# LAPORAN PRAKTIKUM
# REKURSIF DALAM BAHASA C


Oleh:


| | |
|---|---|
| Nama | : Muhammad Ajra Zemima Muda |
| NIM | : 2023903430022 |
| Kelas | : TRKJ 1.C |
| Jurusan | : Teknologi Informasi dan Komputer |
| Program Studi | : Teknologi Rekayasa Komputer Jaringan |
| Dosen Pembimbing | : Indrawati, SST. MT |



**POLITEKNIK NEGERI LHOKSEUMAWE**
**TAHUN AJARAN 2023/2024**

**LEMBAR PENGESAHAN**

| | |
|---|---|
| No. Praktikum | : 09 /TIK/TRKJ-1C/ Data Structure And Algorithms Practice |
| Judul | : Laporan Pratikum Rekursif |
| Nama | : Muhammad Ajra Zemima Muda |
| NIM | : 2023903430022 |
| Kelas | : TRKJ-1C |
| Jurusan | : Teknologi Informasi Dan Komputer |
| Prodi | : Teknologi Rekayasa Komputer Jaringan |
| Tanggal Praktikum | : 20 November 2023 |
| Tanggal Penyerahan | : 27 November 2023 |

Buketrata, 27 November 2023

Dosen Pembimbing,

Indrawati, SST.MT

Nip. 19740815 200112 2 001

## 1.2. PENDAHULUAN

### A. Tujuan

1. Mampu menguasai konsep penulisan program Rekursif dalam bahasa C.
2. Mampu mengimplementasi program Rekursif dalam bahasa C.
3. Mampu menyelesaikan masalah dengan memanfaatkan Rekursif dalam bahasa C.

### B. Alat dan Bahan:

- Komputer atau Laptop Dengan Kompiler C (GCC)
- Editor Teks (Misalnya Notepad ++, DEV – C++)
- Buku Panduan atau Materi Pratikum
- Kertas dan Pensil
- Ruang Praktikum

### C. Dasar Teori

Dasar teori tentang rekursi dalam bahasa C mencakup konsep bahwa rekursi merupakan suatu proses di mana sebuah fungsi memanggil dirinya sendiri secara berulang. Dalam konteks rekursi, setiap fungsi selalu diikuti oleh kondisi yang menentukan kapan proses rekursi tersebut harus berhenti. Dalam setiap panggilannya, fungsi rekursif selalu memanggil dirinya sendiri sambil mengurangi atau memecahkan data masukan.

Rekursi dapat diinterpretasikan sebagai suatu teknik perulangan, walaupun dengan konsep yang berbeda dalam konteks pemrograman. Kelebihan utama dari rekursi adalah kemampuannya menciptakan program yang lebih elegan. Namun, perlu diperhatikan bahwa jika kinerja program menjadi faktor kritis, lebih disarankan untuk menggunakan loop sebagai pengganti rekursi karena rekursi cenderung memiliki kinerja yang lebih lambat.

Penggunaan rekursi juga merupakan konsep penting dalam bahasa C, dan sering digunakan dalam implementasi struktur data dan algoritma.

Contohnya adalah pada penyelesaian masalah traversal tree, di mana pendekatan rekursif dapat memberikan solusi yang terstruktur dan mudah dipahami.

## 1.3. URAIAN PRAKTIKUM

### A. Menghitung Bilangan Fibonacci

1. **Program:**

```c
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
int main() {
    int n;
    printf("Masukkan jumlah bilangan dalam deret Fibonacci: ");
    scanf("%d", &n);
    if (n < 0) {
        printf("Masukkan jumlah bilangan positif.\n");
    } else {
        printf("Deret Fibonacci:\n");
        for (int i = 0; i < n; i++) {
            printf("%d ", fibonacci(i));
        }
        printf("\n");
    }
    return 0;
}

```

*Gambar 0 1.Program*

2. **Output**:

```
Masukkan jumlah bilangan dalam deret Fibonacci: 20
Deret Fibonacci:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

--------------------------------
Process exited after 16.68 seconds with return value 0
Press any key to continue . . .
```

*Gambar 0 2.Hasil Output*

3. **Penjelasan:**

a. Fungsi fibonacci:

- Fungsi ini mengimplementasikan rekursi untuk menghitung nilai deret Fibonacci ke-n.
- Jika n kurang dari atau sama dengan 1, maka fungsi mengembalikan nilai n (basis rekursi).
- Jika n lebih besar dari 1, maka fungsi memanggil dirinya sendiri untuk

dua nilai sebelumnya dalam deret dan mengembalikan penjumlahan keduanya.

