visual programming

Function

Bachelor of Information Systems
Institut Teknologi Del

Learning Objective(s)

This material should address the following question(s).

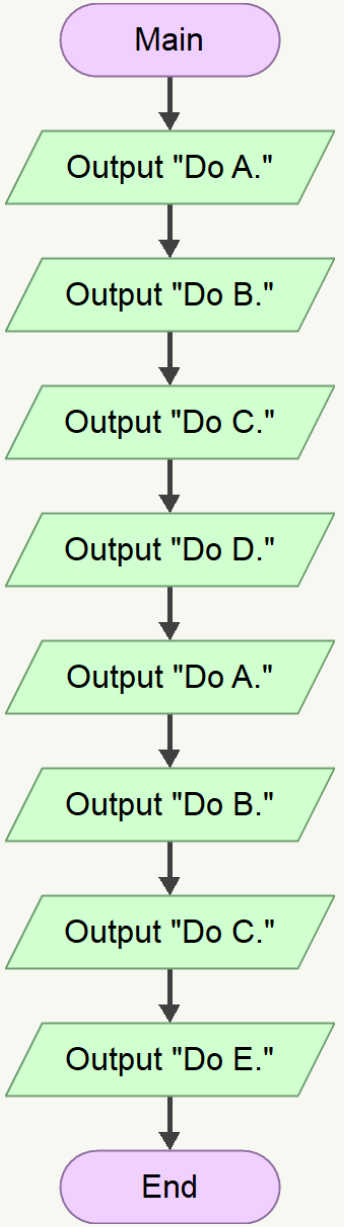
- What is function?
- How to use it?

Discussion Point

Function:
The Core Concepts.

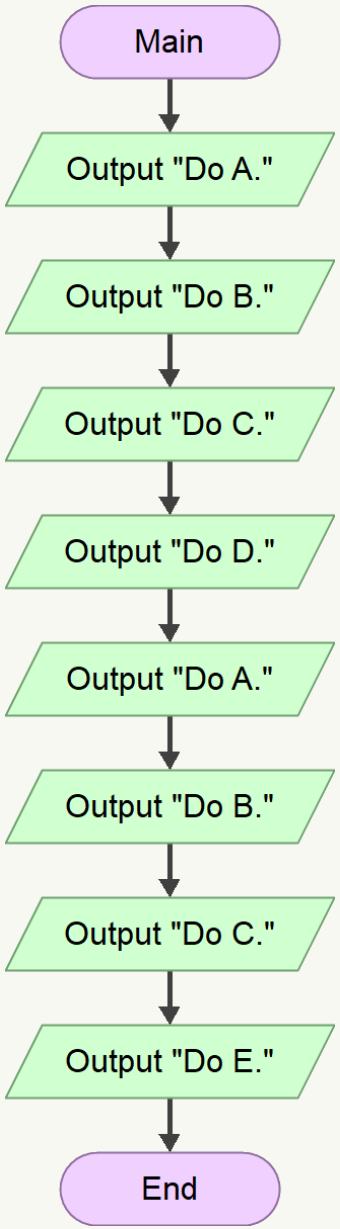
An Example

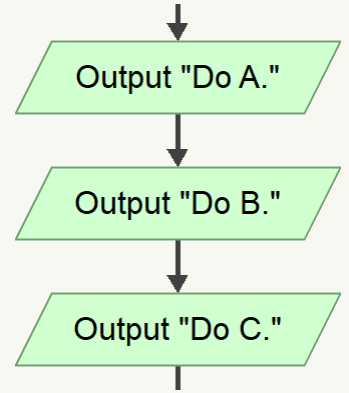
- A solution consists of a set of instructions.
  - Within it, repetitive patterns of instructions may exist.

