

# Algoritma & Pemrograman

Pertemuan 11:

- PROSEDUR dan FUNGSI dalam C
  - OPERASI FILE dalam C

# Mengapa perlu ada prosedur dan fungsi?

- Semakin besar cakupan permasalahan yang akan diselesaikan, maka semakin banyak pula baris kode/perintah program yang harus ditulis
  - Contoh:
    - Windows XP memiliki 45 juta baris kode ([wikipedia.com](http://wikipedia.com))
    - Linux kernel 2.6.35 memiliki 13.5 juta baris kode
- Program yang memiliki baris kode yang banyak perlu diatur agar:
  - Memudahkan kita (yakni programmer) untuk menyelesaikan permasalahan (yakni dengan menulis program) secara sistematis → mulai dari hal yang abstrak lalu menuju ke hal yang lebih detail
  - Memudahkan kita untuk menelusuri program jika terdapat *bug*
  - Memudahkan kita atau programmer lain untuk menambahkan baris kode lain sebagai bentuk pengembangan aplikasi

# Prosedur/Fungsi/Method

- Merupakan pengelompokan beberapa instruksi/baris program yang melakukan sebuah perhitungan/komputasi tertentu
- Prosedur dapat menerima **parameter input** dan menghasilkan/mengembalikan **parameter output**
  - **Parameter input** : variabel yang diberikan pada prosedur untuk diproses di dalam prosedur
  - **Parameter output** : disebut juga nilai kembalian (*return value*), yakni variabel yang merupakan “output” dari prosedur
- Sebuah prosedur boleh tidak memiliki parameter input dan/atau parameter output

# Prosedur/Fungsi/Method

- Prosedur tidak dapat berdiri sendiri dalam sebuah program, ia harus dipanggil oleh program lain (dapat oleh program utama atau prosedur lainnya)
- Prosedur umumnya ditulis sebelum bagian `main()` dari program
- Cara pemanggilan prosedur adalah dengan menuliskan namanya
  - Jika prosedur memiliki parameter input, maka parameter input disertakan dalam pemanggilan prosedur
  - Jika prosedur memiliki parameter output, maka perlu disiapkan **tempat penampung** (dalam bentuk variabel) untuk menerima output dari prosedur
- Tidak ada ketentuan wajib untuk penamaan sebuah prosedur, namun biasanya menggunakan kata kerja.

# Perbedaan Prosedur, Fungsi, dan Method

- **Prosedur**: tidak memiliki nilai kembalian (*return value*)
- **Fungsi**: memiliki nilai kembalian (*return value*)
- Dalam bahasa C, istilah fungsi digunakan untuk mewakili baik prosedur maupun fungsi
- **Method**: sama dengan fungsi, biasanya digunakan pada bahasa pemrograman yang berorientasi objek.

# Ruang lingkup variabel

- Terdapat dua jenis ruang lingkup variabel:
  - **Variabel global**
    - Dideklarasikan di luar fungsi
    - Dapat dibaca oleh baris program manapun, termasuk dari dalam sebuah fungsi
  - **Variabel lokal**
    - Dideklarasikan di dalam sebuah fungsi
    - Hanya dapat dibaca di dalam fungsi tempat variabel dideklarasikan

# Contoh Fungsi 1: Menghitung Luas Segitiga

(a) Tanpa parameter input & output

```
1  #include <stdio.h>
2
3  void hitungLuasSegitiga() {
4      float alas,tinggi,luas;
5      printf("Panjang alas segitiga:");
6      scanf("%f",&alas);
7      printf("Tinggi segitiga:");
8      scanf("%f",&tinggi);
9      luas=(alas*tinggi)/2;
10     printf("Luas segitiga=%f",luas);
11 }
12 int main() {
13     hitungLuasSegitiga();
14 }
```

(b) Dengan parameter input

```
1  #include <stdio.h>
2
3  void hitungLuasSegitiga(float a,float t){
4      float alas,tinggi,luas;
5      alas=a;
6      tinggi=t;
7      luas=(alas*tinggi)/2;
8      printf("Luas segitiga=%f",luas);
9  }
10 int main() {
11     float a,t;
12     printf("Panjang alas segitiga:");
13     scanf("%f",&a);
14     printf("Tinggi segitiga:");
15     scanf("%f",&t);
16     hitungLuasSegitiga(a,t);
17 }
```

# Contoh Fungsi 1: Menghitung Luas Segitiga

(c) Dengan parameter input & *return value*

```
1  #include <stdio.h>
2
3  float hitungLuasSegitiga(float a, float t) {
4      float alas, tinggi, luas;
5      alas=a;
6      tinggi=t;
7      luas=(alas*tinggi)/2;
8      return luas;
9  }
10 int main() {
11     float a, t, L;
12     printf("Panjang alas segitiga:");
13     scanf("%f", &a);
14     printf("Tinggi segitiga:");
15     scanf("%f", &t);
16     L=hitungLuasSegitiga(a, t);
17     printf("Luasnya=%f", L);
18 }
```