b. Fungsi main:
   - Mendeklarasikan variabel n untuk menyimpan jumlah bilangan dalam deret Fibonacci.
   - Mengambil input dari pengguna untuk nilai n menggunakan scanf.
   - Memeriksa apakah n kurang dari 0. Jika ya, program mencetak pesan kesalahan dan berhenti.
   - Jika n tidak negatif, program mencetak deret Fibonacci dengan memanggil fungsi fibonacci(i) untuk setiap nilai i dari 0 hingga n-1.

c. Input dan Output:
   - Program meminta pengguna memasukkan jumlah bilangan dalam deret Fibonacci.
   - Menggunakan printf dan scanf untuk mencetak pesan dan menerima input.
   - Jika input n tidak negatif, program mencetak deret Fibonacci dengan format tertentu menggunakan printf.

d. Pentingnya Batasan pada Input:
   - Program memberikan pesan kesalahan jika pengguna memasukkan jumlah bilangan negatif. Ini penting untuk memastikan bahwa deret Fibonacci hanya dihitung untuk nilai yang masuk akal.

e. Rekursi pada Deret Fibonacci:
   - Meskipun pendekatan rekursif sederhana untuk menghitung deret Fibonacci, dapat terjadi masalah kinerja untuk nilai n yang besar karena banyak perhitungan yang diulang.

f. Iterasi Melalui Deret Fibonacci:
   - Program menggunakan perulangan for untuk mencetak deret Fibonacci dari 0 hingga n-1.

g. Penggunaan Fungsi Rekursif:
   - Program menggunakan fungsi rekursif untuk menghitung nilai deret Fibonacci. Ini adalah contoh penggunaan rekursi dalam pemrograman.

## B. Menghitung Faktorial

1. **Program:**

```c
#include <stdio.h>
int faktorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * faktorial(n - 1);
    }
}
int main() {
    int angka;
    printf("Masukkan angka untuk menghitung faktorial: ");
    scanf("%d", &angka);
    if (angka < 0) {
        printf("Masukkan angka non-negatif.\n");
    } else {
        printf("Faktorial dari %d = %d\n", angka, faktorial(angka));
    }
    return 0;
}
```

*Gambar 0 3.Program*

2. **Output:**

```
Masukkan angka untuk menghitung faktorial: 4
Faktorial dari 4 = 24

--------------------------------
Process exited after 3.344 seconds with return value 0
Press any key to continue . . .
```

*Gambar 0 4.Hasil Output*

3. **Penjelasan**:

   a. Fungsi faktorial

   - Program memiliki satu fungsi, yaitu faktorial(int n), yang bertanggung jawab untuk menghitung faktorial dari suatu bilangan bulat n.
   - Fungsi ini menggunakan pendekatan rekursif. Jika n sama dengan 0 atau 1, maka nilai faktorialnya adalah 1. Jika tidak,

fungsi akan memanggil dirinya sendiri dengan parameter n - 1 dan mengalikan hasilnya dengan n.

b. Fungsi main
    - Pada fungsi main, program meminta pengguna memasukkan suatu angka untuk dihitung faktorialnya.
    - Angka yang dimasukkan oleh pengguna disimpan dalam variabel angka menggunakan fungsi scanf.
    - Program melakukan pemeriksaan, jika angka yang dimasukkan kurang dari 0, maka program memberikan pesan bahwa pengguna harus memasukkan angka non-negatif.
    - Jika angka yang dimasukkan tidak kurang dari 0, program mencetak hasil faktorial dari angka tersebut dengan memanggil fungsi faktorial dan menampilkan hasilnya menggunakan printf.

## C. Mencari Nilai Pangkat

1. **Program:**

```c
#include <stdio.h>
double pangkat(double x, int n) {
    if (n == 0) {
        return 1;
    } else if (n > 0) {
        return x * pangkat(x, n - 1);
    } else {
        return 1 / (x * pangkat(x, -n - 1));
    }
}
int main() {
    double angka;
    int pangkat;
    printf("Masukkan angka: ");
    scanf("%lf", &angka);
    printf("Masukkan pangkat (bilangan bulat): ");
    scanf("%d", &pangkat);
    printf("%.2lf^%d = %.4lf\n", angka, pangkat, pangkat(angka, pangkat));
    return 0;
}
```

*Gambar 0 5.Program*

2. **Output:**

```
Masukkan angka: 3
Masukkan pangkat (bilangan bulat): 2
3.00^2 = 9.0000

--------------------------------
Process exited after 2.535 seconds with return value 0
Press any key to continue . . .
```

*Gambar 0 6.Hasil Output*

3. **Analisa:**

a. Fungsi Rekursif pangkat:

