

IK131 Algoritma dan Pemrograman
Prosedur & Fungsi



Di Susun Oleh :
Tim Pendamping Pak Jajang

2425 – Jajang Kusnendar, M.T.

PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

1. Identitas

Nama Mata Kuliah – Kode	:	IK131 – Algoritma dan Pemrograman
Materi	:	Prosedur & Fungsi
Sub Materi	:	<ul style="list-style-type: none">- Konsep Prosedur- Contoh Kasus Prosedur- Cara Penulisan Prosedur- Variabel Scope- Pass By Value & Pass By Reference- Konsep Fungsi- Cara Penggunaan Fungsi
Beban (Waktu Pelaksanaan)	:	150 menit
Semester	:	1
Pengampu MK	:	Jajang Kusnendar, M.T

2. Capaian Praktikum

Capaian Praktikum
<ul style="list-style-type: none">- Siswa dapat memahami bagaimana perbedaan prosedur & fungsi.- Siswa dapat menjelaskan definisi prosedur dan juga fungsi.- Siswa dapat menuliskan kode program dengan benar.- Siswa dapat mengimplementasikannya dalam contoh kasus yang diberikan.

3. Perangkat yang Dibutuhkan

Perangkat lunak
<ul style="list-style-type: none">- Compiler bahasa C- Pada windows : DevC++, VisualStudio

Ringkasan Materi
1. Prosedur
A. Konsep & Definisi

Di dalam dunia programming ada sebuah prinsip yaitu **DRY (Don't Repeat Yourself)**, gunanya agar kita dapat menghindari kode program yang berulang secara terus menerus, dengan melakukan hal tersebut kita dapat mencegah beberapa kasus seperti:

- 1) Kesulitan dalam pemeliharaan / maintenance,
- 2) Kode menjadi tidak konsisten,
- 3) Peningkatan kompleksitas,
- 4) Pemborosan waktu.

Contoh Kasus:

Input 1 4 3 2 Output 1 **** **** **** **** *** *** *** ** **	Input 2 1 2 3 Output 2 * ** ** *** *** ***	Input 3 3 2 3 Output 3 *** *** *** ** ** *** *** ***
---	--	--

KODE PROGRAMNYA

```
#include <stdio.h>

int main () {
    int a, b, c;

    scanf ("%d %d %d", &a, &b, &c);

    for (int i = 0; i < a; i++){
        for (int j = 0; j < a; j++){
            printf("*");
        }

        printf("\n");
    }
}
```

```
for (int i = 0; i < b; i++){
    for (int j = 0; j < b; j++){
        printf("*");
    }

    printf("\n");
}

for (int i= 0; i<c; i++) {
    for (int j = 0; j < c; j++) {
        printf("*");
    }

    printf("\n");
}

return 0;
}
```

APAKAH KEKURANGAN KODE PROGRAM TERSEBUT??

```
#include<stdio.h>

void pola(int n){
    for(int i = 0 ; i < n; i++){
        for(int j = 0; j < n; j++){
            printf("*");
        }

        printf("\n");
    }
}

int main(){
    int a, b, c;

    scanf("%d %d %d", &a, &b, &c);

    pola(a);
    pola(b);
    pola(c);

    return;
}
```

Adapun definisi **prosedur** itu sendiri adalah suatu bagian program yang melakukan tugas tertentu. **Prosedur** didefinisikan **di luar** fungsi *main*, lalu prosedur dapat digunakan **berulang kali** dengan cara memanggil prosedur tersebut.

Struktur Prosedur

```
void nama_prosedur (variable1 nama_parameter, variable2 nama_parameter2,...){  
    /* Isi Prosedur*/  
}
```

Deskripsi

Void merupakan salah satu tanda bahwa kode selanjutnya yang akan ditulis merupakan sebuah prosedur.

Nama_prosedur merupakan nama yang akan digunakan saat proses deklarasi dan pemanggilan prosedur. Aturan penulisan nama prosedur sama dengan seperti aturan penulisan nama variable.

