DigiPen
INSTITUTE OF TECHNOLOGY
SINGAPORE

[CS 225] Advanced C/C++

Lecture 1: Introduction to the Course

Sławomir "Swavek" Włodkowski
(Fall 2020)

Agenda

- Overview of the Course
  - Organization
  - Learning Objectives
  - Schedule
  - Grading
- Overview of Programming Paradigms
  - Declarative
  - Imperative

Overview of the Course

- Overview of the Syllabus
- Attendance
- Grading

Overview of Programming Paradigms

A **programming paradigm** is a certain recognized pattern, model, perspective on how we can communicate the logic of programs to computers.

Key families of programming paradigms:

- *Imperative* – listing a sequence of computational steps.
- *Declarative* – listing dependencies of steps, but not explicitly the order of their execution.

__Think__: "I do not need to tell Excel which cell to calculate first in the spreadsheet!"

Functional programming

- Declarative programming paradigm.
- Influenced by Alonzo Church's Lambda Calculus (1936).
- Popular programming languages:
  Haskel, LISP, JavaScript, F#, Closure.
- Many features supported by C++.

Functional programming

- Functions are ***first-class citizens***:
  - Act as values.
  - Can be passed to functions.
  - Can be composed of other functions.
  - Can be returned from functions.
  - Support suitable operators.
  - Can produce results and side-effects.

- Other objects can act as functions (“callable objects”)
  - Lambda expressions,
  - Functors.

Imperative programming

- Influenced by Alan Turing's Turing Machine (1936).
- *Think*: assembler, C.
- Programs are lists of statements and expressions.
- Popular programming languages:
  C, C++, Java, C#...

***Think**: "In C and C++ I write statements and expressions that describe some task."*

Imperative programming

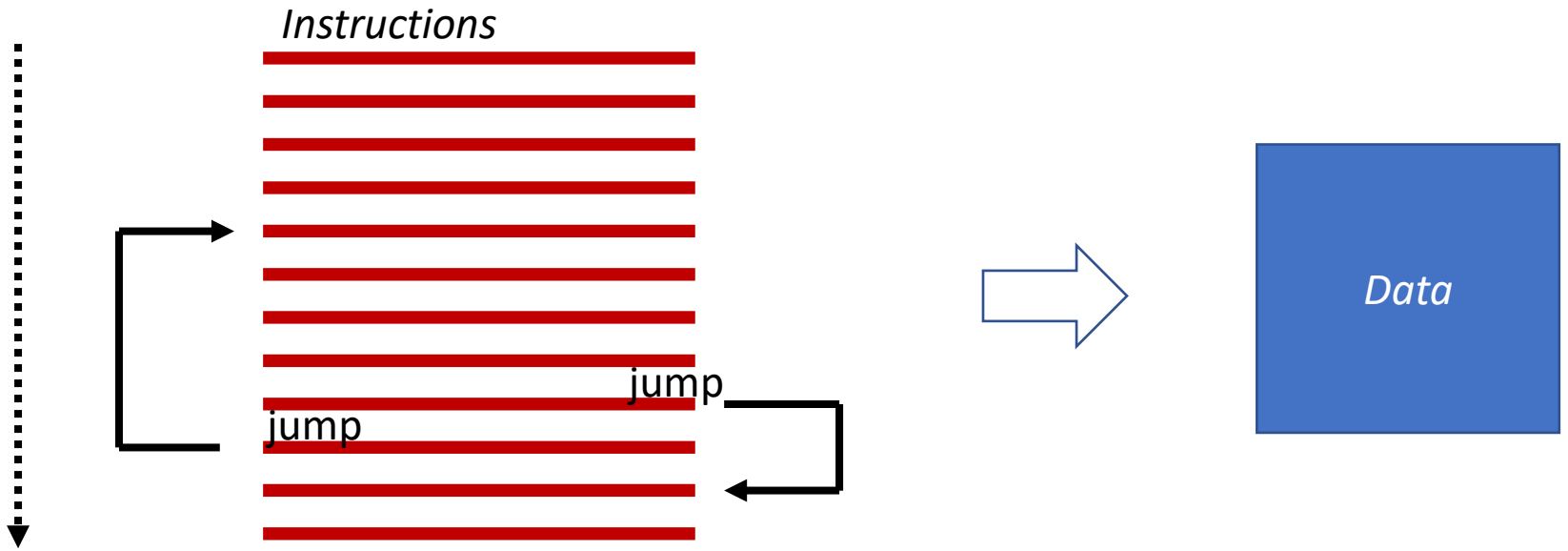
- Programs are sequences of steps or collections of functions that represent sequences of steps.
- Code explains ***how*** to calculate results.
- Code does not have to express ***what*** is calculated.
- Processors only understand *machine code*, which is imperative.

