IF1210 Algoritma dan Pemrograman 1

Paradigma Pemrograman Prosedural & Notasi Algoritmik

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Agenda

- Pemrograman Prosedural
- Notasi Algoritmik

Pemrograman Prosedural

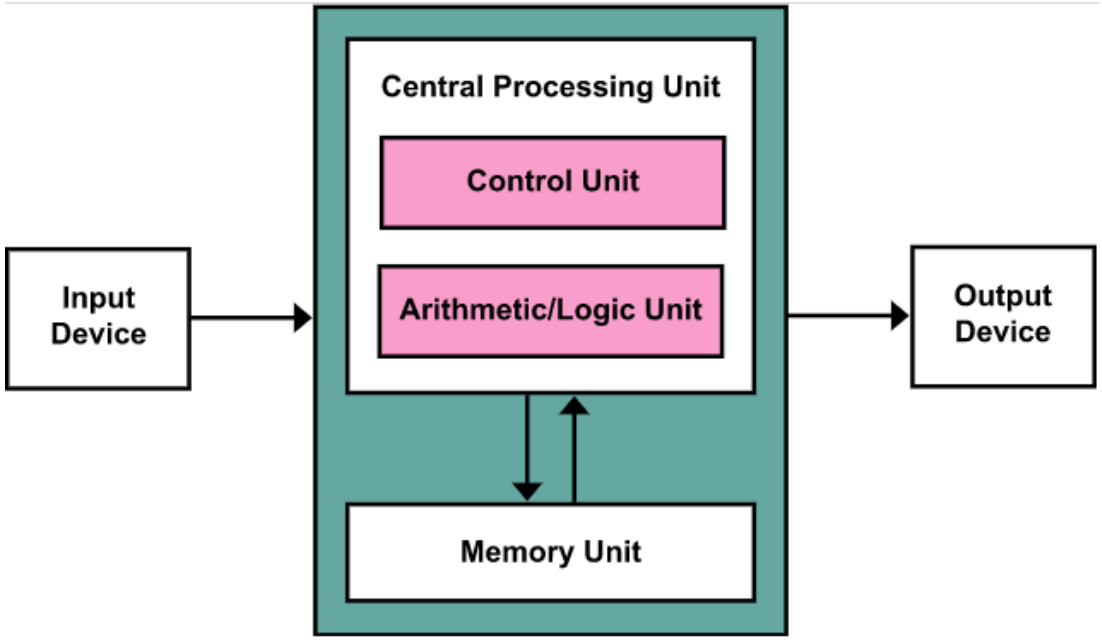
Berpikir Komputasional

- Perlunya kemampuan: Computational Thinking
- Konsep dasar berpikir komputasi:
  - Dekomposisi masalah
  - Pengenalan Pola
  - Generalisasi Pola dan Abstraksi
  - Perancangan Algoritma
  - Analisis Data dan Visualisasi
- Memformulasikan persoalan komputasi dan menyelesaikannya dengan bantuan komputer → pemrograman
- Paradigma pemrograman: sudut pandang penyelesaian persoalan dengan program