- Fungsi pangkat menghitung nilai x pangkat n. Fungsi ini rekursif dan memiliki tiga kondisi:
- Jika n sama dengan 0, maka fungsi mengembalikan 1 (basis rekursi).
- Jika n lebih besar dari 0, maka fungsi memanggil dirinya sendiri dengan n - 1 dan mengalikan hasilnya dengan x.
- Jika n kurang dari 0, maka fungsi memanggil dirinya sendiri

dengan -n - 1 dan mengembalikan hasil bagi 1 dengan hasil perkalian x.

b. Fungsi main:
   - Mendeklarasikan variabel angka dan pangkat.
   - Mengambil input dari pengguna untuk nilai angka dan pangkat menggunakan scanf.
   - Memanggil fungsi pangkat untuk menghitung hasil pangkat.
   - Mencetak hasil pangkat dalam bentuk yang sudah ditentukan.

c. Input dan Output:
   - Program meminta pengguna memasukkan nilai angka dan pangkat.
   - Menggunakan fungsi printf dan scanf untuk mencetak pesan dan menerima input.
   - Hasil pangkat dari angka^pangkat dicetak dengan format tertentu menggunakan printf.

d. Penanganan Error:
   - Program tidak memiliki penanganan error atau validasi input. Ini dapat menyebabkan masalah jika pengguna memasukkan input yang tidak valid, misalnya, bukan bilangan bulat untuk pangkat.

e. Kesalahan pada pemanggilan fungsi di main:
   - Terdapat kesalahan di baris printf("%.2lf^%d = %.4lf\n", angka, pangkat, pangkat(angka, pangkat));. Seharusnya, pemanggilan fungsi pangkat dilakukan seperti ini: pangkat(angka, pangkat).

Program ini sederhana namun dapat digunakan untuk menghitung pangkat bilangan real dengan eksponen bilangan bulat. Namun, perlu diperhatikan bahwa pendekatan rekursif mungkin tidak efisien untuk nilai n yang sangat besar karena menyebabkan pemanggilan fungsi bertumpuk (stack overflow).

## 1.4. PENUTUP

### A. Kesimpulan

Berdasarkan analisis program yang telah kita lihat, dapat disimpulkan beberapa hal terkait praktikum atau materi yang mungkin sedang dipelajari:

1. Rekursi:
    - Program-program tersebut menggunakan pendekatan rekursi untuk menyelesaikan permasalahan, seperti perhitungan faktorial, deret Fibonacci, dan perpangkatan. Rekursi dapat menjadi cara yang elegan untuk menangani perhitungan berulang dengan memecahnya menjadi kasus yang lebih kecil.
2. Pemeriksaan Input:
    - Kedua program memiliki langkah-langkah untuk memeriksa input pengguna agar sesuai dengan kebutuhan. Hal ini merupakan praktik baik dalam pengembangan perangkat lunak untuk memastikan keandalan dan keselamatan program.
3. Ketidakefisienan Rekursi untuk Input Besar:
    - Meskipun rekursi dapat memberikan solusi yang mudah dipahami dan diimplementasikan, terdapat potensi ketidakefisienan untuk input yang besar. Rekursi dapat mengakibatkan duplikasi perhitungan dan penumpukan tumpukan pemanggilan fungsi.
4. Manipulasi Tipe Data:
    - Program-program tersebut menggabungkan penggunaan tipe data yang berbeda, seperti integer (int) dan bilangan real (double). Hal ini menunjukkan kemampuan untuk memanipulasi tipe data yang berbeda sesuai dengan kebutuhan perhitungan.
5. Format Output:
    - Format output dipertimbangkan dengan baik, seperti pada penggunaan printf untuk mencetak hasil dengan presisi desimal tertentu. Hal ini memperlihatkan keahlian dalam menampilkan output yang informatif dan mudah dibaca.

6. Penerapan Konsep Matematis:
    - Program-program tersebut mencerminkan penerapan konsep matematis, seperti faktorial, deret Fibonacci, dan perpangkatan. Ini mengilustrasikan cara pemrograman dapat digunakan untuk merepresentasikan dan menyelesaikan permasalahan matematis.

Dengan memahami dan menganalisis program-program tersebut, diharapkan peserta praktikum dapat mengembangkan pemahaman yang lebih baik tentang rekursi, pemeriksaan input, manipulasi tipe data, dan penerapan konsep matematis dalam konteks pemrograman.

# DAFTAR PUSTAKA

https://id.scribd.com/document/407152981/Contoh-Program-Sederhana-Rekursif-Dengan-Bahasa-C

https://kelasprogrammer.com/contoh-fungsi-rekursif/

https://www.duniailkom.com/latihan-kode-program-bahasa-c-fungsi-rekursif-menghitung-faktorial/