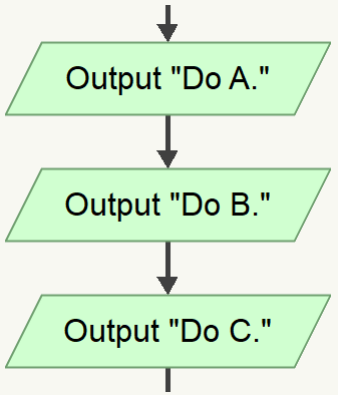


An Example

- What if, these repetitive instructions:
  - Appear in different places?



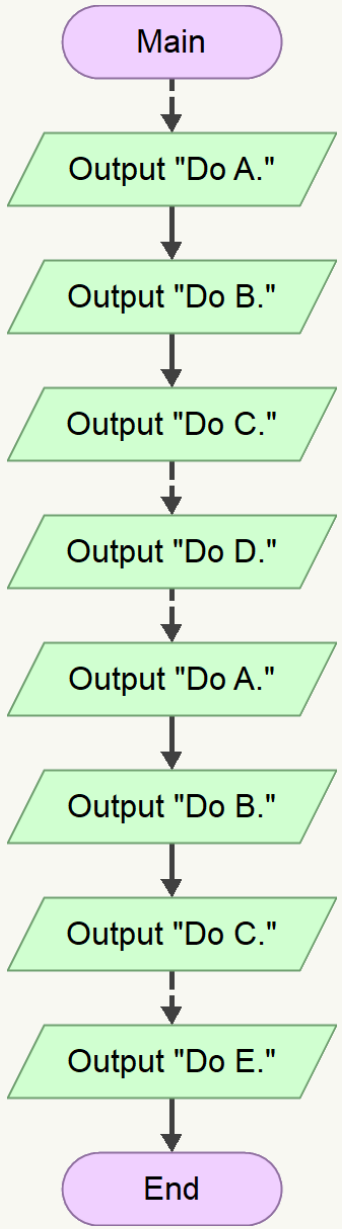


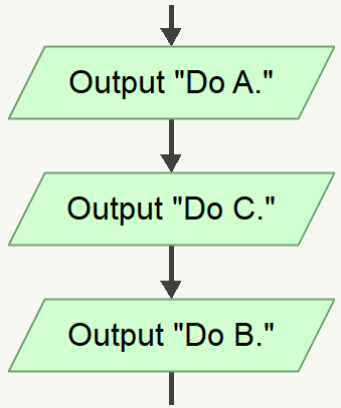


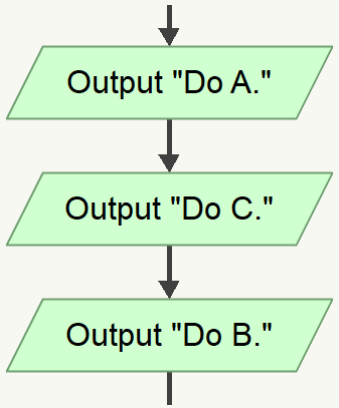
An Example

- What if, these repetitive instructions:
  - Appear in different places?
  - Are required to change sometimes in the future?

Could be **disastrous** and **impractical** when the solution is **gigantic** and very **complex**.







Can we **isolate** the repetitive instructions
and make them **callable** when needed?

