# IB111 Programování a algoritmizace

Programovací jazyky

# Programovací jazyky

- Programovací jazyk
  - Prostředek pro zápis algoritmů, jež mohou být provedeny na počítači
- Program
  - Zápis algoritmu v programovacím jazyce.

# Dělení programovacích jazyků

- Podle míry abstrakce
  - Vyšší programovací jazyky
    - Sem patří většina jazyků
    - Např. C/C++, Pascal, Basic, Java, PHP
  - Nižší programovací jazyky
    - Assembler („jazyk symbolických adres“)
    - Strojový jazyk (strojový kód)

# Dělení programovacích jazyků

- Podle způsobu překladu a spuštění
  - Kompilované jazyky
    - Kompilátor přeloží program kompletně do kódu cílového systému (strojového kódu).
    - Např. C/C++, Pascal
    - Vyžaduje překlad ale přeložený program běží rychle.
  - Interpretované jazyky
    - Při spuštění programu je spuštěn interpret a ten interpretuje program v programovacím jazyce.
    - Není nutný překlad, ale pro běh programu je nutný interpreter, který musí načíst program a ten postupně interpretovat. Výsledný běh je pomalejší.

# Interpretované jazyky

- Protože plně interpretované jazyky běží relativně pomalu, byly navrženy úpravy pro zvýšení rychlosti běhu
  - Kompilace do mezikódu
    - Interpretuje se pak mezikód ne originální program (např. Java, Python)
  - Za běhu se program zkompiluje do strojového kódu počítače
    - Po iniciálním překladu pak běží rychle (např. Java JIT)

# Interpretované jazyky

- Výhodou interpretovaných jazyků je jejich platformní nezávislost
  - Je možné distribuovat jeden program, který „poběží“ na řadě platforem
- Nevýhodou interpretovaných jazyků je pomalost běhu
  - Buď plně interpretované, nebo alespoň zpoždění díky iniciálnímu překladu
- Některé jazyky mohou být implementovány jako kompilované i interpretované
  - Např. Java, BASIC

# Vyšší programovací jazyky

- Imperativní (procedurální)
  - Strukturované (např. C, Pascal)
  - Objektově orientované (např. C++, Java)
- Deklarativní (neprocedurální)
  - Funkcionální (např. Lisp, Haskell)
  - Logické (např. Prolog, Gödel)
- Některé jazyky mohou kombinovat přístupy
  - Např. C/C++ kombinuje str. a obj. or. přístup

# Historie

- 30. a 40. léta minulého století
  - Lambda kalkul a Turingův stroj
  - Vhodné jako matematické vyjádření algoritmu
- Strojové kódy jednotlivých počítačů a jejich assemblery
- V polovině 50. let vznikl první jazyk vyšší úrovně

# Historie jazyků vyšší úrovně

- FORTRAN (1954)
  - Pojmenování proměnných, složené výrazy, podprogramy, …
  - Vědeckotechnické výpočty
- ALGOL (1960)
  - Matematické algoritmy
- COBOL (1960), BASIC (1964)
  - Syntaxe podobná angličtině
- Pascal (1971), C (1972), Ada (1983)
  - Dodnes používány pro nové projekty

# Imparativní programování

- Příklad Fortran:

```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
      INTEGER A,B,C
      READ(5,501) A,B,C
  501 FORMAT(3I5)
      IF(A.EQ.0 .OR. B.EQ.0 .OR. C.EQ.0) STOP 1
      S = (A + B + C) / 2.0
      AREA = SQRT( S * (S - A) * (S - B) * (S - C))
      WRITE(6,601) A,B,C,AREA
  601 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,12HSQUARE UNITS)
      STOP
      END
```

