General programming knowledge
Programming paradigm
Programming languages

Course: Object Oriented Programming (OOP)
CTU, FS, U12110
Matouš Cejnek

What is object oriented programming?

Programming language paradigm

Programming paradigms are a way to classify **programming languages** based on their features.

Languages can be classified into multiple paradigms.

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

What is paradigm?

In science and philosophy, a paradigm is a distinct set of concepts or thought patterns, including theories, research methods, postulates, and standards for what constitute legitimate contributions to a field.

(source: Wikipedia)

Programming language

A programming language is any set of rules that converts strings, or graphical program elements in the case of visual programming languages, to various kinds of machine code output.

There are many of them:
https://en.wikipedia.org/wiki/List_of_programming_languages

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Programming paradigms

- *Imperative* (instructions how to change the machine state)
  - **Procedural**
  - **Object-oriented**
- *Declarative* (definition of the desired result)
  - **Functional**
  - Logic
  - Mathematical
  - Reactive

Imperative paradigms (popular languages)

- Fortran
- Java
- Pascal
- ALGOL
- C
- C#
- C++
- Assembler
- BASIC
- COBOL
- Python
- Ruby

Declarative paradigms (popular languages)

- Prolog
- Lisp
- Haskell
- Miranda
- Erlang
- *SQL (not completely)*

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Interesting paradigms

Simplified classification (used for programming language comparison):

- Procedural programming
- Functional programming
- Object-oriented programming (OOP)

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Procedural programming paradigm

1. **Sequence of commands**: Programs is a series of instructions executed in order.
2. **State changes** / **Use of variables**
3. **Control structures**: Constructs like loops and conditionals control the program flow.
4. **Emphasis on algorithms**: Focuses on the step-by-step process to solve a problem.

Procedural programming paradigm

Python

```
A = 1
B = 2
c = A + B
print(c)
```

BASIC

```
0 let A = 1
1 let B = 2
3 let C = A + B
4 print C
```

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Functional programming paradigm

1. **Function-based:** functions take inputs and return outputs
2. **Immutability:** variables are not changed after they are set
3. **No state changes:** instead relying on returning new values.
4. **Emphasis on expressions**: every function returning a value.

Functions return values instead of modifying them

Object oriented paradigm

It will be introduced later.

To remember:

- OOP is programming language paradigm.
- OOP is the most widely used paradigm for designing applications and software

Review of basic programming terminology

Function, Algorithm, Computer Program

- **Mathematical function** is formal notion of a mapping from inputs to outputs
- **Algorithm** - general instructions how to achieve result.
- **Computer program** - implementation of algorithm(s), instructions for specific device.

Algorithm

It can be understood as:

- List of instructions
- Something between mathematical function and computer program
- Can be represented in many forms:
  - Natural language
  - Diagram (for example flow chart)
  - Pseudo code

Computer program

A program that is executable by device (computer). Can be written in any supported language.

Source code - Human-readable form of computer program

Machine instructions - source code translated to executable form

Interpreter vs Compiler

- Interpreter and Compiler translate a source-code to machine instructions.
- Interpreter is a translator that directly translates and executes the code line by line (Python, Ruby, etc.).
- Compiler is a translator that converts (compile) the whole code before the execution (C, C++)

Variable / reference

Computer programming variable is less abstract than variable in mathematics.

In computer programming, a **variable** is a storage location paired with an associated symbolic name, which contains some known or unknown quantity of information referred to as a value.

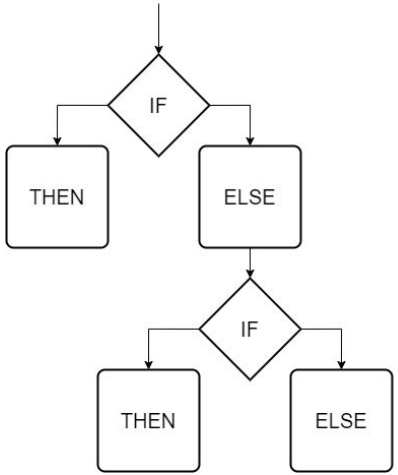
Variable vs reference - it is *programming language specific issue*.