Parameter merupakan variable – variable yang nilainya diambil dari luar prosedur dan akan digunakan di dalam prosedur. Sebuah prosedur dapat memiliki minimal 0 buah parameter dan maksimal ∞ buah parameter (selama memori computer mencukupi).

Pemanggilan

```
nama_prosedur(nilai_parameter1, nilai_parameter2);
```

B. Contoh Kode

- Prosedur untuk membandingkan nilai

```
void bandingkan(char a, char b)  
{  
    if (a > b){  
        printf("Parameter kiri lebih besar\n");  
    }else if (a < b){  
        printf("Parameter kanan lebih besar\n");  
    }else{  
        printf("Parameter kiri dan kanan sama\n");  
    }  
}
```

- Prosedur untuk print pembatas

```
void pembatas(){  
    printf("-----\n");  
}
```

- Prosedur dengan parameter struct

```
typedef struct{
    char nama[101];
    int nim;
    float ipk;
} mahasiswa;

void print_data_mahasiswa(mahasiswa mhs){
    printf("Data Mahasiswa\n");
    printf("Nama : %s\n", mhs.nama);
    printf("NIM : %d\n", mhs.nim);
    printf("IPK : %.2f\n", mhs.ipk);
}
```

- Prosedur dengan parameter array

```
void reverse(char str[]){
    int n = strlen(str);

    for (int i = n - 1; i >= 0; i--){
        printf("%c", str[i]);
    }
    printf("\n");
}
```

Kapan kita harus menggunakan prosedur?

- Saat sekumpulan kode dengan pola yang sama muncul lebih dari satu kali.
- Saat kode yang ditulis sangat panjang, lebih baik dikelompokkan menjadi beberapa bagian.
- Saat ada potongan kode yang bertujuan untuk menjalankan tugas tertentu.

C. Cara Penulisan Prosedur

Proses pembuatan sebuah prosedur dapat dibagi menjadi dua bagian, yaitu deklarasi dan definisi.

- **Deklarasi** adalah saat kita menuliskan void, nama prosedur, dan parameter tanpa menuliskan isi prosedur tersebut.
- **Definisi** adalah saat kita menuliskan void, nama prosedur, parameter, dan isi prosedur.

Penulisan definisi prosedur tanpa deklarasi

```
#include<stdio.h>

//definisi prosedur
void prosedur1(){
    //isi prosedur 1
}

void prosedur2(){
    //isi prosedur 2
}

void prosedur3(){
    //isi prosedur 3
}

int main(){
    return 0;
}
```

Penulisan prosedur dengan deklarasi dan definisi

```
#include<stdio.h>

//definisi prosedur
void prosedur1(){
    //isi prosedur 1
}

void prosedur2(){
    //isi prosedur 2
}

void prosedur3(){
    //isi prosedur 3
}

int main(){
    //deklarasi prosedur
    void prosedur1();
    void prosedur2();
    void prosedur3();

    return 0;
}
```

D. Variable Scope

Ruang Lingkup Variabel (Variable Scope) merujuk pada batasan yang menentukan lokasi-lokasi di mana sebuah variabel dapat digunakan. Sebuah variabel hanya dapat diakses dalam area atau blok kode tempat variabel tersebut dideklarasikan.

```
#include <stdio.h>
int var1;

void prosedur(){
    int var2;
}

int main(){
    int var3;

    if(var3 > 0){
        int var4;
    }
}
```

```
return 0;
}
```

E. Pass By Value & Pass By Reference

Pass By Value berarti hanya nilai dari variabel yang diberikan kepada prosedur. Oleh karena itu, jika nilai parameter diubah dalam prosedur, nilai variabel di luar prosedur tidak akan terpengaruh.

Sementara itu, **Pass By Reference** berarti baik nilai maupun alamat memori variabel (pointer) diberikan kepada prosedur. Dengan demikian, jika nilai parameter diubah dalam prosedur, nilai variabel di luar prosedur juga akan ikut berubah.