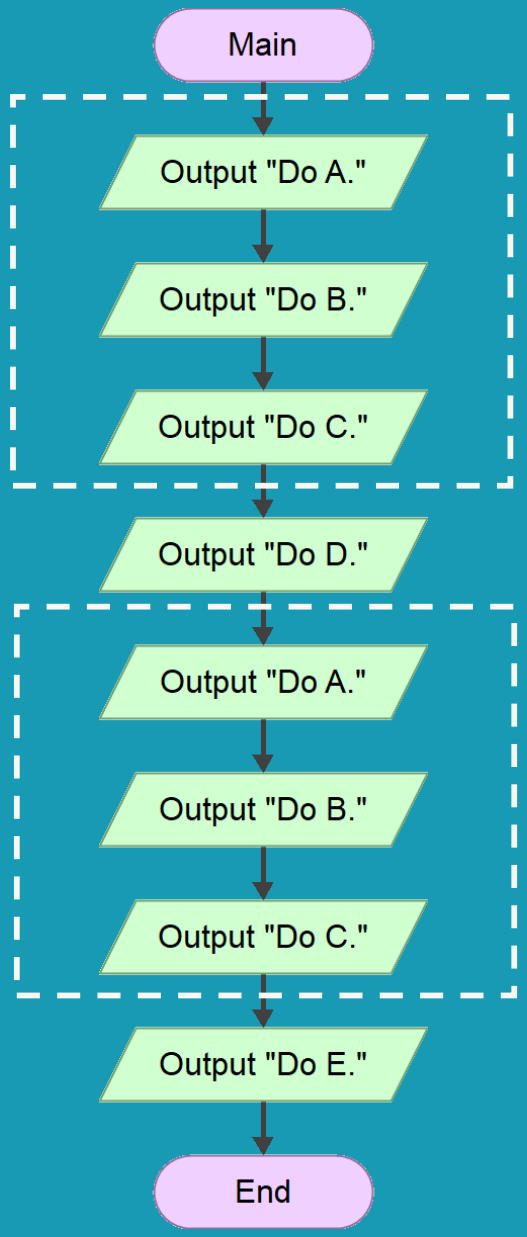
Definition

*A function is a **callable routine** with an atomic purpose. It is an essential building block to develop more complex routines.*

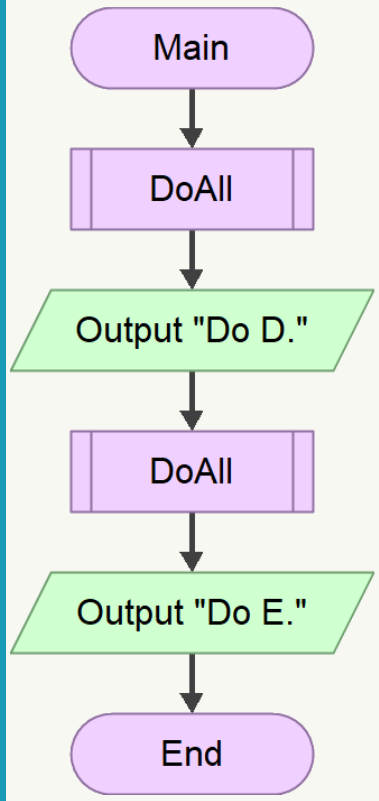
Function

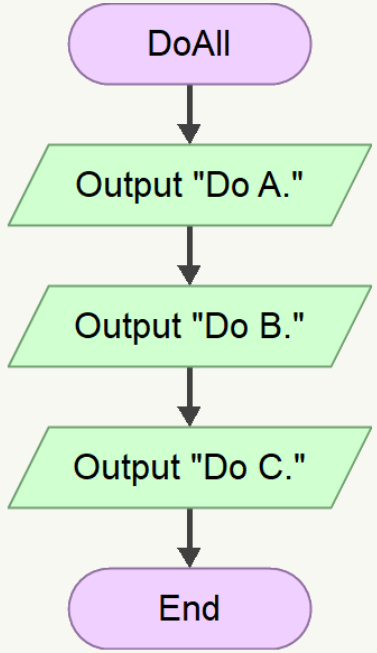
- A function has a unique identifier that is:
  - Meaningful and showing its purpose.
  - In the form of active action.

