Programming Languages & Paradigms

Jacek Wasilewski

Overview

- Programming Languages
- Evolution of Programming Languages
- Paradigms
  - Imperative
  - Object-Oriented
  - Functional
- Creators of Programming Languages

Programming Language

- **Programming Language** is a notation that gives us a possibility of telling a computer what to do.
- Like English has words, symbols and rules, Programming Language also has words symbols and grammatical rules called **syntax**.

Why?

- There are plenty of programming languages: Ada, ALGOL 60, Assembler, C, C++, C#, Fortran, Java, JavaScript, Pascal, Perl, PHP, Python, Ruby, Scala, SQL…
- The question is – why?
  - 1950s – present
  - Different level of abstraction
  - Math, grid, web, general purpose language…

Evolution (1)

- First generation - machine language:
  - Programming using 0s and 1s
  - Operation code – arithmetic operations
  - Operands – data to be processed
  - Architecture dependent
  - Optimised but difficult and prone to errors
- Example:
  `00000001` Turn bulb fully on
  `00000010` Turn bulb fully off
  `00000100` Brighten bulb by 10%

Evolution (2)

- Second generation - assembly language:
  - Codes replace binary operation codes
  - Assembly language code is translated into machine language code
  - Optimised as well but still rather difficult
- Example:

```
Assembly        Machine code
SUB AX, BX      001010111000011
MOV CX, AX      100010111001000
MOV DX, 0       10111010000000000000000
```

Evolution (3)

- Third generation:
  - High-level programming languages – closer to human languages
  - Source code must be translated into machine code.
    - translation is done before the execution – **compilation**
    - translation is done during the execution – **interpretation**
  - Platform independent
- Example:

```
#include <stdio.h>
void main() {
    printf("Hello world!");
}
```

Evaluation (4)

- Fourth generation:
  - 3GL where the same can be done using fewer instructions
- Example:

```
SELECT name, salary
FROM employees
WHERE age > 20
ORDER BY name;
```

Evaluation (5)

- Fifth generation:
  - Programming language focused on solving problems using constraints rather algorithms written by a programmer
    - Also graphical programming where we drag and drop
  - Declarative languages
  - Functional languages
  - Logic languages

Programming Paradigm

- Paradigm – a theory or a group of ideas about how something should be done, made, or thought about (Merriam-Webster)
- Programming Paradigm – fundamental style of computer programming, serving as a way of building the structure and elements of computer programs (Wikipedia)

Paradigms

- Imperative Paradigm
- Object-Oriented Paradigm
- Functional Paradigm
- Logic Paradigm
- Declarative Paradigm

Imperative Paradigm (1)

- **Idea:** Do things step by step
- **More formally:** paradigm that uses statements to change a program's state / data fields
- **Examples:** C, C++, C#, Java, PHP, Python, …

Imperative Paradigm (2)

- Example:

```
    result = []
    i = 0
start:
    numPeople = length(people)
    if i >= numPeople goto end
    p = people[i]
    nameLength = length(p.name)
    if nameLength <= 5 goto next
    upperName = toUpper(p.name)
    addToList(result, upperName)
next:
    i = i + 1
    goto start
end:
    return sort(result)
```

Source: http://cs.lmu.edu/~ray/notes/paradigms/

Stepwise Refinement (1)

- **Stepwise refinement** is a process of programming in which we start from an idea to finished, refined, code.
- Sometimes it is called **top-down** design.
- Stepwise design is essentially about breaking down a system into sub-systems.

Stepwise Refinement (2)

- Our task: Make some pancakes.
- “Make some pancakes” is not that simple so we have to break it down into simpler tasks.
- We could break it down into:
  - Organize kitchen.
  - Make pancakes.
  - Serve pancakes.

Stepwise Refinement (3)

- Our simpler tasks are still too complex so we have to split them:
  - Organize kitchen.
    - Clean surfaces.
    - Get out mixing bowl, whisk, spoon, sieve.
    - Get out plain flour, salt, eggs, milk, butter.
    - Put on apron.
  - Make pancakes.
    - Sift salt and flour into bowl.
    - Break eggs into bowl.
    - …
    - Cook.
  - Serve pancakes.

Stepwise Refinement (4)

- If needed, tasks can be split into even smaller:
  - Cook:
    - Get pan to temperature.
    - Pour batter in.
    - Spread batter to edges.
    - Use plastic spatula to check bottom of pancake.
    - When brown, flip.
    - User plastic spatula to check bottom of pancake.
    - When brown, finish.

Object-Oriented Paradigm (1)

- **Idea:** Model the real world, models communicate
- **More formally:** “objects” contain data (attributes) and code defining behaviour (methods)
- **Examples:** C++, C#, Java, Ruby, Common Lisp, Scala, …

Object-Oriented Paradigm (2)

- Example:

```
result = []
for p in people {
    if p.name.length > 5 {
        result.add(p.name.toUpper);
    }
}
return result.sort;
```

Source: http://cs.lmu.edu/~ray/notes/paradigms/

Functional Paradigm (1)

- **Idea:** everything is an expression, put and chain expressions together
- **More formally:** treats computation as the evaluation of mathematical functions and avoids changing state and mutable data
- **Examples:** Clojure, Scala, Haskell, Lisp, Erlang, …

Functional Paradigm (2)

- Example:

```
sort(
    filter((s => s.length() > 5),
        map((p => p.name.to_upper()), people)
```

Source: http://cs.lmu.edu/~ray/notes/paradigms/

Programming Languages – Once more

- Lisp (1958) – F
- COBOL (1959) – I, O
- ALGOL 60 (1960) – I
- Pascal (1970) – I
- Prolog (1972) – L
- C (1973) – I
- SQL (1978) – D
- C++ (1980) – I, O
- Ada (1983) – I, O
- Erlang (1986) – F
- Mathematica (1988) – F, L
- Haskell (1990) – F
- Ruby (1993) – I, F, O
- Java (1995) – I, F, O
- Delphi (1995) – I, O
- C# (2000) – I, F, O
- Scala (2003) – F, O
- Go (2009) – I

Few names… (1)

- Alan Turing – Universal Turing Machine, ACM Turing Award
- John von Neumann – von Neumann Architecture, functional analysis,
- John Backus – FORTRAN, BNF
- John McCarthy – Lisp, AI
- Niklaus Wirth – Pascal and other languages

Few names… (2)

- Edsger Dijkstra – structured programming, “A Case against the GO TO Statement”, shortest path algorithm
- Tony Hoare – quicksort, Hoare logic for correctness
- Donald Knuth – analysis of algorithms, “The Art of Computer Programming”
- Bjarne Stroustrup – C++

Summary

- Programming language – notation of writing computer programs
- 5 generations of programming languages
- Programming paradigm – style of computer programming
- Many programming paradigms exist – we use them for different purposes