal(row - 1, col - 1) + pascal(row - 1, col);

}

// Fungsi untuk mencetak Segitiga Pascal

void PascalTriangle(int n) {

  for (int i = 0; i < n; i++) {

    List<int> row = [];

    for (int j = 0; j <= i; j++) {

      row.add(pascal(i, j));

    }

    print(row.join(" "));

  }

}

void main() {

  stdout.write("Masukkan jumlah baris: ");

  int? n = int.tryParse(stdin.readLineSync()!);

  if (n != null && n > 0) {

    PascalTriangle(n);

  } else {

    print("Input harus berupa angka positif.");

  }

}

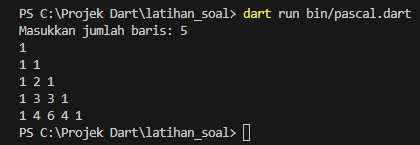
**Pada bagian Fungsi Rekursif pascal** (row, col) di *(syntax)*. Fungsi ini digunakan untuk menghitung angka dalam segitiga Pascal.

**Pada bagian** pascaltriangle di *(syntax)* menggunakan looping/perulangan untuk mencetak segitiga pascal

Di dalam *code* ini menggunakan for untuk loop pada loop 1: (for i = 0; i < n; i++) loop 2**:** (for (int j = 0; j <= i; j++) setelah itu menggunakan list row untuk menyimpan angka dalam satu baris, row akan di cetak dengan menggunakan (row.join(" "))

**Main function** void main() untuk proses input dan output , untuk membaca input dari user dalam bentuk string menggunakan stdin.readLineSync(), dan di ubah **string** ke angka **(int)** menggunakan int.tryParse,Dan ada if untuk meriksa apakah input valid atau tidak, Jika n adalah angka /**bukan null** dan lebih dari **0**, maka lanjut (akan di tampilkan dengan segitiga pascal).

**Hasil**

****

**KESIMPULAN**

Praktikum rekursif memberikan kita pemahaman yang lebih dalam tentang cara kerja fungsi rekursif dalam pemrograman, dan perlunya perhatian dalam penggunaannya untuk menghindari masalah performa dan memori.

**DAFTAR REFERENSI**

1. **Modul Struktur Data (BAB 3 REKURSIF)**

<https://lms.poliwangi.ac.id/pluginfile.php/21387/mod_resource/content/2/Modul%20Struktur%20Data%2001%20-%20Rekursif.pdf>

1. **Chat GPT**

<https://chatgpt.com/>

1. **GeeksforGeeks. (n.d.). *Recursion*.**

<https://www.geeksforgeeks.org/recursion/>