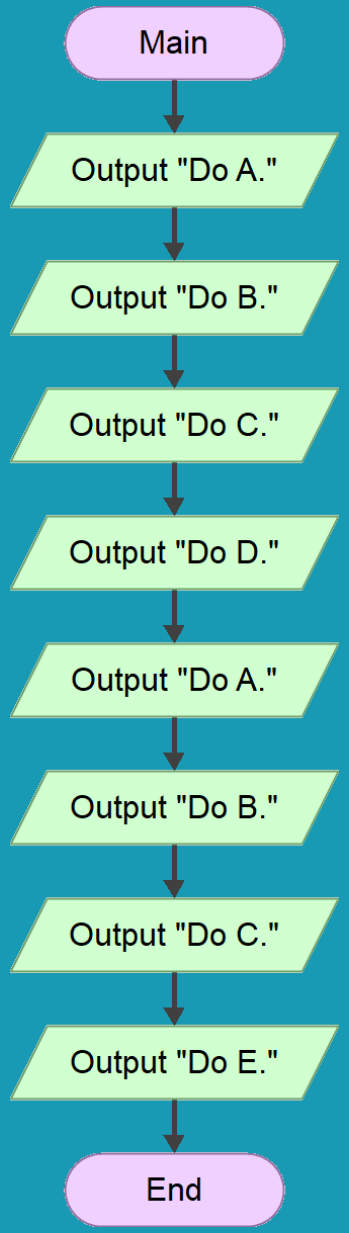
- E.g. `cancel`, `remove_first`, `showMessage`, etc.



Isolating instructions and call them at will.

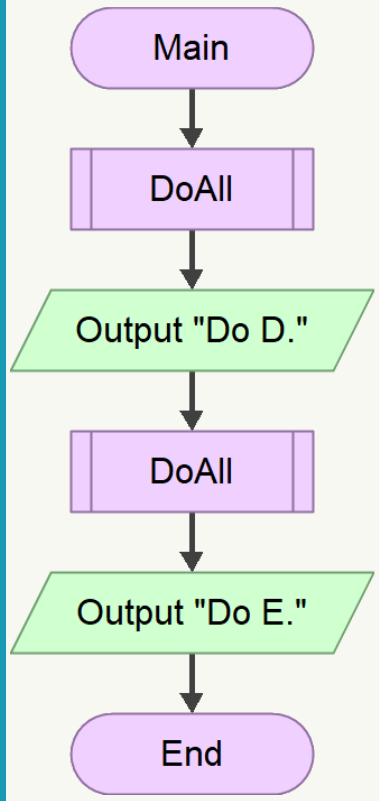


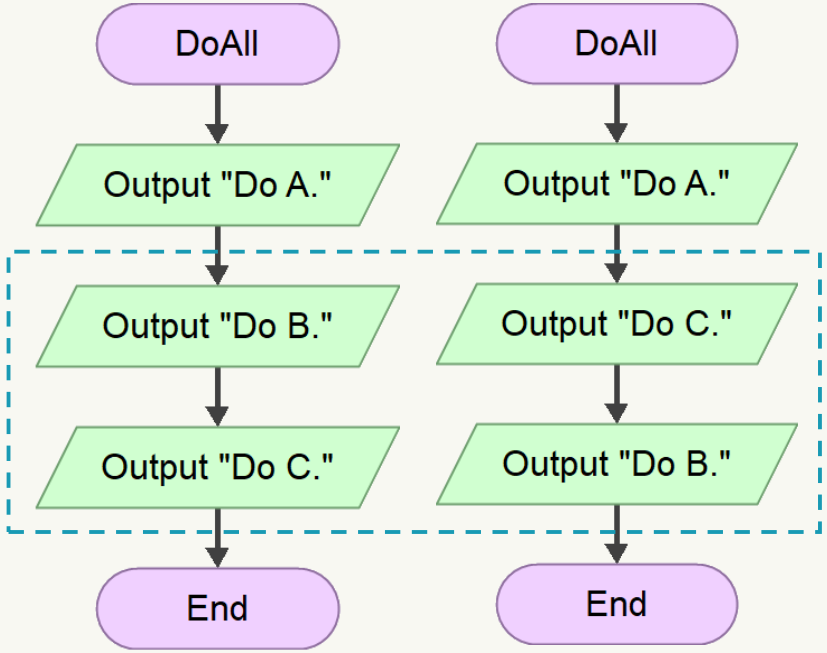




What about **changes**?