Catatan: Parameter yang bertipe array selalu bersifat pass by reference!!!

```
#include <stdio.h>

// Fungsi dengan Pass By Value
void passByValue(int a) {
    a = a + 1; // Mengubah salinan nilai lokal
    printf("Di dalam passByValue: %d\n", a); // Hanya mengubah salinan,
    bukan nilai asli
}

// Fungsi dengan Pass By Reference
void passByReference(int *a) {
    *a = *a + 1; // Mengubah nilai yang ditunjuk oleh pointer
    printf("Di dalam passByReference: %d\n", *a); // Nilai asli berubah
}

int main() {
    int x = 5;
    printf("Sebelum passByValue: %d\n", x);
    passByValue(x); // Pass By Value (nilai tidak berubah)
    printf("Setelah passByValue: %d\n", x); // Nilai x tetap 5

    printf("Sebelum passByReference: %d\n", x);
    passByReference(&x); // Pass By Reference (nilai berubah)
    printf("Setelah passByReference: %d\n", x); // Nilai x berubah menjadi 6

    return 0;
}
```

Output


```
Sebelum passByValue: 5
Di dalam passByValue: 6
Setelah passByValue: 5
Sebelum passByReference: 5
Di dalam passByReference: 6
Setelah passByReference: 6
```

Pass By Reference bertipe struct

```
#include <stdio.h>

typedef struct {
    char nama[101];
    int nim;
    float ipk;
} mahasiswa;

void scan_data_mahasiswa(mahasiswa *mhs){

    scanf("%s %d %f", &(*mhs).nama, &(*mhs).nim, &(*mhs).ipk);
}

int main(){
    mahasiswa m1;

    scan_data_mahasiswa(&m1);
    printf("%s %d %.2f", m1.nama, m1.nim, m1.ipk);

    return 0;
}
```

Pass By Reference bertipe array

```
#include <stdio.h>

void passByReferenceArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] = arr[i] * 2; // Mengubah nilai array di dalam fungsi
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Sebelum passByReferenceArray: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
}
```

```
printf("\n");

passByReferenceArray(arr, size); // Pass by reference untuk array

printf("Setelah passByReferenceArray: ");
for (int i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}
printf("\n");

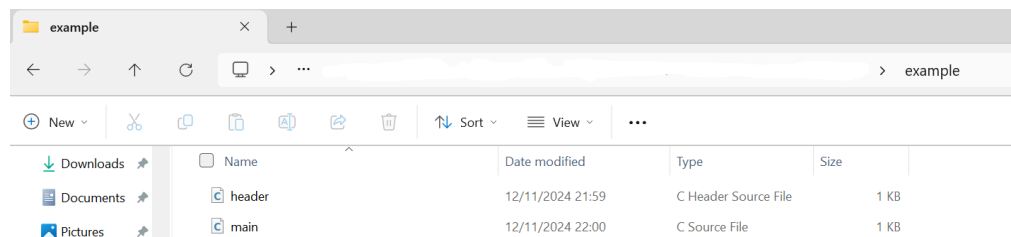
return 0;
}
```

Output

```
Sebelum passByReferenceArray: 1 2 3 4 5
Setelah passByReferenceArray: 2 4 6 8 10
```

F. Pemanggilan File

Step 1: Satukan file header dan main dalam satu folder yang sama.



Step 2: Tuliskan kode program yang akan dimasukkan ke dalam header.

```
#ifndef HEADER_H
#define HEADER_H

void tukar(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int nilaiGlobal = 10;

#endif
```

Step 3: Panggil file header bersamaan dengan library input dan output.

```
#include <stdio.h>
#include "header.h"

int main() {
    int num1 = 5, num2 = 10;

    printf("Sebelum penukaran:\n");
    printf("num1 = %d, num2 = %d\n", num1, num2);

    tukar(&num1, &num2);

    printf("Setelah penukaran:\n");
    printf("num1 = %d, num2 = %d\n", num1, num2);

    printf("Nilai dari variabel global: %d\n", nilaiGlobal);

    return 0;
}
```

Last step: Jalankan!

```
Sebelum penukaran:
num1 = 5, num2 = 10
Setelah penukaran:
num1 = 10, num2 = 5
Nilai dari variabel global: 10
```

2. Fungsi

A. Definisi

Fungsi dalam Algoritma & Pemrograman hampir sama seperti prosedur dengan fitur tambahan, yaitu dapat **mengembalikan nilai** ke pemanggilnya.