***Think**: "It is hard to figure out how did the program get here."*

Imperative programming

Unstructured programming (e.g. assembly language):

- Program is a single sequence of instructions
- Flow of execution can *jump* to various locations (*goto*)

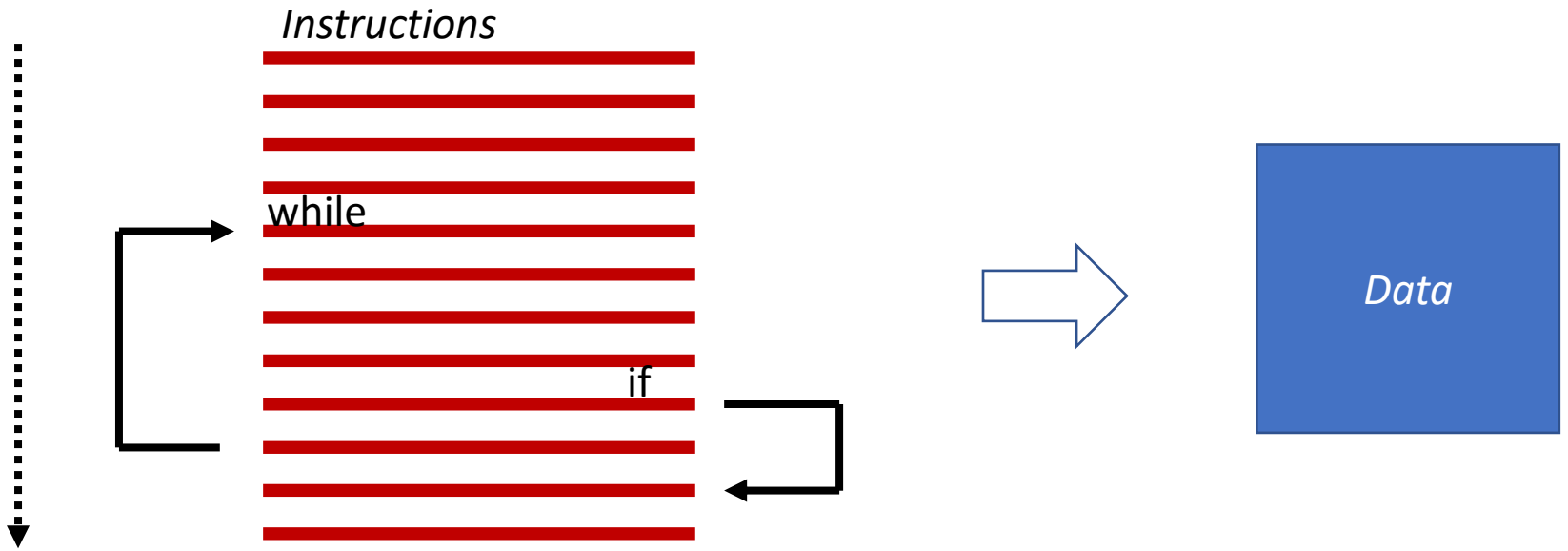


Imperative programming

***Think**: "We introduce the structure to a program to understand the flow better."*

Structured programming (e.g. OS scripts):

- Program is a single sequence of instructions
- Flow of execution is controlled by structured statements (selection, iteration, structured and unstructured jumps)