More manageable due to isolation.



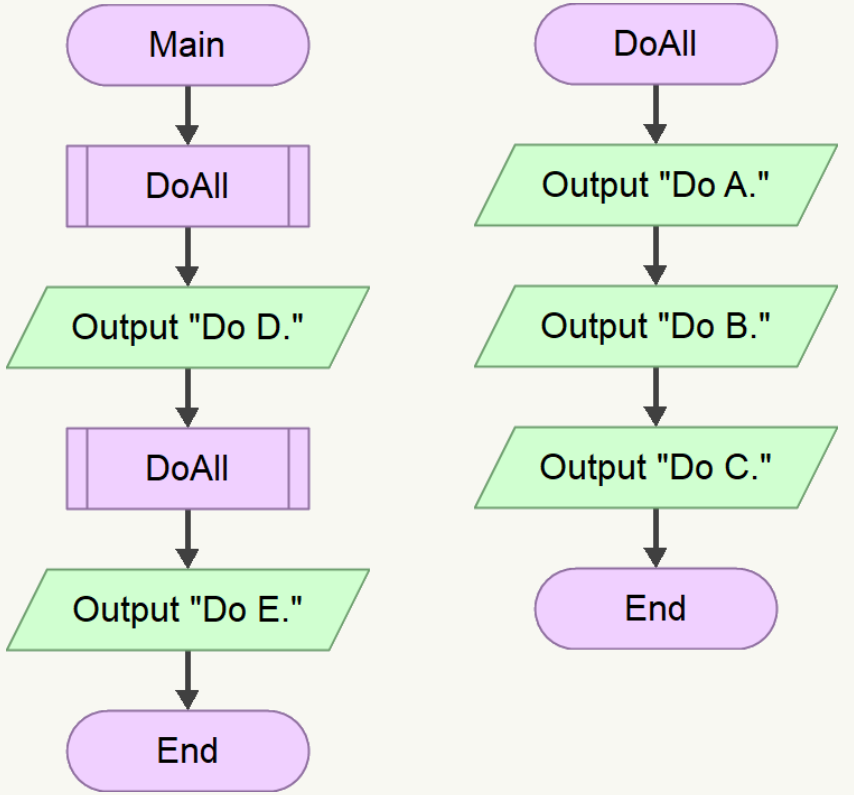


Transparency

- The internals of a function is invisible from the outside.
  - Other parts of the solution do not need to know it.
  - All they need to know is its interface and how to use it.