Paradigma Imperatif/Prosedural

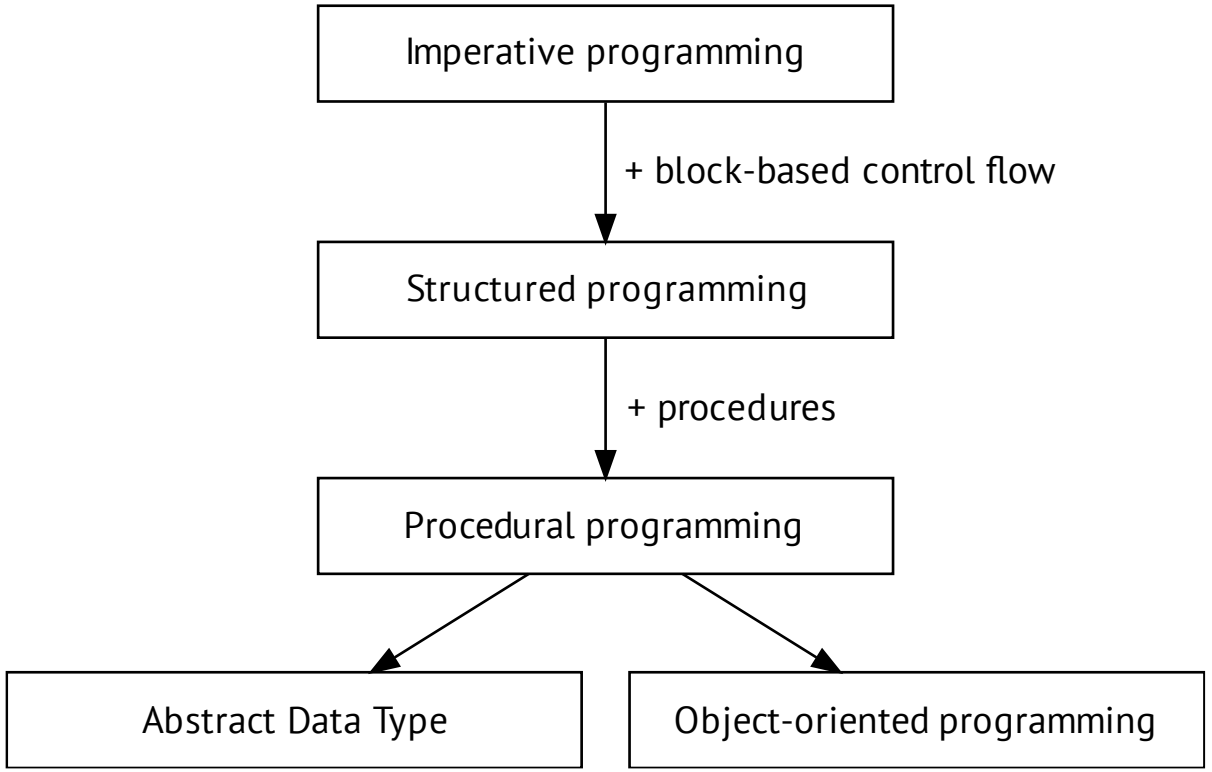
- Didasari oleh konsep mesin Von Neumann (*stored program concept*)
- Program didasari oleh **strukturisasi informasi** di dalam memori dan **manipulasi** dari informasi yang disimpan tersebut

Program = Algoritma + Struktur Data



Skema arsitektur mesin Von Neumann

Sejarah paradigma prosedural



Imperative Programming

- Paradigma mula-mula dalam pemrograman komputer.
- Didasari pada cara kerja komputer yang mengeksekusi instruksi satu per satu secara berurutan.
  - Program pun ditulis sebagai suatu **untaian instruksi** berbentuk *statement*.
- Merupakan paradigma memrogram dalam bahasa mesin ataupun *assembly*.
- Menggunakan *statement* ”goto” untuk alur kendali (*control flow*).

Contoh: loop pada *imperative programming*

KODE (C)	HASIL EKSEKUSI

`#include <stdio.h>`
 	
`int main() {`
`  int i = 0;`
`  loop:`
`  printf("%d\n", i);`
`  if (++i < 5) goto loop;`
`  printf("Done!\n");`
`}`
	0
	1
	2
	3
	4
	Done!

Structured programming

- *Statement* goto dianggap berbahaya karena dapat membuat alur program lompat ke mana saja → sepotong kode dapat mengakses memori yang tidak valid:

```
int x = 0;
goto there;
int y = 1;
there:
printf("%d, %d\n", x, y);
```

- Karena itu diperkenalkan ***control flow* berbasis blok** dengan lingkup (*scope*) memori yang terbatas pada setiap blok.
  - Kode di luar blok tidak dapat mengakses memori milik blok tersebut.

