IPP
Principles of Programming Languages and OOP

Dušan Kolář

kolar@fit.vutbr.cz

+420-54-114-1238

Points

Mid-term exam	20
Project	20
Inclusion	*20*
Final exam	60 (25)
SUM	100 (**45**)

Important Dates

- Mid-term exam: 31st March & 1st April (8th week)
- Project: *two periods*
  - *Will be explained in detail during the next lecture*
- Final Exam
  - Regular: ??.??. 2025
  - 1st retake: ??.??. 2025
  - 2nd retake: ??.??. 2025
  - *Too many students => global decision*

Time Schedule

- Lectures:
  - Day: Monday, **Wednesday**
  - Room: E112 (& others) – Mo, We
  - Time: 08:00 – 10:50 (Mo)
    10:00 – 12:50 (We)

Lecturers

- Dušan Kolář
- Zbyněk Křivka
- Guest (Dominika Regéciová)
- Structuring (supposed)
  - see Moodle pages (IS VUT)
  - may be changed (hopefully not)

Project

- Specification will be released during the 1st week of summer term
  - Unclear formulation and error/typos reports are expected and welcome
  - 2nd week – fix and freeze of the spec.
- 1st task: 18th March 2025 (11th March)
- 2nd task: 23rd April 2025 (16th April)

Students Involvement in Study

- University study
  - Full time job
    - 8h (optimistic, usually 10h ☹) => 40h a week
- 30 credits per term
  - 6 courses, each 5 credits
  - 6x4 hours (50 minutes) = 24x50 = 1200m
  - Which is 20h, that means a half

Study Model

- Official model expectation
  - 20h at home
- Recommended
  - Active participation on lectures
  - Repeat the new stuff at home

Practice

- Probably not the one expected ☺
- Missing prerequisite/s

- Effect
  - Mid-term exam
    - Force studying at least something during term
  - Projects
    - Get the things working

Lecture Streaming

- Original intention:
  - Additional studying resource
- Reality:
  - The only (if any) resource ☹
- Lectures will be online as processed
  - We cannot influence processing
    - Must be both voice and video, otherwise empty
- !! Use additional resources !!

Recommendations

- Solve the project as soon as possible – except this lecture, on organizational topics next, and the guest (Python) there is no other devoted to it
  - **Try to beat the deadline about one week**!
- A feeling you understand a lecture does not imply you can reproduce the content later
  - **We do require full, deep and clear answers**!

Literature/References

- **Reynolds, J.C.: Theories of Programming Languages**
- Sethi, R.: Programming Languages, Concepts and Constructs
- Friedman, D.P., Wand, M., Haynes, C.T.: Essentials of Programming Languages

Literature/References

- Horowitz, E.: Fundamentals of Programming Languages, 2nd edition
- Henderson, P.: Functional Programming, Application and Implementation
- Schmidt, D.A.: The Structure of Typed Programming Languages
- Hruška, T., Beneš, M., Češka, M.: Překladače
- Češka, M., Motyčková, L., Hruška, T.: Vyčíslitelnost a složitost

WWW Resources

- http://www.uml.org
- Manuals and standards (ANSI) of programming languages
- Implementations of programming languages compilers/interpreter
- Lecture notes and supplementary texts for programming languages coming from universities worldwide

Lectures Overview – I

1. Introduction – Terms, PHP
2.-4. Object Oriented P.L. – features, insight, terms
5. Language classification, Non-structured languages
6. Structured languages
7. Modular languages

Lectures Overview – II

8. Design patterns
9. Introduction to Declarative P.L. – Lambda Calculus
10. Functional P.L.
11. Logic P.L.

Lectures Overview – III

12. Other declarative P.L., Main Differences in Use and Implementation Among Imperative and Declarative P.L.
13. Summary, Discussion about Following Courses, etc.