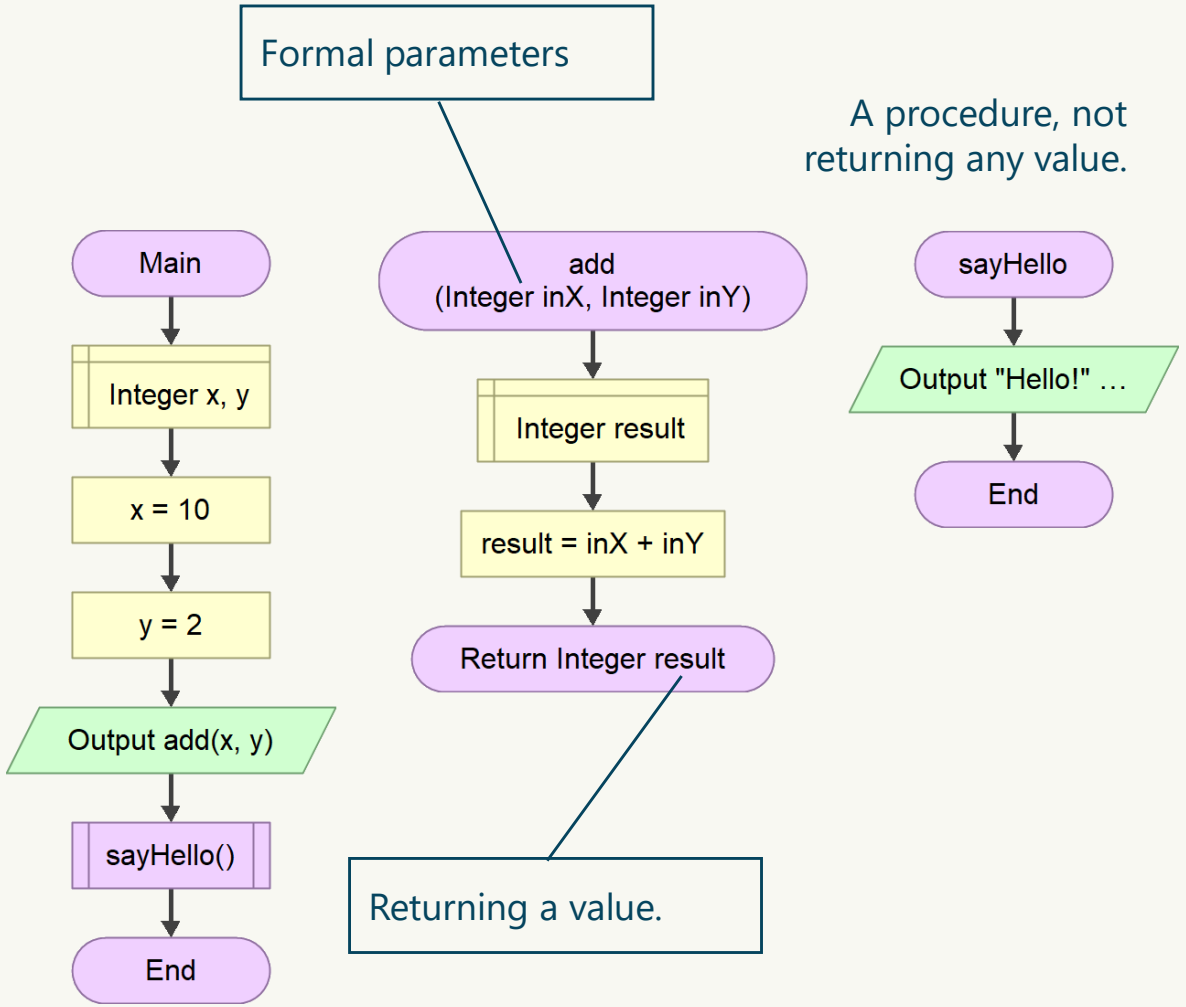
- E.g. the `Main` function has no idea what instructions are inside the `DoAll` function.

A function may have its own data in the form of **local variable** and **formal parameter**.