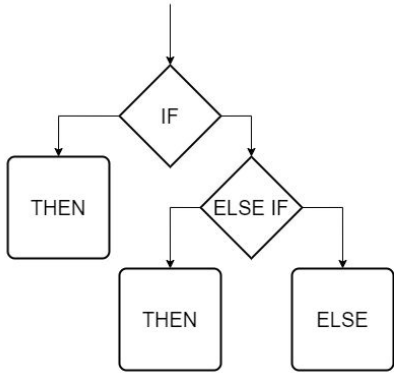
Common operations

- The most common: `+`, `-`, `*`
- Division (`/`) is often only for integers
- Modulo (`%`)
- Assignment (`=`, or `:=`)
- Comparison `==`, `>=`, `<=`, `!=`, `===`

Conditional statement

Generally called IF THEN condition.The syntax can vary across languages (including variants of else if etc.).





Conditional statement in examples

Python

```
if value == 1:
    print(“Yes”)
elif value == 2:
    print(“Almost no”)
else:
    print(“No!”)
```

Javascript

```
if (value === 1) {
  console.log(“Yes”)
} else if (values === 2) {
  console.log(“Almost no”)
} else {
  console.log(“No!”)
}
```

Goto (jump)

- Jump to a specific line or tag
- Statement for one-way transfer in program flow
- Popular mainly before 1980

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Loops (for, while)

A loop is a sequence of instruction s that is continually repeated until a certain condition is reached.

- FOR loop - condition is given by number of iterations
- WHILE loop - condition is tested in every iteration, beware of infinite loop

For loop (iteration over range)

Python

```
for i in range(1, 11):
    print(i)
```

C

```
int i;
  for (i = 1; i < 11; ++i)
  {
    printf("%d ", i);
  }
```

For loop (iteration over container)

Python

```
items = [1, 20, 33, -5]
for item in items:
    print(item)
```

Javascript

```
items = [1, 20, 33, -5]
for (let i = 0; i < items.length; i++) {
  console.log(items[i])
}
```

While loop

Python

```
i = 0
while i < 10:
    print(i)
    i += 1
```

Javascript

```
let i = 0;
while (i < 10) {
    console.log(i)
    ++i
}
```

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Practical example

Example: Collatz conjecture

Algorithm description with mathematical functions:

Consider the following operation on an arbitrary positive integer:

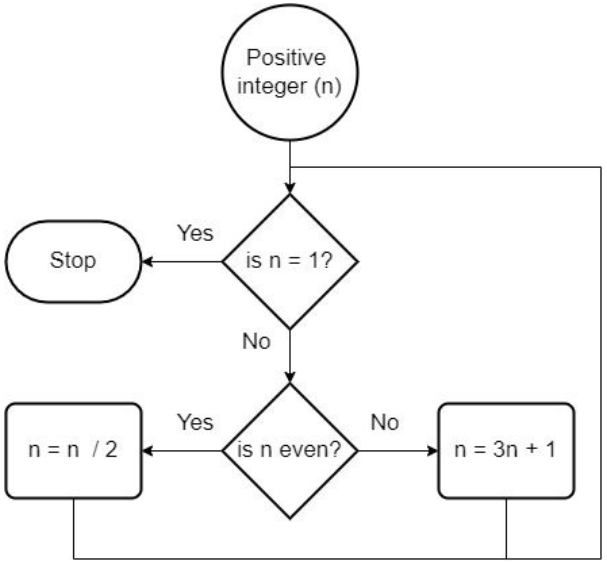
$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \equiv 0 \pmod 2 \\ 3n+1 & \text{if } n \equiv 1 \pmod 2 \end{cases}$$

Now form a sequence by performing this operation repeatedly, beginning with any positive integer, and taking the result at each step as the input at the next.

$$a_i = \begin{cases} n & \text{for } i = 0 \\ f(a_{i-1}) & \text{for } i > 0 \end{cases}$$

Example: Collatz conjecture

Diagram as a tool
to display an algorithm:



Example: Collatz conjecture

One of the possible solutions with Python:

```
num = < USER INPUT INTEGER VALUE >
while num != 1:
    print(num)
    if num % 2 == 0:
        num = num / 2
    else:
        num = 3 * num + 1

print(num)
```

Brief review of programming languages

What is the best programming language?

What is the best language?

This is problem specific question.

See *no free lunch theorem* for better explanation.

(https://en.wikipedia.org/wiki/No_free_lunch_theorem)

It would be a shame to not use the right tool for the right job!

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Some attributes of programming languages

Writability / Readability - how easy is creating and editing of a code?

Efficient execution / translation - computer time cost

Reliability / Security - is the language predictable and secure?

Implementability / Portability - what target devices are available?

Maintainability - is the development active, updated, maintained?