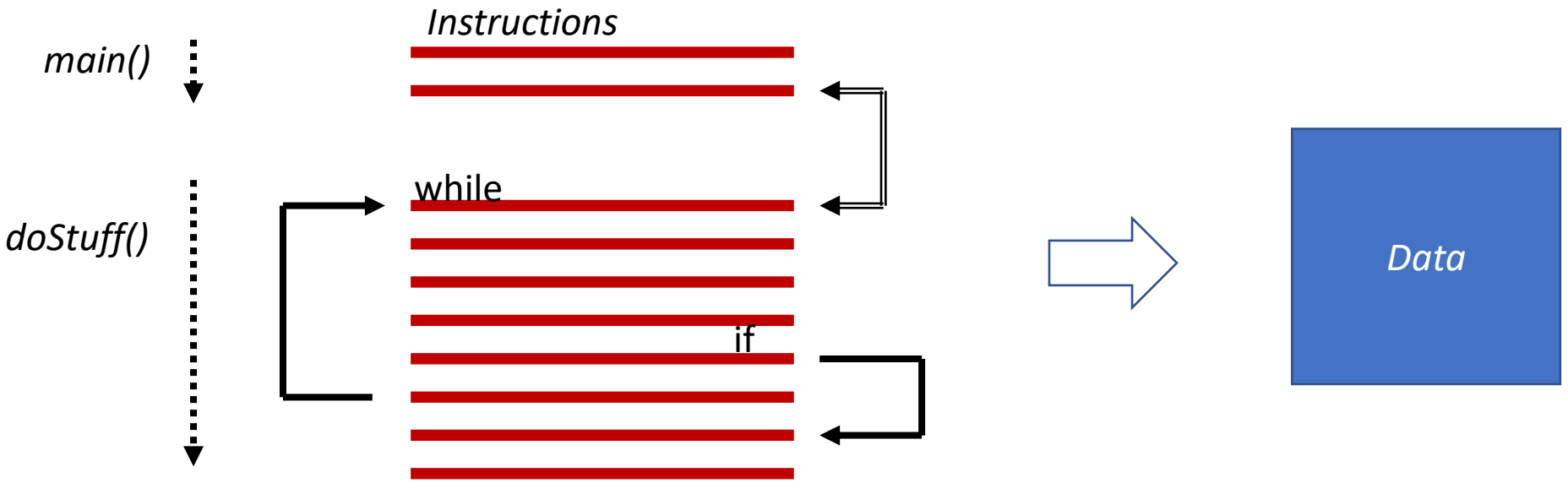


__Think__: "I divide the program into smaller functions to abstract away complexity."

Imperative programming

Procedural programming (e.g. C):

- Program is a collection of procedures (in C terminology: *functions*)
- Flow of execution is controlled by structured statements