# Imparativní programování

- Příklad Basic

```
10 INPUT "What is your name: ", U$
20 PRINT "Hello "; U$
30 INPUT "How many stars do you want: ", N
40 S$ = ""
50 FOR I = 1 TO N
60 S$ = S$ + "*"
70 NEXT I
80 PRINT S$
90 INPUT "Do you want more stars? ", A$
100 IF LEN(A$) = 0 THEN GOTO 90
110 A$ = LEFT$(A$, 1)
120 IF A$ = "Y" OR A$ = "y" THEN GOTO 30
130 PRINT "Goodbye "; U$
140 END
```

# Historie - OOP

- Smalltalk-80 (1980)
  - Principy převzaty ze Simuly
- C++ (1985)
  - Objektově orientovaná verze jazyka C (původně jen rozšíření)
- Perl (1987)
  - Populární pro tvorbu CGI skriptů
- Python (1990)
  - Skriptovací jazyk, vhodný i pro větší aplikace
- PHP (1994)
  - Webové programování
- Java (1994)
  - Platformní nezávislost

# OOP příklad

- Příklad Java

```
class myfirstjavaprog
{
        public static void main(String args[])
        {
            System.out.println("Hello World!");
        }
}
```

```
public class ReadArgs
{
    public static final void main(String args[])
    {
        for (int i=0;i<args.length;++i)
        {
            System.out.println( args[i] );
        }
    }
}
```

# OOP příklad

- Příklad Perl

```
@P=split//,".URRUU\c8R";@d=split//,"\nrekcah xinU / lreP rehtona tsuJ";sub p{
@p{"r$p","u$p"}=(P,P);pipe"r$p","u$p";++$p;($q*=2)+=$f=!fork;map{$P=$P[$f^ord
($p{$_})&6];$p{$_}=/ ^$P/ix?$P:close$_}keys%p}p;p;p;p;p;map{$p{$_}=~/^[P.]/&&
close$_}%p;wait until$?;map{/^r/&&<$_>}%p;$_=$d[$q];sleep rand(2)if/\S/;print
```

```
#!/usr/bin/perl

use strict;
use warnings;

open(my $fh, "hello.txt") or die("Can't open hello.txt");

while(my $line = <$fh>) {
    print $line;
}

close $fh;
```

# OOP příklad

- Příklad PHP

```
<html>
        <head>
        <title>PHP Test&lt;/title>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        </head>

<body>
        <h1>PHP Test</h1> &lt;p>
                <b>An Example of PHP in Action</b><br />
                <?php echo "The Current Date and Time is: <br />";
                echo date("g:i A l, F j Y.");?> &lt;/p>
        <h2>PHP Information</h2>
                <p> <?php phpinfo(); ?> </p>
        </body>
</html>
```

# Deklarativní programování

- Definuji „co se má udělat“ (cíl)
  - Ne jak se to má udělat
- Mohu se tak vyvarovat implementačních chyb
- Nepoužíváme for nebo while cyklus, proměnné se používají jen omezeně
  - Vše je řešeno pomocí rekurse
- Výsledkem je však snížená optimalita běhu programu

# Logické programování - Prolog

- Příklad v Prologu

```
rodic(ladislav, adriana).
rodic(ladislav, lubomir).
rodic(sarka, lubomir).

sourozenec(Sourozenec, X) :- rodic(Y, X), rodic(Y, Sourozenec),
    \+(Sourozenec = X).
```

Dotaz: **sourozenec(adriana, X)**.
Výstup: **X = lubomir ;**
**No**

Dotaz: **rodic(X, lubomir).**
Výstup: **X = ladislav ;**
**X = sarka ;**
**No**

# Funkcionální programování

- Příklad v Haskellu

```
factorial :: Integer -> Integer
factorial 0 = 1
factorial n = n * factorial (n-1)
```