B. Cara Penggunaan Fungsi

Salah satu kata kunci yang membedakan prosedur dengan fungsi adalah tipe data yang didefinisikan untuk dikembalikan seperti int, float, double, dll. Bukan void!!

```
tipeDataKembalian namaFungsi(tipeData namaParameter1,
tipeData namaParameter2, ...);
```

Contoh deklarasi:

```
int tambah(int x, int y);  
float rata_rata(int x, int y);
```

Saat mendefinisikan fungsi terdapat statement yang harus dituliskan, yaitu statement **return**. Statement tersebut berfungsi untuk mengembalikan nilai kepada pemanggilnya.

Contoh definisi:

Cara 1

```
int tambah(int x, int y){  
    return x + y;  
}
```

```
float rata_rata(int x, int y){  
    return x / y;  
}
```

Cara 2

```
int tambah(int x, int y){  
    int hasil;  
  
    hasil = x + y;  
    return hasil;  
}
```

```
float rata_rata(int x, int y){  
    int hasil;  
  
    hasil = x / y;  
    return hasil;  
}
```

C. Istilah – Istilah dalam Fungsi

Pada saat mendeklarasikan fungsi atau prosedur, sintaks yang terdapat di dalam tanda kurung merupakan parameter. Sedangkan saat pemanggilan prosedur atau fungsi maka nilai atau variable yang ada di dalam kurung adalah argument.

Kata kunci **perbedaan argument dan parameter** adalah tempat nya berada.

```
float rata_rata(float x, float y){
    return x/y;
}

int main(){
    float hasil = rata_rata(4.0,2.0);

    return 0;
}
```

- **Parameter** adalah variabel lokal yang dideklarasikan dan hanya berlaku pada scope fungsi atau prosedur yang berkaitan, nama lain dari Parameter adalah Formal Argument.
- **Argument** adalah nilai atau variabel yang akan diberikan atau dicopy ke parameter pada fungsi atau prosedur terkait.

D. Contoh Kode Program

Fungsi untuk Mencari Bilangan Terbesar dalam Array

```
#include <stdio.h>

int cariMaksimum(int arr[], int n) {
    int maks = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > maks) {
            maks = arr[i];
        }
    }
    return maks;
}

int main() {
    int n;

    printf("Masukkan jumlah elemen array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Masukkan elemen array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int maksimum = cariMaksimum(arr, n);
    printf("Bilangan terbesar dalam array adalah: %d\n", maksimum);

    return 0;
}
```

Output

```
Masukkan jumlah elemen array: 4
Masukkan elemen array:
2 100 0 10
Bilangan terbesar dalam array adalah: 100
```

Fungsi Jumlah Sekaligus Tambah (Menggunakan Pass By Reference)

```
#include <stdio.h>

int tukarDanJumlah(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
    return *a + *b;
}

int main() {
    int num1, num2, hasil;

    printf("Masukkan bilangan pertama: ");
    scanf("%d", &num1);

    printf("Masukkan bilangan kedua: ");
    scanf("%d", &num2);

    printf("\nSebelum penukaran:\n");
    printf("Bilangan pertama = %d\n", num1);
    printf("Bilangan kedua = %d\n", num2);

    hasil = tukarDanJumlah(&num1, &num2);

    printf("\nSetelah penukaran:\n");
    printf("Bilangan pertama = %d\n", num1);
    printf("Bilangan kedua = %d\n", num2);
    printf("Jumlah dari kedua bilangan setelah penukaran adalah: %d\n",
hasil);

    return 0;
}
```

Output

```
Masukkan bilangan pertama: 27
Masukkan bilangan kedua: 31

Sebelum penukaran:
Bilangan pertama = 27
Bilangan kedua = 31

Setelah penukaran:
Bilangan pertama = 31
Bilangan kedua = 27
Jumlah dari kedua bilangan setelah penukaran adalah: 58
```