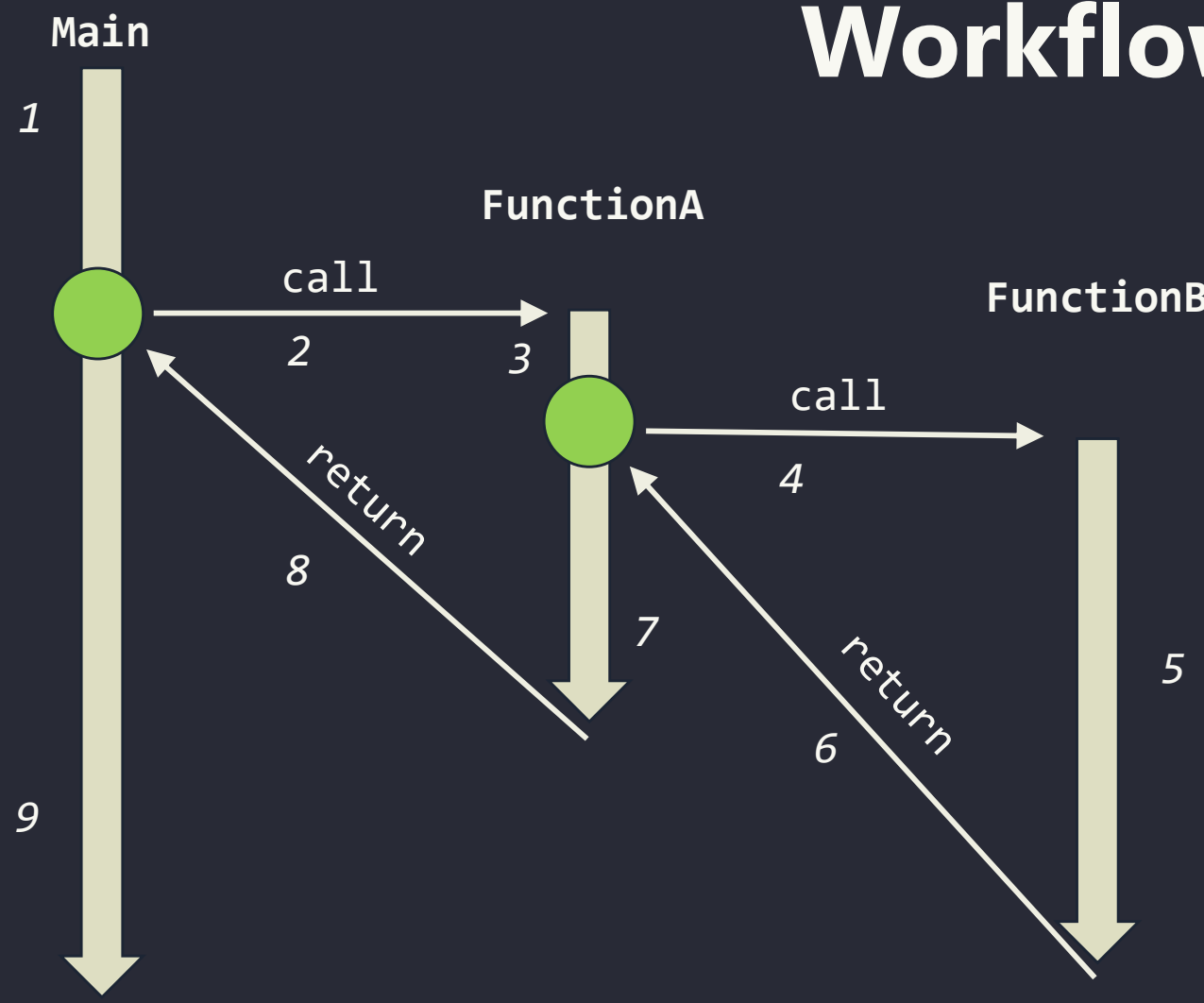


Characteristics

- A function has:
  - An atomic purpose.
  - A unique identifier.
  - A body (instructions).
  - *A set of parameters (optional).*
  - *A returned value (optional).*



Workflow



When:
`FunctionA` is called, the execution flow is transferred from the `Main` flow to the `FunctionA`.
The same also applied when `FunctionB` is called from within `FunctionA`.

When:
The execution of `FunctionB` reaches its end, the execution flow is sent back to the corresponding caller, `FunctionA`., along with the returned value (if any).
The same applies when `FunctionA` reaches its end.

Final **Thoughts.**

Conclusion

1. A function is a callable routine with an atomic purpose.
2. The detail of a function is transparent from the external.

References

Wassberg, J. (2020). Computer Programming for Absolute Beginners. Packt.

– E O F –

Course Lecturer

Mario E. S. Simaremare
Institut Teknologi Del

@simaremare
@dasar-pemrograman

Supported by

Kementerian Pendidikan, Kebudayaan,
Riset, dan Teknologi RI

Inovasi Modul Digital 2022