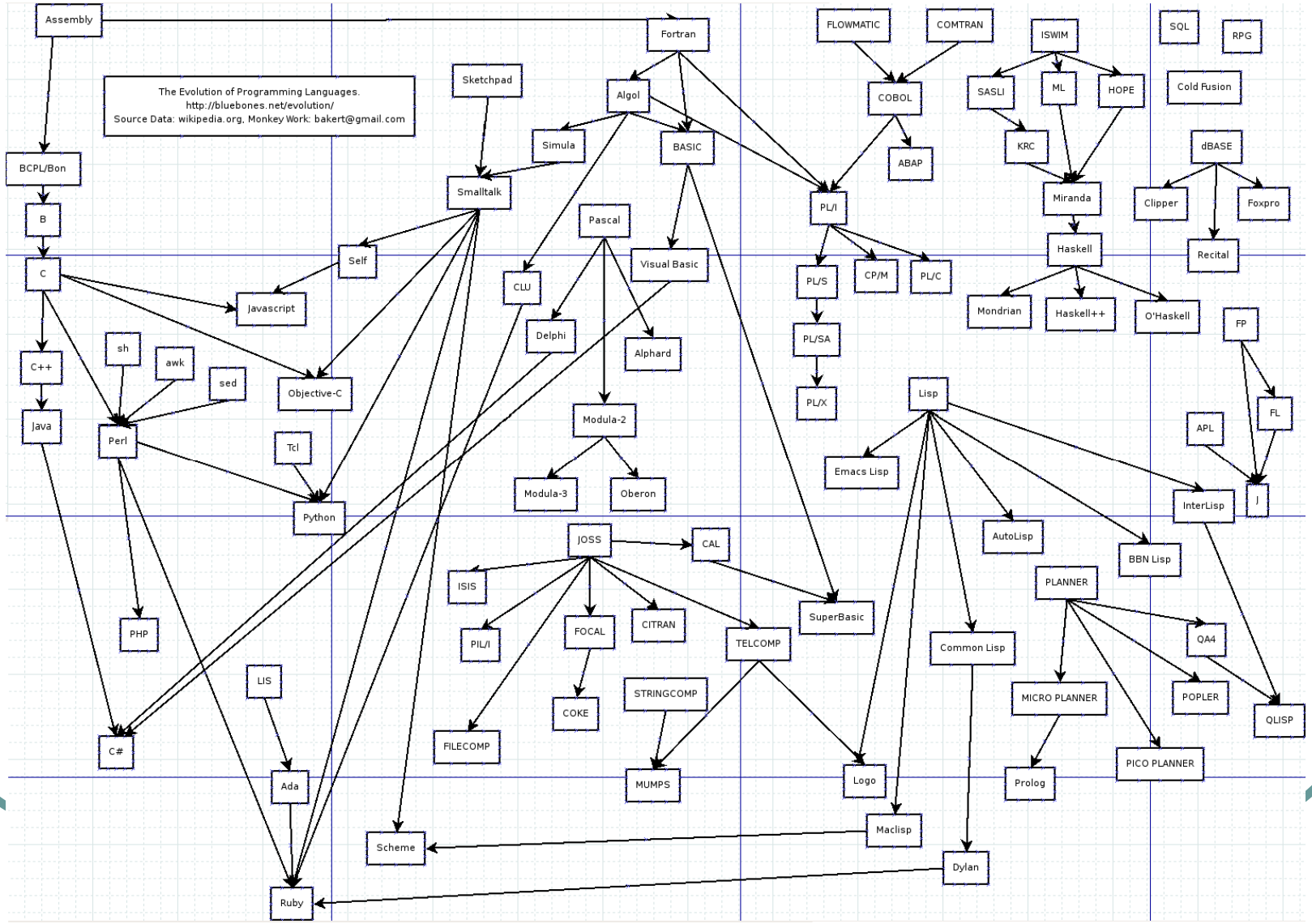
```
data Sex = Male | Female
data Person = Person String Sex Int -- Notice that Person is both a constructor and a type

-- An example of creating something of type Person
tom :: Person
tom = Person "Tom" Male 27
```