1. Introduction – Terms

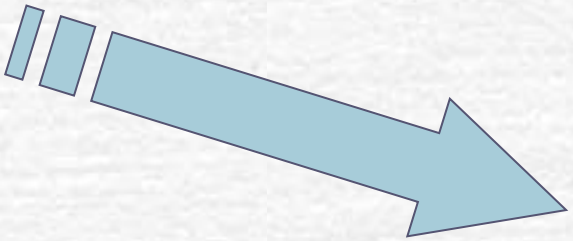
Aims of the Lecture

- Definition of Terms Known and Used in the Area of Programming Languages
  - Language Categories
  - Terms for Characterization
  - Etc.

Why IPP?

- Programming is really hard stuff!
  - Warning!!
    - Schematic work with programming languages is not programming!
    - “Google programming”, “COPILOT programming” is not programming!
- And, moreover: https://x.com/Rainmaker1973/status/1752426357747818695?s=20

Programming Process



Programming Process

- Difficult time & resources consuming task
- Knowledge of P.L. not sufficient
- Requirements specification usually not complete (vague determination, etc.)
- Under/Over-estimation
- Tight time schedules

and many others…

What a Programming Language Is?

- Programming language is an *intermediary mean* between common speech and sequence of usually binary digits

What a Programming Language Is? – II

- A *finite set of commands* of a specific syntactic form with strictly defined semantics

Why Programming Languages

- Natural languages are not suitable
  - Too complex analysis (none, so far)
  - Ambiguous, exact description too difficult
  - Etc.
- Binary code of target architecture is
  - Too complex to remember
  - Not suitable for more complex problems
  - Not usable for *fast* programming

What a Computer Program Is?

- Computer program is an abstraction of reality
  - It implements abstract models
  - CPU abstraction – model is implemented over it

Why We Need to Understand

- Make our programs better!
- Not sufficient to guess and test a programs does what we want..
- We must know why!
  Why not another way!

A Bit of History

- Early Fifties of 20th century
  - Symbolic names only (memory cell, operation)
  - Subroutines (macros)
    - Names for groups of actions, separation of views
    - Subroutine **implementation**
      - What is the realization
    - Subroutine **invocation**
      - What it does (call/interface)

History – cont.

- Early Sixties of 20th century
  - Level of abstraction increases
  - Programming comfort increases
    - Data abstraction
      - Data model used within a program
    - Control abstraction
      - Computational model, combination of elementary actions

History – cont.

- Seventies of 20th century
  - Simplicity, abstraction on higher level, research, broader application
    - Algol-W
    - Pascal
    - C
    - CLU
    - Euclid
    - Mesa

