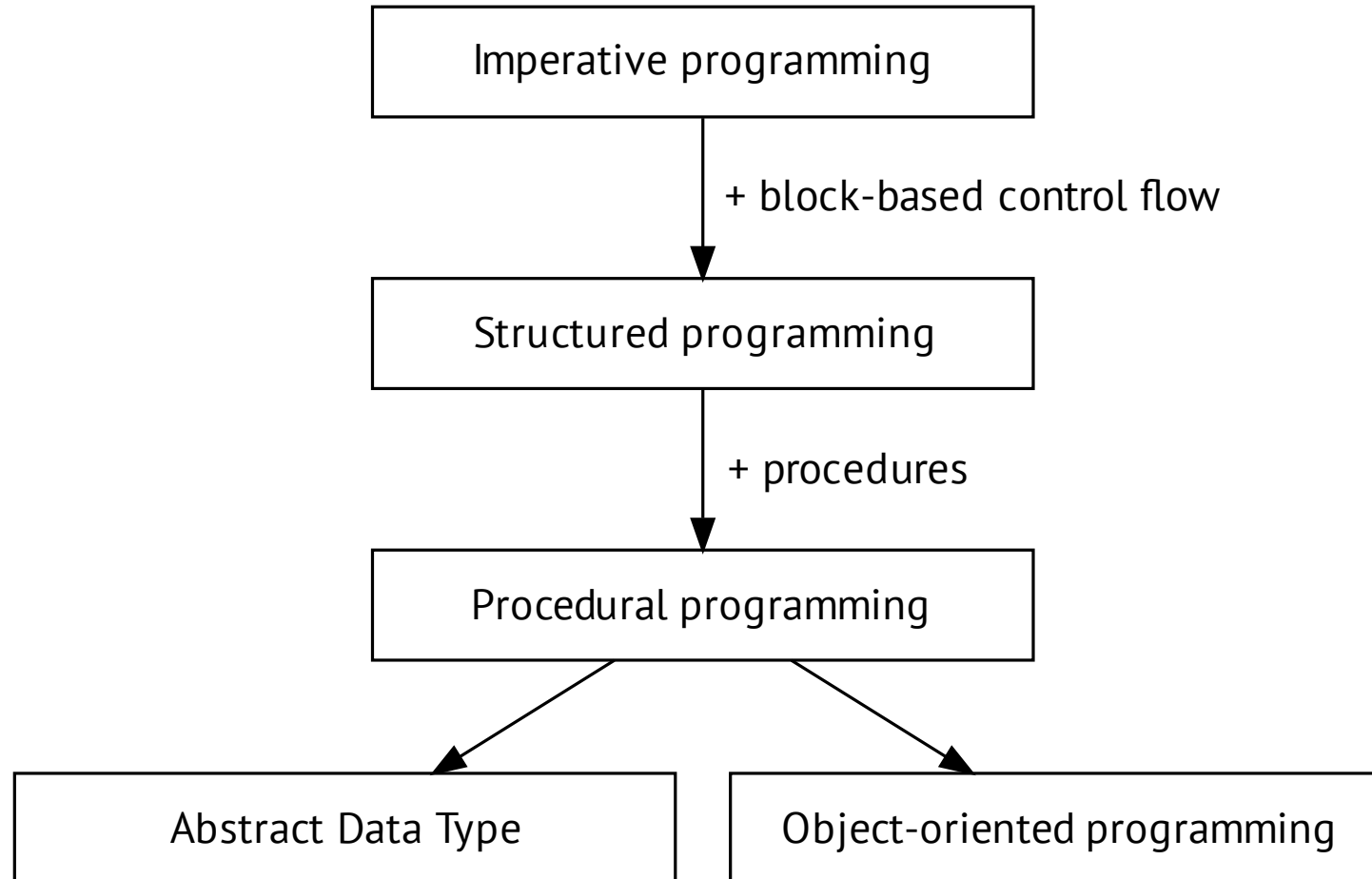


Paradigma Prosedural

IF2110 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

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

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)	HASIL EKSEKUSI
<pre>#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 }</pre>	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)	HASIL EKSEKUSI
<pre>#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); }</pre>	<pre>x=5, y=10 x=10, y=5</pre>

Contoh: fungsi

KODE (C)	HASIL EKSEKUSI
<pre>#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); }</pre>	<p>12² = 144</p>

+ Abstract data type (ADT)

Sebagaimana prosedur dan fungsi meningkatkan *reusability* pada aspek **algoritma**, ADT meningkatkan *reusability* pada aspek **struktur data**.

Dibahas lebih lanjut pada segmen berikutnya: **Algoritma, Struktur Data, dan ADT**.