# Přehled programovacích jazyků

3APL – Ada – Algol – JSA – Baltík – Baltazar – BASIC – C – C++ – C# – COBOL – ColdFusion – Clean – Delphi – Eiffel – Erlang – Flex – Forth – Fortran – FoxPro – Gödel – Haskell – Java – JavaScript – J# – Lisp – Miranda – Magic – Modula-2 – Oberon – Object Pascal – Objective-C – Karel – Pascal – Perl – PHP – PL/1 – PL/SQL – Prolog – Python – Ruby – Scheme – Simula 67 – Smalltalk – Tcl/Tk – VBScript – Whitespace

shell (bash) – PostScript – POV-Ray – SQL – TeX – XSLT

# Přehled programovacích jazyků



# Který jazyk vybrat

- Ideální programovací jazyk použitelný pro všechny systémové a aplikační programy neexistuje.
- Každý jazyk má své pro a proti
  - Právě kvůli nevýhodám existujících jazyků vznikaly a vznikají jazyky nové
- Rychlost programování vs. rychlost běhu programu
  - Musíme zvolit rozumný kompromis

# Který jazyk vybrat

- Nemá smysl programovat rok program v assembleru abych ho spustil jedinkrát pro provedení určitého krátkého výpočtu
  - Program v Basicu bude sice běžet výrazně pomaleji, ale budu schopen ho naprogramovat podstatně rychleji
- Rychlost hraje roli u kódu, který je spouštěn velice často
  - Operační systém, zatížené webové servery, souborové servery apod.

# Který jazyk vybrat

- Obvykle tedy vybíráme jazyk, ve kterém bude vyřešení našeho problému nejsnazší
  - Až pokud se výsledné řešení ukáže jako pomalé, má smysl jej optimalizovat
    - Např. přepsáním kritických částí do jiného jazyka

# Dostupnost knihoven

- V většině případů nemusíme řešení problému programovat úplně sami
- Existuje řada existujících knihoven, které můžeme využít
- Mnoho knihoven je specifických pro určité jazyky
- Volbu jazyka proto může ovlivnit i dostupnost knihoven pro oblasti, které nás při řešení problému zajímají
- Např. pro Javu existuje obrovské množství existujících knihoven

# Kombinace jazyků

- V praxi může být výhodné kombinovat několik jazyků/přístupů
- Př. Programování pro WWW
  - Oddělujeme funkci (aplikační logiku - kód na serveru, např. PHP), data (databázový server, např. SQL), vzhled a uživatelské rozhraní (HTML, CSS, Javascript).
  - Příklad na Javascript ve cvičení