Contoh: lingkup sebuah blok

KODE (C)

```c
#include <stdio.h>

int main() {
  int x = 5;
  if (x != 10) {
    int y = 8;
    printf("%d\n", y);
    // di sini `z` tidak dikenali
  } else {
    int z = 9;
    printf("%d\n", z);
    // di sini `y` tidak dikenali
  }
  // di sini `y` dan `z` tidak dikenali
}
```

HASIL EKSEKUSI

```
8
```

Procedural programming

- Menambahkan *reusability* melalui subprogram berbentuk **prosedur** dan **fungsi**.
  - Prosedur: mengubah *state* program, tidak mengembalikan sebuah nilai.
  - Fungsi: melakukan pemetaan nilai (dalam parameter fungsi) ke nilai lain (sebagai *return value*). **Tidak** mengubah *state* program.
  - *State* program dalam konteks ini: nilai variabel di luar prosedur/fungsi.

Contoh: prosedur

KODE (C)

```c
#include <stdio.h>

void swap(int *xp, int *yp) {
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

int main() {
  int x = 5;
  int y = 10;
  printf("x=%d, y=%d\n", x, y);
  swap(&x, &y);
  printf("x=%d, y=%d\n", x, y);
}
```

HASIL EKSEKUSI

```
x=5, y=10
x=10, y=5
```

Contoh: fungsi

KODE (C)	HASIL EKSEKUSI

`#include <stdio.h>`
 	
`int square(int x) {`
`  return x * x;`
`}`
 	
`int main() {`
`  int a = 12;`
`  int sq = square(a);`
`  printf("%d² = %d\n", a, sq);`
`}`
	`12² = 144`

+ Abstract data type (ADT)

- Sebagaimana prosedur dan fungsi meningkatkan *reusability* pada aspek **algoritma**, ADT meningkatkan *reusability* pada aspek **struktur data.**
- Dibahas lebih lanjut pada paruh kuliah berikutnya.

Notasi Algoritmik

Perlunya Notasi Algoritmik (1)

- Tidak mungkin mempelajari semua bahasa pemrograman → yang penting dipahami: **pola pikir komputasi** dan **paradigma pemrograman**

Belajar memrogram ≠ Belajar bahasa pemrograman

- **Notasi algoritmik**: notasi standar yang digunakan untuk menuliskan teks algoritma [dalam paradigma prosedural]
  - **Algoritma** adalah solusi detail secara prosedural dari suatu persoalan dalam notasi algoritmik.
  - **Program** adalah program komputer dalam suatu bahasa pemrograman yang tersedia di dunia nyata.

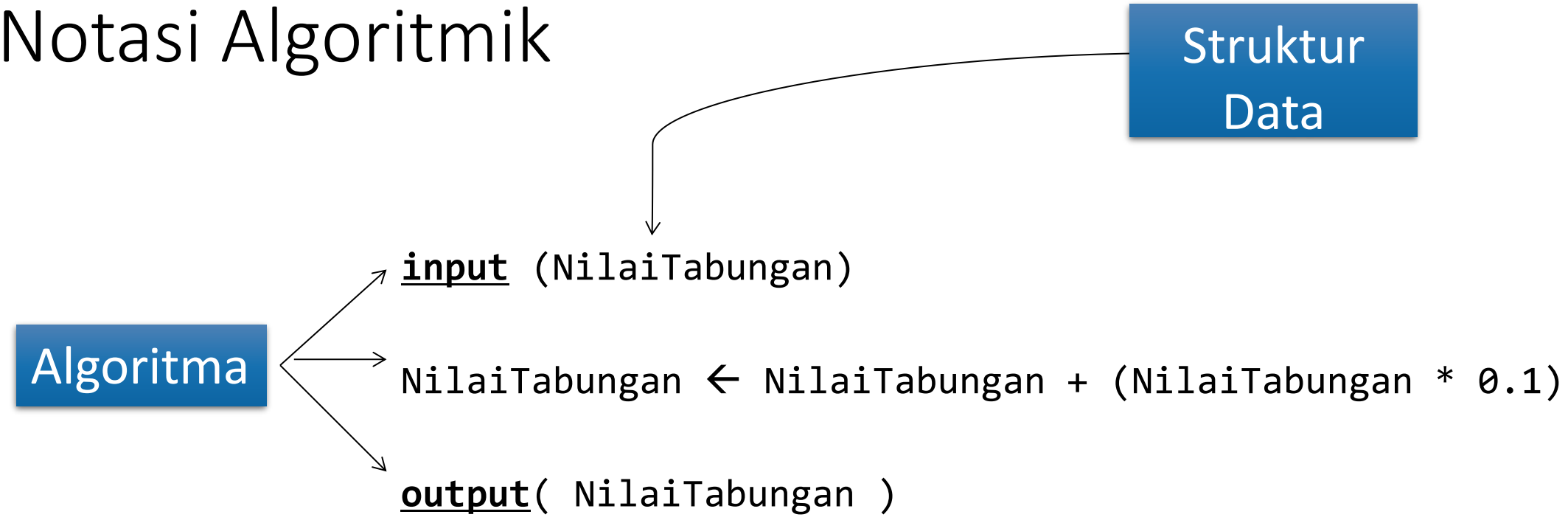
Perlunya Notasi Algoritmik (2)