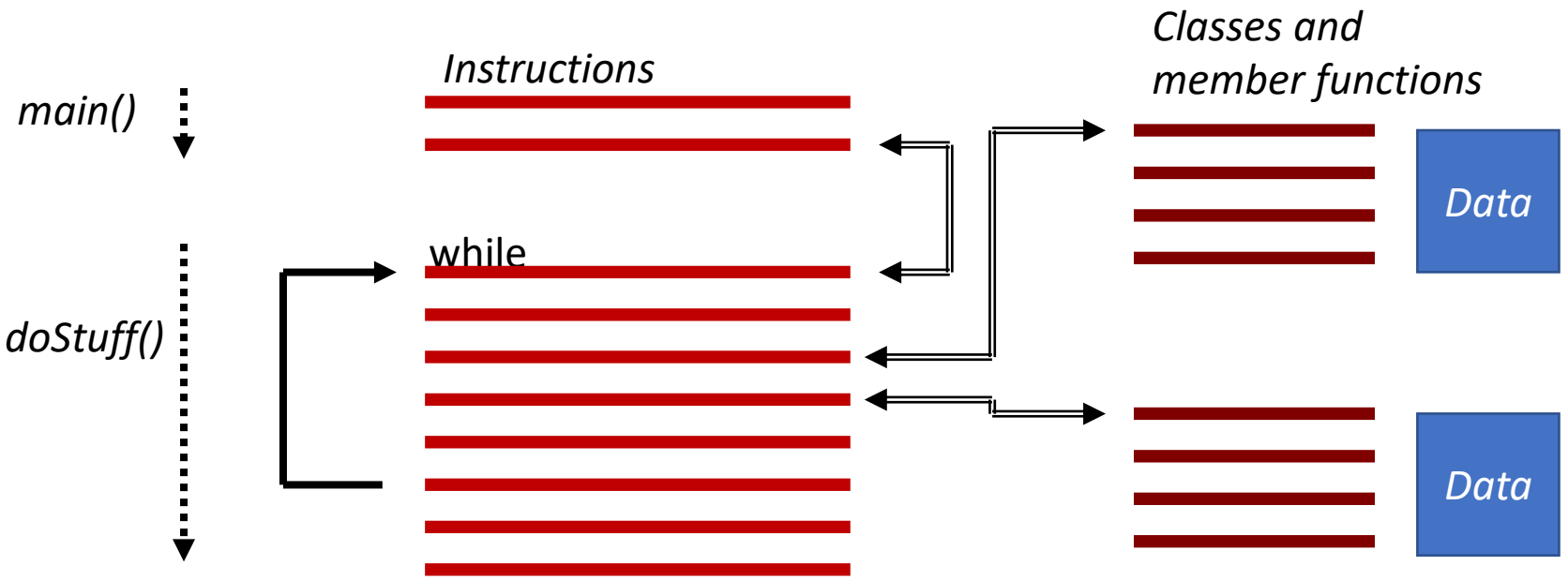


Imperative programming

***Think**: "I encapsulate data and abstract away functions that work on it to control its state modifications."*

Object-oriented programming (e.g. C++):

- Program is a collection of procedures (C++: *functions*) and methods (C++: *member functions*) that work on ***objects*** of ***class*** types.

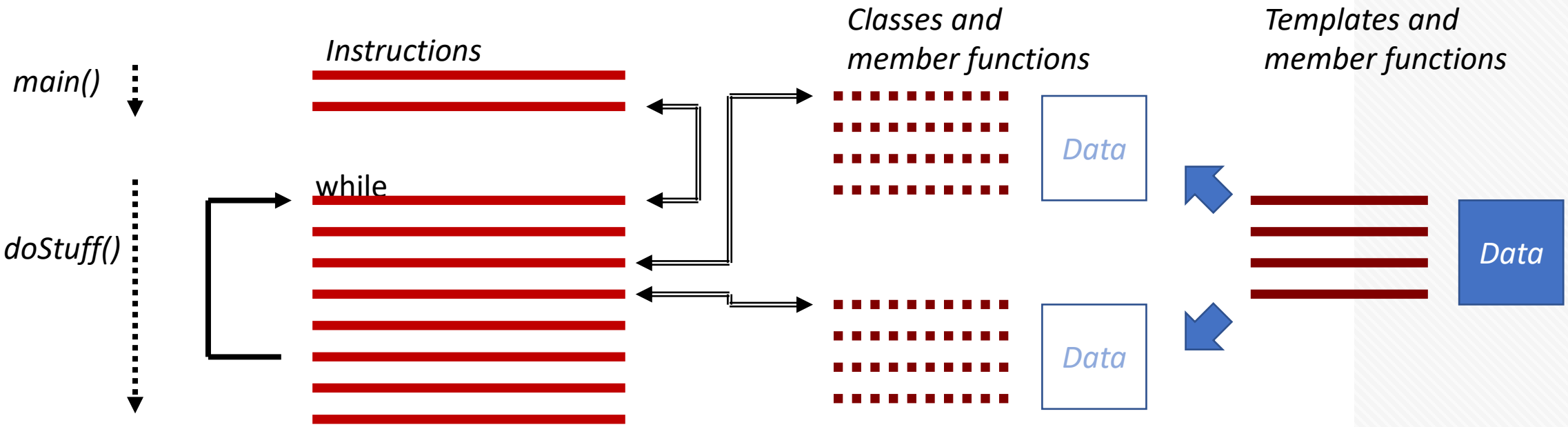


***Think**: "I abstract away classes and other types that only differ in data types they operate on."*

Imperative programming

Generic programming (e.g. C++):

- State of the program is split into separate objects
- Types use static polymorphism for better reuse (*templates*)



Imperative programming

Template meta-programming (e.g. C++):

- Using generic programming techniques to generate temporary source code resulting in new data types, constants and compile-time behaviours.

Plan for this trimester!

- We will dive deeper into:
  - Object-oriented programming
  - Generic programming
  - Template meta-programming
  - Functional programming

- We no longer just want to:
  - Write programs [CS 120] and
  - Use libraries [CS 170], but also
  - Create our own efficient libraries [CS 225].