## Contoh Fungsi 2: Mencari Pembagi Bersama Terbesar (Greatest Common Divisor-GCD) dari dua bilangan

- Cari GCD dari 80 dan 12.
  - Semua faktor pembagi dari 80: 1,2,**4**,5,8,10,16,20,40,80
  - Semua faktor pembagi dari 12: 1,2,3,**4**,6,12
  - Maka gcd dari 80 dan 12 = 4
  - Langkah-langkah:
    - 80 dibagi 12 hasilnya=6, sisa 8
    - 12 dibagi 8 hasilnya=1, sisa 4
    - 8 dibagi 4 hasilnya=2, sisa 0
    - Karena pembagian terakhir menghasilkan sisa 0, maka sisa pembagian terakhir sebelum 0, yakni 4, menjadi gcd(80,12).
    - $\text{gcd}(80,12) = \text{gcd}(12,8) = \text{gcd}(8,4) = \text{gcd}(4,0) = 4$

$$\begin{array}{l} 80 = 6 * 12 + 8 \\ \downarrow \quad \downarrow \\ 12 = 1 * 8 + 4 \\ \downarrow \quad \downarrow \\ 8 = 2 * 4 + 0 \end{array}$$

## Contoh Fungsi 2: Mencari Pembagi Bersama Terbesar (Greatest Common Divisor-GCD) dari dua bilangan

- Algoritma Euclidean: [Diberikan 2 bilangan bulat positif,  $m$  dan  $n$  ( $m \geq n$ ). Algoritma Euclidean mencari pembagi bersama terbesar, gcd, dari kedua bilangan tersebut, yakni sebuah bilangan bulat positif yang habis membagi  $m$  dan  $n$ ]
  1. Jika  $n=0$ , maka
    - a.  $m$  adalah jawabannya
    - b. Stop
    - c. Tetapi jika  $n \neq 0$ , lanjutkan ke langkah 2
  2. Bagi  $m$  dengan  $n$  dan misalkan  $r$  adalah sisanya
  3. Ganti nilai  $m$  dengan nilai  $n$  dan nilai  $n$  dengan nilai  $r$ , lalu ulangi kembali ke langkah 1

## Contoh Fungsi 2: Mencari Pembagi Bersama Terbesar (Greatest Common Divisor-GCD) dari dua bilangan

```
1  #include <stdio.h>
2
3  int findGCD(int m, int n) {
4      int r;
5      while (n!=0) {
6          r=m%n;
7          m=n;
8          n=r;
9      }
10     if(n==0)
11         return m;
12
13 }
14 int main()
15 {
16     int m,n,gcd;
17     printf("Masukkan bilangan positif pertama (m):");
18     scanf("%d",&m);
19     printf("Masukkan bilangan positif kedua (n):");
20     scanf("%d",&n);
21     if (m<n) {
22         int x=m;
23         m=n;
24         n=x;
25     }
26     gcd=findGCD(m,n);
27     printf("GCD-nya=%d",gcd);
28 }
```

# Operasi File pada C

- Dalam pemrograman skala menengah dan besar, input dan output data dalam program sering berjumlah banyak dan perlu ditulis/disimpan dalam sebuah file
- Umumnya file berupa file teks atau biner
- Ada 3 mode pengaksesan file: **READ** (**r**), **WRITE** (**w**), dan **APPEND** (**a**)
- Sebelum dapat diakses, file harus dibuka terlebih dahulu
- Setelah selesai diakses, file harus ditutup kembali
- Tipe variabel untuk operasi file adalah `FILE`
  - Contoh deklarasi nama variabel file:  
`FILE *f;` (variabel untuk mengakses file, tanda \* menandakan variabel pointer)

# Sintaks Operasi File pada C

```
FILE *f;
```

Operasi	Sintaks
Buka file ( <i>read</i> )	<code>f=fopen("hasil.txt", "r");</code>
Buka file ( <i>write</i> )	<code>f=fopen("hasil.txt", "w");</code>
Buka file ( <i>append</i> )	<code>f=fopen("hasil.txt", "a");</code>
Buka file ( <i>read &amp; write</i> )	<code>f=fopen("hasil.txt", "r+");</code> <code>f=fopen("hasil.txt", "w+");</code> <code>f=fopen("hasil.txt", "a+");</code>
Buka file biner ( <i>read</i> )	<code>f=fopen("hasil.txt", "rb");</code>
Buka file biner ( <i>write</i> )	<code>f=fopen("hasil.txt", "wb");</code>
Buka file biner ( <i>append</i> )	<code>f=fopen("hasil.txt", "ab");</code>
Buka file biner ( <i>read &amp; write</i> )	<code>f=fopen("hasil.txt", "rb+");</code> <code>f=fopen("hasil.txt", "wb+");</code> <code>f=fopen("hasil.txt", "ab+");</code>
Tutup file	<code>fclose(f); //returns 0 if success,</code> <code>//or EOF if error occurs</code>