Programming languages: Writability / Readability

Keep in mind, the resulting readability is not only result of language quality, but also coder effort plays significant role.

Example of FizzBuzz solution (both solutions are not optimized)

Python

```
for fizzbuzz in range(N):
    if fizzbuzz % 3 == 0 and fizzbuzz % 5 == 0:
        print("fizzbuzz")
        continue
    elif fizzbuzz % 3 == 0:
        print("fizz")
        continue
    elif fizzbuzz % 5 == 0:
        print("buzz")
        continue
    print(fizzbuzz)
```

Brainfuck

```
+[-[>+<<]>-]>--[>+>++>++>++++++>+>>>++++++<<<[<]>-
]<+++++[>+>+>->>->++>+>>>--[<<]>-]>[>]<--<->>>[<<-[
<]<[>+++>+<<.<.<..[<]<]>>-[<<]>[<+++++>.>.>..[>>>]
<<+[<]]>[>>>]<<[[-]>>]>[>++++++++++[->++++++++++>+
<<]<[->+>-<<]>>++[>[>>>]<[>++++++++++>+>]<<-<-]+++
++++++>[-<->]<[->+<]>>[>]++++++++[-<++++++<++++++>
>]<<<[>[-]]>[>]<[.[-]<]<[-<+>]>]++++++++++.[-]<<-]
```

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Programming languages: Efficient execution / translation

- **compile time** - translation into machine instructions
- **load time** - loading into machine memory
- **execution time** - actual run

Keep in mind:

- Program can be written in suboptimal way (computer time vs human time)
- Different hardware can struggle with different tasks
- Higher languages can exploit speed of lower languages for specific tasks

Programming languages: Reliability / Security / Maintainability

- Programming languages (translators etc.) also contain vulnerabilities, weaknesses and bugs.
- If languages and their tools are not updated, they are obsolete (security point of view).
- Implementation of a new version of language can cause problems with already running applications.

Programming languages: Reliability / Security / Maintainability

Keep in mind:

Projects in IT development are never done.
They are under development, or obsolete.

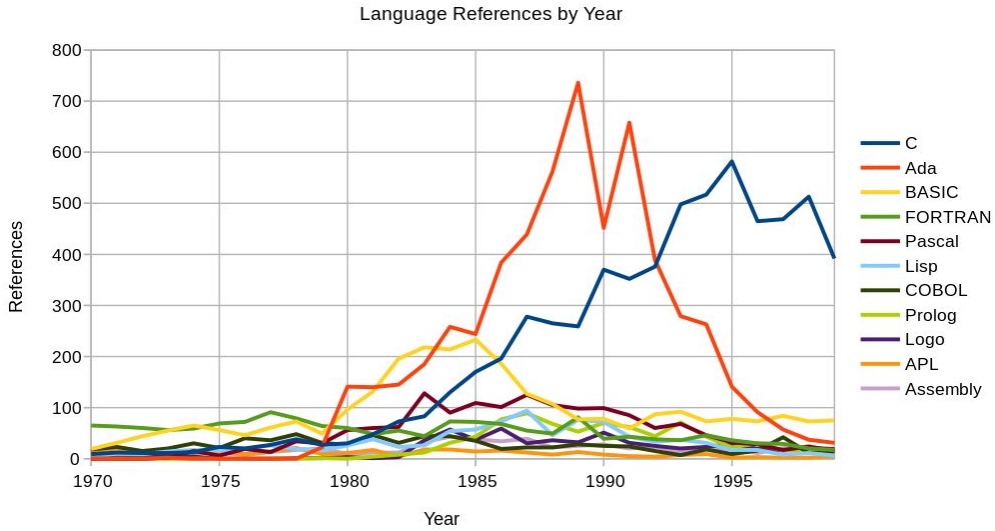
Language usage and popularity

IEEE spectrum:

https://spectrum.ieee.org/top-programming-languages-2024

Stackoverflow survey:
https://survey.stackoverflow.co/2024/technology

Language popularity is time varying



Source: https://retrocomputing.stackexchange.com/questions/11289/is-there-data-for-programming-language-popularity-pre-year-2000

Bonus content

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Some interesting (but not practical) examples

https://codegolf.stackexchange.com/questions/18541/period-of-the-decimal-representation

https://codegolf.stackexchange.com/questions/247192/bitwise-xor-of-floats

Message of examples:

- Shorter is not necessarily simpler or better
- There is no universally best programming language