source codes included

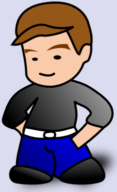
Software paradigms

and a tiny introduction to ASM

Lukas Prokop—viktring12 talk

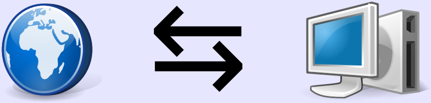
Lukas Prokop :: meisterluk

- BRG Viktring matura 2009
- developer of BRGproj
- I ♥ software
- CS student at TU Graz
- web, typesetting, jazz



The discrepancy

- Houston, we have a problem!
- The real world does not fit to the computer's memory



The discrepancy

economy, math, chemistry, geology, ... objects, dependencies, security,

Software paradigms

- Different approaches to recurring software problems
- "Software" or "programming" paradigms
- low-level (machine-dependent)

  TO

  high-level (real-world-alike)

Assembler—ASM

- primitive instructions
- imperative style
- arithmetic, transfer, control
- x86 → {amd64, x86_64}
- PowerPC, ARM, ...

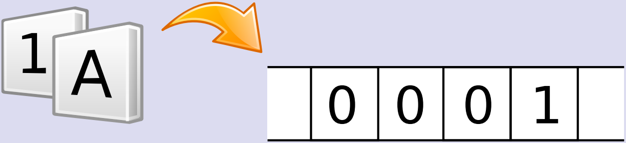
TOY ; educational processor

- instruction set
- registers
- memory

look at the docs

Concept #1: assignment

- <variable> := <value>
- memory stores binary values
    - 1 = 1 = 0b0000001
    - "A" = 65 (ASCII) = 0b1000001



Concept #2: iteration

Do the loop dance!

- Do things again and again ↺
- (in)finite times ∞
- 'i' is our iterator ○

```
10  i = 1
11  # code block
12  i = i + 1
13  if i < 5, GOTO 11
```

Assembler

conclusion

- ASM is primarily generated
- highest performance
- one error can lead to serious failures
- we don't want to write apps
- we need abstraction

Concept #3: abstraction

- we take all the details
- construct a general model
- implement this model
- ... and get a new layer
- therefore, we hide complexity
- ie. "generalization"

Structured programming

- model of SP:
  sequence-selection-repetition
- we get more structure
- more readability
- we use data types
- eg. C, pascal

Iteration in C

```
int i;
for (i=1; i<5; i++)
{
  printf("Hello World ");
}
```

Iteration in pascal

```
VAR i : Integer;

for i := 1 TO 5
DO BEGIN
  Write('Hello World ');
END
```

Concept #4: Records

called "struct" in C

- a car has several properties
  - color, id, #wheels

```
Car = RECORD
  id : String;
  wheels : Integer;
  color : RgbColor
END;
```

Structured programming

conclusion

- We can structure data
- nice abstraction
- still good performance
- operating systems, graphics, ...

Object-oriented programming

- objects : {behavior, attributes}
- classes are templates of objects
- concepts:
  - instantiation
  - inheritance
  - polymorphism
  - encapsulation

Concept #5: classes

```
class Car
{
  var wheels:Int = 4;
  var id:String = "K viktring12";
  var color = Red;
  var speed = 0;

  def accelerate() { speed += 10 }
  def stop() { speed = 0 }
}

val c = new Car();
c.accelerate();
c.accelerate();
c.stop();
```

Object-oriented programming

conclusion

- Very good encapsulation of state
- Design patterns introduced
- heavily introduced by industry
- Under academic research
- websites, desktop app, ...
- eg. C++, ObjC, Java, C#, *

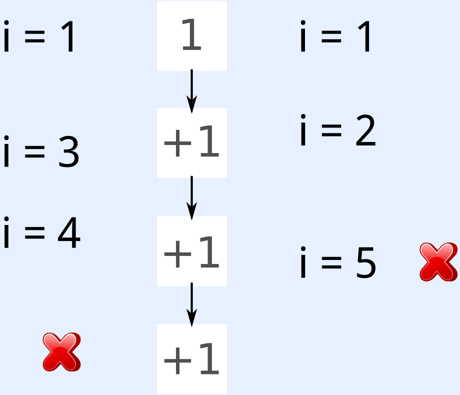
Functional programming

impossible and interesting

- state is the devil of concurrency
- we need to abolish state
- ⇒ concurrency boost
- functions as central element
- eg. Haskell, Dylan, Clojure

Synchronization problem

shared memory



i = 1

1

i = 1

i = 2

i = 3

+1

i = 4

+1

i = 5 ✗

✗

+1

Concept #6: immutability

- all operations applied on data (variables, collections, objects) do not change the data itself.
- data is immutable.
- no synchronization problem, but no iterators!

Concept #7: recursion

```
(defn hello [i]
  (if (< i 5)
    (hello (+ i 1))
  )
)
```

Functional programming

conclusion

- recursion instead of iteration
- lazy evaluation
- functions as first-class citizens
- impossible, because I/O is state
- functional is hip, not in industry

Logic programming

- PROgramming LOGic
- eg. Prolog, Erlang, Scala
- everything is a logical equation
- programming against "truth"
- computational intensive
- used merely domain-specific

Conclusion—concepts

- assignment
- iteration
- abstraction
- records
- classes
- immutability
- recursion

Conclusion—paradigms

- structured
- object-oriented
- functional
- logic

- So, what about the real world?

The real world

- Different approaches, different use cases
- Use the right tool for the job
- programming can be fun
- good programmers learn a language every year

Multi-paradigm languages

- several paradigms in parallel

Lua
The Perl Programming Language
python™
php
Scala

Thanks!

Keep on hacking and always make a backup!

Q / A?

http://lukas-prokop.at/talks