# Sintaks Operasi File pada C

```
FILE *f;  
char nama[10];  
int umur;  
char c[]="Surabaya kota Pahlawan";  
char buffer[100];
```

Operasi	Contoh Sintaks
Membaca data	<pre>fscanf(f, %s %d, nama, &amp;umur);</pre> <p>//membaca dari file <code>f</code> untuk 2 data bertipe string dan int dan menyimpannya ke variabel <code>nama</code> dan <code>umur</code></p>
	<ul style="list-style-type: none"><li><pre>fread(x, sizeof(x), 1, f);</pre><p>//membaca nilai ke variabel <code>x</code> dengan ukuran (<i>sizeof</i>) sebesar ukuran tipe <code>x</code> sebanyak 1 elemen dari file <code>f</code></p></li><li><pre>fread(buffer, strlen(c)-1, 1, f);</pre><p>//membaca nilai ke variabel <code>buffer</code> dengan ukuran sebesar panjang string <code>c-1</code> sebanyak 1 elemen dari file <code>f</code></p></li></ul>
Menulis data	<pre>fprintf(f, "%s %d\n", nama, umur);</pre> <p>//menulis ke file <code>f</code> untuk 2 data bertipe string dan int dari variabel <code>nama</code> dan <code>umur</code></p>
	<pre>fwrite(c, strlen(c)+1, 1, f);</pre> <p>//menulis data dari variabel <code>c</code> dengan ukuran sebesar panjang string <code>c+1</code> sebanyak 1 elemen ke file <code>f</code></p>

# Contoh 1

```
1  #include<stdio.h>
2  #include<conio.h>
3
4  int main(){
5      char name[10];
6      int age;
7      FILE *p,*q;
8      p = fopen("hasil.txt", "a");
9      q = fopen("hasil.txt", "r");
10     printf("Enter Name and Age:");
11     scanf("%s %d", name, &age);
12     fprintf(p,"%s %d\n", name, age);
13     fclose(p);
14     do
15     {
16         fscanf(q,"%s %d", name, &age);
17         printf("%s %d\n", name, age);
18     }
19     while( !feof(q) );
20     getch();
21     fclose(q);
22     return 0;
23 }
```

Keterangan baris program:

- 7:** deklarasi 2 variabel pointer file
- 8:** membuka file `hasil.txt` untuk append
- 9:** membuka file `hasil.txt` untuk read
- 12:** menyimpan data nama dan umur ke dalam file
- 16:** membaca data dari file dan menyimpannya di variabel `name` dan `age`
- 17:** mencetak ke layar isi dari variabel `name` dan `age`
- 19:** mengerjakan baris 14-18 selama belum mencapai end of file dari pointer file `q`

# Contoh 2

```
int fseek(FILE *stream, long int offset, int whence)
```

stream: pointer ke file

offset: besar pergeseran (dalam bytes) dalam menentukan posisi file

whence: posisi awal penetapan (yang akan ditambah dengan offset).

Nilainya salah satu dari konstanta berikut:

SEEK\_SET : awal file

SEEK\_CUR : *current position* dari file pointer

SEEK\_END : akhir file

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      FILE *fp;
7      char c[] = "this is tutorialspoint";
8      char buffer[100];
9
10     /* Open file for both reading and writing */
11     fp = fopen("file.txt", "w+");
12
13     /* Write data to the file */
14     fwrite(c, strlen(c) + 1, 1, fp);
15
16     /* Seek to the beginning of the file */
17     fseek(fp, 0, SEEK_SET);
18
19     /* Read and display data */
20     fread(buffer, strlen(c)+1, 1, fp);
21     printf("%s\n", buffer);
22     fclose(fp);
23
24     return(0);
25 }
```



# Latihan Fungsi

- Buat fungsi jarak yang menerima 2 masukan berupa 2 buah titik  $P_1 (x_1, y_1)$  dan  $P_2 (x_2, y_2)$  dan menghitung jarak kedua titik tersebut (Gunakan rumus Euclidean)
- Buat fungsi untuk menambahkan dua jam (dalam format hh:mm:ss)
- Buat fungsi lower yang mengubah huruf kecil menjadi huruf besar (kapital)
- Buat fungsi untuk menghitung selisih tanggal (dalam format dd-mm-yy)