- Notasi algoritmik → representasi cara berpikir untuk menyelesaikan persoalan dengan paradigma prosedural
- Membuat program → melihat "kosakata" translasi notasi algoritmik ke bahasa pemrograman yang dipilih
- Dengan notasi algoritmik yang sama, bisa dibuat program dalam bahasa pemrograman yang berbeda, dengan paradigma pemrograman yang sama
  - Contoh: untuk prosedural: Pascal, , C/C++, Fortran

Contoh Kasus: Tabungan

Program = Algoritma + Struktur Data

Notasi Algoritmik



Kode Program Bahasa Python

```
input (NilaiTabungan)
NilaiTabungan ← NilaiTabungan + (NilaiTabungan * 0.1)
output(NilaiTabungan)
```

Input

```
NilaiTabungan = int(input())
NilaiTabungan = NilaiTabungan + NilaiTabungan * 0.1
print(NilaiTabungan)
```

Output

Kode Program Bahasa Pascal

```
input (NilaiTabungan)
NilaiTabungan ← NilaiTabungan + (NilaiTabungan * 0.1)
output(NilaiTabungan)
```

readln akan membaca dari hasil ketik di keyboard

writeln akan menulis hasil di layar komputer

```
readln(NilaiTabungan);
NilaiTabungan := NilaiTabungan + NilaiTabungan * 0.1;
writeln(NilaiTabungan);
```

Kode Program Bahasa C

```
input (NilaiTabungan)
NilaiTabungan ← NilaiTabungan + (NilaiTabungan * 0.1)
output(NilaiTabungan)
```

scanf akan membaca dari hasil ketik di keyboard

```
scanf("%d",&NilaiTabungan);
NilaiTabungan = NilaiTabungan + NilaiTabungan * 0.1;
printf("NilaiTabungan = %d\n",NilaiTabungan);
```

Printif: mencetak string ke layar

Kode Program Bahasa C++

```
input (NilaiTabungan)
NilaiTabungan ← NilaiTabungan + (NilaiTabungan * 0.1)
output(NilaiTabungan)
```

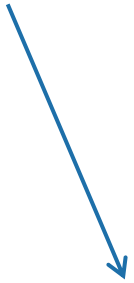
cin: Console Input (diketikkan lewat keyboard)

```
cin >> NilaiTabungan;
NilaiTabungan = NilaiTabungan + NilaiTabungan * 0.1;
cout << NilaiTabungan;
```

cout: Console Output

Kode Program Bahasa Fortran

```
input (NilaiTabungan)
NilaiTabungan ← NilaiTabungan + NilaiTabungan * 10%
output(NilaiTabungan)
```



read akan membaca dari hasil ketik di keyboard

```
read *, NilaiTabungan
NilaiTabungan = NilaiTabungan + NilaiTabungan * 0.1;
print *, NilaiTabungan
```

print akan menulis hasil di layar komputer

Struktur Dasar Program Prosedural - Notasi Algoritmik

Program <JudulProgram> { Spesifikasi Program }
KAMUS { Deklarasi type, variabel, konstanta, fungsi, prosedur }
ALGORITMA { Deretan langkah algoritmik untuk penyelesaian persoalan }