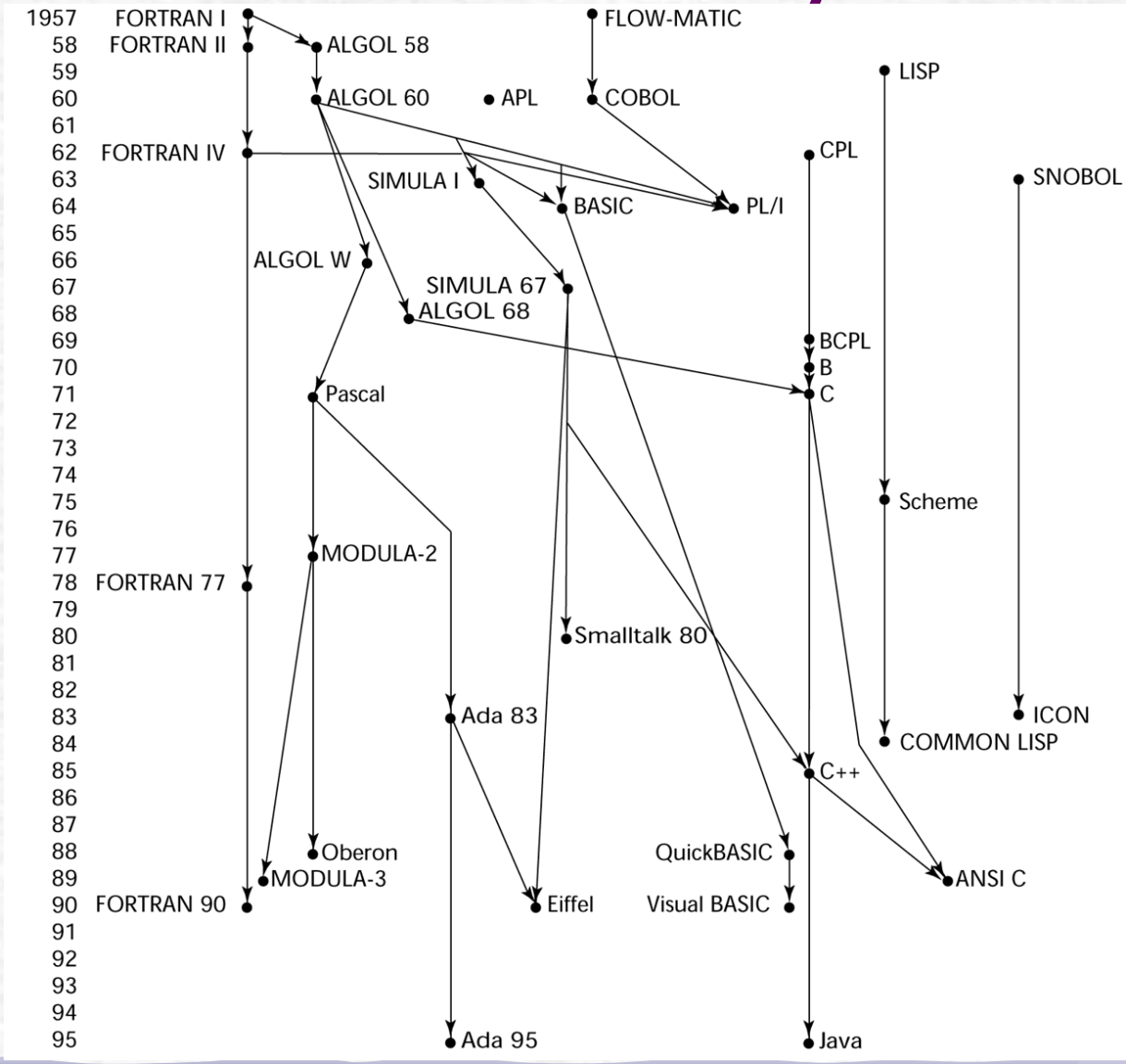
History – cont.

- Eighties of 20th century
  - New directions, modularity, object orientation
    - Ada
    - Modula-2
    - Smalltalk
    - C++
    - Scheme
    - ML
    - Miranda
    - FP
    - Prolog

History – cont.

- Nineties of 20th century
  - Internet, libraries, scripting languages, etc.
    - Java
    - Haskell
    - AWK
    - Perl
    - Tcl
    - Javascript
    - Python
    - Visual Basic

Summary


1957
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
FORTRAN I
FORTRAN II
ALGOL 58
ALGOL 60
APL
FLOW-MATIC
COBOL
LISP
FORTRAN IV
CPL
SNOBOL
SIMULA I
BASIC
PL/I
ALGOL W
SIMULA 67
ALGOL 68
BCPL
B
C
Pascal
Scheme
MODULA-2
FORTRAN 77
Smalltalk 80
Ada 83
ICON
COMMON LISP
C++
Oberon
QuickBASIC
MODULA-3
ANSI C
FORTRAN 90
Eiffel
Visual BASIC
Ada 95
Java

Future ?

- Another new technologies
- Area of computer science is extremely dynamic, thus, it is hard to predict
- Nevertheless, there is enough space for new ideas

Term Simplification

- Program = computer program
- Language, P.L. = programming, or another computer-processed language for description of input data transformation to output data

Programming languages specification

- Syntax
  - Defines program structure
    - Grammars, formal languages (IFJ)
- Semantics
  - Description/definition of meaning of individual constructs, way of evaluation, etc.