Lembar Kerja Mahasiswa

1. Lengkapilah program yang belum selesai ini hingga dapat menampilkan output

```
#include <stdio.h>

int hasilPangkatDua(int num) {
    // isi kode
}

void jenisBilangan(int num) {
    printf("Jenis bilangan: ");
    if (num % 2 == 0) {
        // isi kode
    } else {
        // isi kode
    }
}

int main()
{
    int num, pangkat;
    scanf("%d", &num);

    pangkat = hasilPangkatDua(/*isi parameter*/);
    printf("Hasil pangkat: %d\n");
    jenisBilangan(num);

    return 0;
}
```

Output:

```
2
Hasil pangkat: 4
Jenis bilangan: Bilangan Genap
```

2. Carilah dan perbaiki kesalahan dari program di bawah ini.


```
#include <stdio.h>

void isiKarakter(int n, char kotak[n][n], char karakter) {
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            kotak[j][i] = karakter;
        }
    }
}

void tampilkanKarakter(int n, char kotak[n][n]) {
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            printf("%c", kotak[i][j]);
        }
        printf("\n");
    }
}

int main()
{
    int n;
    char karakter;
    char kotak[n][n];
    scanf("%d", &n);
    scanf(" %c", &karakter);

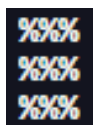
    isiKarakter(kotak, n, karakter);
    tampilkanKarakter(n, kotak);

    return 0;
}
```

Input:



Output:



3. Loni suka bunga.

1. Pada hari pertama, ia membeli sejumlah P ikatan bunga, dengan setiap ikatan berisi P bunga.
2. Pada hari kedua, ia membeli Q kotak bunga, dengan setiap kotak berisi Q bunga.
3. Pada hari ketiga, ia membeli R bunga.

Loni ingin membagi rata semua bunga yang dibeli ke 4 temannya, sehingga masing-masing temannya mendapatkan banyaknya bunga yang sama. Jika bunga yang dimilikinya tidak dapat dibagi rata ke semua temannya, maka tentukan bahwa pembagian tidak memungkinkan.

Bantu Loni untuk menghitung banyaknya bunga yang diterima oleh setiap temannya atau tentukan jika bunga-bunganya tidak dapat dibagi rata. (Gunakan fungsi atau prosedur untuk setiap langkah perhitungan).

Input 1:

```
Masukkan nilai P (ikatan bunga): 12
Masukkan nilai Q (kotak bunga): 73
Masukkan nilai R (bunga per hari ketiga): 1
```

Output 1:

```
Pembagian tidak memungkinkan
```

Penjelasan: Jumlah total bunga yang dibeli adalah $(P \times P) + (Q \times Q) + R$. Jika hasilnya tidak bisa dibagi rata ke 4 teman, cetak "Pembagian tidak memungkinkan".

Input 2:

```
Masukkan nilai P (ikatan bunga): 4
Masukkan nilai Q (kotak bunga): 4
Masukkan nilai R (bunga per hari ketiga): 0
```

Output 2:

```
Jumlah bunga per teman: 8
```

4. Lino menyukai sebuah mercon, saking sukanya pada mercon setiap ada bilangan A yang dia temukan, selalu dipangkatkan dengan bilangan B sehingga menghasilkan sebuah bilangan yang dinamakan **bilangan mercon**. Lino ingin mengetahui apakah total bilangan mercon yang dia hasilkan dari beberapa angka dapat dibagi rata oleh teman-temannya atau tidak.

Pada setiap langkah, Lino melakukan hal berikut:

- Setiap bilangan $A[i]$ dipangkatkan dengan $B[i]$, menghasilkan sebuah bilangan yang disebut **bilangan mercon**.
- Lino akan menghitung jumlah dari semua bilangan mercon yang dihasilkan dan ingin memastikan apakah jumlahnya **genap** atau **ganjil**.

Jika jumlah total bilangan mercon yang dihasilkan adalah **genap**, maka pembagian bisa dilakukan secara merata ke teman-temannya. Jika jumlahnya **ganjil**, maka pembagian tidak bisa dilakukan dengan rata.