Struktur Dasar Program – Bahasa C

```
{ Program <JudulProgram> }
{ Spesifikasi Program }

int main () {
    { KAMUS }
    { Deklarasi variabel, konstanta, fungsi, prosedur }

    { ALGORITMA }
    { Deretan langkah algoritmik untuk penyelesaian persoalan }
    return 0;
}
```

Contoh Program Notasi Algoritmik

```
Program Test
{ Spesifikasi Program: menghitung A + B }
```

```
KAMUS
{ Deklarasi variabel }
      A, B : integer
```

```
ALGORITMA
      input(A)
      input(B)
      A ← A + B
      output(A)
      output(B)
```

Contoh Program C

```
{ Program Test }
{ Spesifikasi : Menghitung nilai A dan B }
#include <stdio.h>

int main () {
    { KAMUS }
    int A, B;

    { ALGORITMA }
    scanf("%d",&A);      { input }
    scanf("%d",&B);      { input }

    A = A + B;           { proses }

    printf("%d\n",A);    { output }
    printf("%d\n",B);    { output }
    return 0;
}
```

Untuk selanjutnya hanya akan dibahas notasi algoritmik. Bahasa C akan dibahas kembali pada pertemuan berikutnya.

Komentar

- Dalam bahasa pemrograman komentar adalah bagian program yang tidak dieksekusi
  - Bagian ini hanya digunakan untuk memberikan penjelasan suatu langkah, rumus, ataupun bisa hanya berupa keterangan

Notasi Algoritmik

```
{ ini komentar }
```

Kamus

- Kamus dipakai untuk mendeklarasi nama-nama yang digunakan dalam program
- Deklarasi nama yang didefinisikan pemrogram
  - type
  - variabel
  - konstanta
- Deklarasi BUKAN instruksi

Notasi Algoritmik

```
i : integer
JumlahUang : real
Titik : Point
```

Variabel

- Contoh deklarasi dan inisialisasi variabel:

Notasi Algoritmik
KAMUS i : integer A : real
ALGORITMA .... i ← 100 A ← i * 50 ....

Tipe data primitif/Tipe dasar

- Disediakan oleh bahasa pemrograman

Notasi Algoritmik	Domain Nilai
boolean	true False
integer	bilangan bulat negatif, 0, bilangan bulat positif Contoh: 1; -144; 999; 0
real	bilangan riil, contoh: 3.14; 4.01E+1
character	Sebuah karakter, ditandai dengan kutip tunggal; Contoh: ‘A’; ‘#’; ‘b’
string	Kumpulan karakter, ditandai dengan kutip ganda Contoh: “xcxcx”; “AB”

Tipe Data Bentukan (1)

- Tipe data bentukan/komposit/record
  - Tidak tersedia secara otomatis, harus dibuat oleh programmer
  - Dibentuk dari gabungan tipe dasar
  - Dalam notasi algoritmik terdapat sintaks khusus untuk mendeklarasikan type
    - Komponen type dideklarasikan dengan nama khusus dan nama ini digunakan untuk mengakses komponen type

Tipe Data Bentukan (2)

	Notasi Algoritmik

Deklarasi type (KAMUS)	`type Point :`
	`  < x : integer;`
	`  y : integer >`

Deklarasi variabel (KAMUS)	`P : Point` `a, b : integer`

Akses elemen (ALGORITMA)	`P.x ← 5`
	`P.y ← 6`
	`a ← P.x`
	`b ← P.y + 1`

Konstanta

- Berbeda dengan variable, suatu konstanta **tidak boleh diubah** nilainya dalam algoritma
- Contoh deklarasi di KAMUS:

Notasi Algoritmik
constant PI : real = 3.14159
constant Nilai : integer = 1000
constant NMax : integer = 100

Assignment

- *Assignment*: Pemberian nilai suatu variabel
- Ruas kiri harus **variable**
- Ruas kanan harus **ekspresi/nilai/variabel yang sudah jelas nilainya**

Notasi Algoritmik

```
<RuasKiri> ← <RuasKanan>
```

Contoh:

```
i ← 10
Nama ← "Maya"
X ← i + 10
```

Nilai X di-assign dengan ekspresi

Jenis Ekspresi

	Notasi Algoritmik

Ekspresi Aritmatika: operan numerik (integer/real), hasil: numerik (integer/real)	`A + B`
	`x + 2 * y`
	`P – 2 * Q + R/S`

Ekspresi Relasional: operan numerik (integer/real), hasil: boolean	`A < B`
	`X = Y`
	`Total ≥ nilai`

Ekspresi Logika: operan: boolean, hasil : boolean	`A and B`
	`C or B`
	`not(true)`

Operator Tipe Dasar

Tipe Data	Notasi Algoritmik
Integer	* / + - div mod < > ≤ ≥ = ≠
Boolean	and or not =
Real	* / + - < > ≤ ≥ ≠
Character	= ≠
String	= ≠ +

Input/Output (1)

- Perintah **input**: pemberian nilai **variabel** dari piranti masukan, misal: keyboard → dibaca atas masukan dari pengguna

Notasi Algoritmik
`input`

`Contoh`:
 	
`input(A)`
`input(B)`
`input(C)`

Input/Output (2)

- Perintah **output**: penulisan nilai (variabel/konstanta/hasil ekspresi) ke piranti keluaran, misal: monitor/layar

Notasi Algoritmik
output
Contoh: output(A) output("Hello") output(A*4) output("A = ",A)

Latihan Soal

Kerjakan semua soal berikut dalam notasi algoritmik.

Soal 1

- Dalam Fisika, jarak (s) dapat dihitung berdasarkan kecepatan (v) dan waktu tempuh (t), yaitu: s = v * t
- Buatlah algoritma dalam notasi algoritmik untuk menghitung jarak (dalam m) berdasarkan masukan kecepatan (dalam m/s) dan waktu (dalam s).

Soal 2

- Buatlah program yang menerima input jari-jari lingkaran (bilangan riil, asumsikan > 0) dan menampilkan keliling lingkaran.
- PI (π) dinyatakan sebagai konstanta dengan nilai 3.14159

Soal 3

- Sebuah toko menjual kelereng. Berikut adalah tabel harga kelereng berdasarkan warnanya:

Warna kelereng	Harga 1 butir (dalam ratusan rupiah)
Merah	10
Hijau	15
Kuning	20

- Seorang anak membeli kelereng sejumlah m kelereng merah, h kelereng hijau, dan k kelereng kuning. Asumsikan m >= 0, h >= 0, k >= 0.
- Buatlah algoritma dalam notasi algoritmik yang menerima masukan m, h, dan k dan menghitung serta menampilkan berapa yang harus dibayarkan anak itu.

Soal 4

- Sebuah titik di atas bidang kartesian terdiri atas sebuah absis (x, bertipe real) dan ordinat (y, bertipe real)
- Buat program yang membaca sebuah data bertipe titik (misalnya P) dan menampilkannya ke layar dalam format: (x,y)

Lebih Lanjut ...

- Notasi algoritmik lebih lanjut [dan translasinya ke Bahasa C jika diperlukan] akan dipelajari satu per satu dalam materi-materi berikutnya
- Minggu depan: Bahasa C, coding standard

- Bahan bacaan:
  - Diktat "Dasar Pemrograman Bagian Pemrograman Prosedural" → dengan notasi algoritmik
  - Tersedia di situs Edunex