Syntax (summary)

- 3 levels of description
  - Lexical
  - Context-free
  - Context bindings and limitations
- Methods of description
  - *Natural languages*
  - Syntactic graphs
  - BNF, EBNF
  - Grammars

Semantics

- Static
  - Describes features, that can be studied during translation/compilation
    - Type compatibility, etc.
- Dynamic
  - Describes features, that can be detected and investigated during program evaluation (run)
    - Index size, etc.

Formal Semantics Definition

- Axiomatic semantics
  - Defines axioms, that hold for individual constructs
- Operational semantics
  - Defines program behavior as a sequence of state transitions
- Denotational semantics
  - Defines program as a function mapping input values to output ones

Declaration X Definition

- Declaration
  - Just attributes
  - Explicit X implicit
- Definition
  - Variables
    - Attributes, a way of allocation
  - Function
    - Attributes, function body

Binding

- Connects entity with its feature/attribute
- Time of binding
  - During language design
  - During program implementation
  - During program translation/compilation
  - During program linkage
  - During program load
  - During program runtime/evaluation

A Kind of Binding

- Static
- Dynamic
  - It is created and/or it can be changed during program run/evaluation

Example – variable

- `int count; ... ; count = count + 5;`
  - Set of types of variable count
  - Type of variable count
  - Set of values of variable count
  - Value of variable count
  - Set of meanings of symbol +
  - Meaning of symbol + in this case
  - Internal representation of literal 5

Variable Features

- Name
- Address and location in memory space
- Values
- Type
- Life time
- Scope (rozsah platnosti)

Variable Name

- Name length
  - Maximal X effective
- Character set
  - Letters of (US) alphabet
  - Special characters
- Case sensitivity
- Key words X reserved words

Location, Address, and Value of Variable

- L = R
  - Variable on the left-hand side is *location+address*
    - L-value
  - Variable on the right-hand side is a value
    - R-value
- Name can be connected with various locations and addresses
  - Local variables in different subroutines
  - Local variables in recursive subroutines
  - Global constant variables
- Address can be connected with various names
  - Pointers, parameter passing by reference
  - Advanced optimization

Location, Address, and Value of Variable

- Static binding
  - Location, Address – static variables
  - Value – constants
- Dynamic binding
  - Variables located on the stack
  - Variables located on the heap
    - Allocation
    - De-allocation – implicit X explicit

Type of a Variable

- Denotes
  - Set of possible values
  - Set of possible operations
- Static binding
  - Majority of *classic* languages (declaration)
  - Implicit declaration (Fortran, Basic)
- Dynamic binding
  - Scripting and interpretative languages
  - Dynamic type checking – time consuming

Type of a Variable

- Example of dynamic binding
  - `if ( $p ) $x = 10; else $x = „abc“;`
    `echo $x+1;  // 11 or „abc1“`
- Languages
  - Type-less
  - Non-typed
  - Typed
    - Explicit typing X type inference (odvození)

Scope of a Variable

- Part of a program, in which it is possible to work with the entity
- Variable visibility
  - Variable hiding
- Static binding
  - Denoted by program structure, majority of languages
- Dynamic binding
  - Till next declaration (APL, SNOBOL4, *LISP*)
  - Simple in interpreted languages, not suitable from SE viewpoint

Lifetime of a Variable

- Interval, when there is a memory allocated for the variable
- Static allocation
  - Before program run/evaluation
- Dynamic allocation
  - Automatic
  - Command for creation/allocation

Another Terms

- Dynamic programming languages
  - usually non-typed – dynamic typing
  - features through whole spectrum of paradigms, but not necessarily
    - Non-structured
    - Structured
    - Modular
    - OO

Terms to Remember

- Syntax, semantics
- Binding
  - Static, dynamic
- Abstraction
- Declaration X definition

Exercises/Motivation

- Examine taught terms within your favorite languages:
  - C, C++, C#, Java, JavaScript, Perl, Python, ML, Haskell, Hope, LISP, PHP, Pascal, Object-Pascal, Objective C, SmallTalk, Prolog