Bantu Lino untuk menghitung apakah jumlah total bilangan mercon yang dihasilkan oleh Lino dapat dibagi rata (genap) atau tidak (ganjil). Lino akan membagikan jumlah ini kepada temannya. (Gunakan fungsi atau prosedur untuk setiap langkah perhitungan)

Input 1:

```
5
2 4 6 8 2
1 2 3 4 5
```

Output 1:

```
0
```

Penjelasan: Pada contoh masukan pertama, $2^1 + 4^2 + 6^3 + 8^4 + 2^5 = 2 + 16 + 216 + 4096 + 32 = 4362$ Karena 4362 bernilai genap, maka keluarkan “0” sebagai jawaban.

Input 2:

```
1
11
11
```

Output 2:

```
1
```

5. Murid-murid kelas 6 SD Pilkom terbagi atas kelas 6A dan kelas 6B, yang masing-masing terdiri atas N murid. Murid ke- i di kelas 6A memiliki pantun sepanjang $A[i]$ detik, dan murid ke- i di kelas 6B memiliki pantun sepanjang $B[i]$ detik.

Pada acara perpisahan, setiap murid di kelas 6A akan berbalas pantun dengan setiap murid di kelas 6B. Untuk sepasang murid ke- i di kelas 6A dan murid ke- j di kelas 6B, total waktu yang dibutuhkan mereka untuk berbalas pantun adalah $A[i] + B[j]$.

Panggung perpisahan hanya dapat menampilkan sepasang murid untuk berbalas pantun dalam satu waktu. Tentukan total waktu yang dibutuhkan seluruh kemungkinan pasang murid kelas 6A dan 6B untuk berbalas pantun pada panggung.

Namun, pada acara perpisahan ini, ada aturan tambahan yang lebih rumit:

1. **Penundaan Tertentu:** Jika panjang pantun yang dimiliki oleh murid kelas 6A lebih panjang dari murid kelas 6B, maka ada penundaan sebesar $(A[i]-B[j])$ detik untuk setiap pasangan tersebut.
2. **Penyelesaian Akhir:** Setelah seluruh pasangan berbalas pantun, Anda diminta untuk menghitung total waktu yang dibutuhkan untuk seluruh acara tersebut.

Tugas Anda adalah menghitung total waktu yang dibutuhkan seluruh pasangan murid untuk berbalas pantun, dengan mempertimbangkan aturan - aturan di atas.

Input 1:

```
2
1 2
3 4
```

Output 1:

```
20
```

Penjelasan: Untuk contoh pertama, terdapat 4 kemungkinan pasangan murid yang akan tampil, yaitu setiap murid di kelas 6A akan berbalas pantun dengan setiap murid di kelas 6B. Berikut adalah perhitungan waktu yang dibutuhkan untuk setiap pasangan murid:

1. Murid 1 kelas 6A dan murid 1 kelas 6B:
Total waktu = $1 + 3 = 4$ detik
Penundaan (karena $1 < 3$) = 0 detik, jadi total waktu = 4 detik.
2. Murid 1 kelas 6A dan murid 2 kelas 6B:
Total waktu = $1 + 4 = 5$ detik
Penundaan (karena $1 < 4$) = 0 detik, jadi total waktu = 5 detik.
3. Murid 2 kelas 6A dan murid 1 kelas 6B:
Total waktu = $2 + 3 = 5$ detik
Penundaan (karena $2 < 3$) = 0 detik, jadi total waktu = 5 detik.

4. Murid 2 kelas 6A dan murid 2 kelas 6B:

Total waktu = $2 + 4 = 6$ detik

Penundaan (karena $2 < 42 < 42 < 4$) = 0 detik, jadi total waktu = 6 detik.

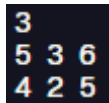
Total waktu:

Total waktu yang dibutuhkan untuk semua pasangan adalah:

$4 + 5 + 5 + 6 = 20$ detik.

Jadi, waktu total yang diperlukan untuk seluruh pasangan murid berbalas pantun adalah 20 detik.

Input 2:



3		
5	3	6
4	2	5

Output 2:



87