# WWW programování

- Příklad PHP
  + MySQL

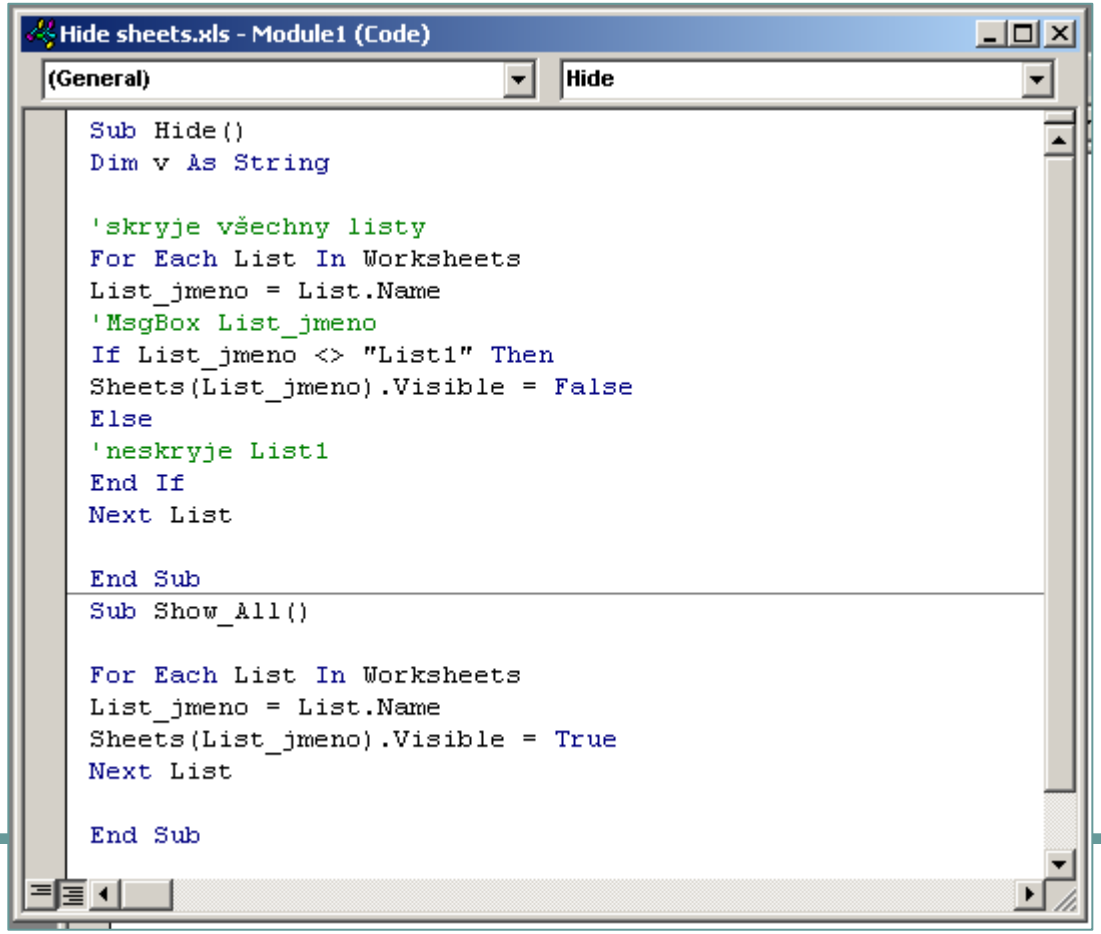
```
<?
$username="username";
$password="password";
$database="your_database";

mysql_connect(localhost,$username,$password);
@mysql_select_db($database) or die( "Unable to select database");
$query="SELECT * FROM contacts";
$result=mysql_query($query);

$num=mysql_numrows($result);

mysql_close();

echo "<b><center>Database Output</center></b><br><br>";

$i=0;
while ($i < $num) {

$first=mysql_result($result,$i,"first");
$last=mysql_result($result,$i,"last");
$phone=mysql_result($result,$i,"phone");
$mobile=mysql_result($result,$i,"mobile");
$fax=mysql_result($result,$i,"fax");
$email=mysql_result($result,$i,"email");
$web=mysql_result($result,$i,"web");

echo "<b>$first $last</b><br>Phone: $phone<br>Mobile: $mobile<br>Fax:
$fax<br>E-mail: $email<br>Web: $web<br><hr><br>";

$i++;
}

?>
```

# Jaký jazyk vybrat

- Jak to vypadá v praxi?
- Systémové programování (OS, utility)
  - Dříve assembler a C, dnes C++.
- Aplikační programy
  - Java, C++, Dephi
- WWW - aplikační logika
  - Perl, Python, Java, PHP, ASP
- Vědeckotechnické výpočty
  - Matlab, Fortran
- Kancelářské balíky
  - Visual Basic for Applications (VBA)

# Programování pro MS Office

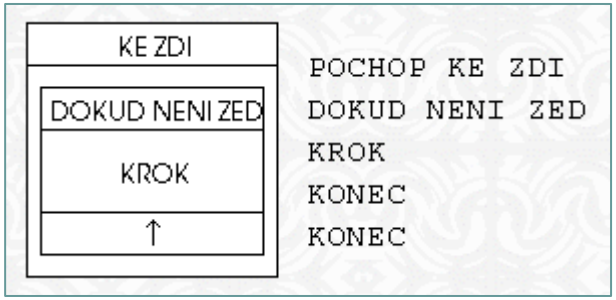
- Př. Visual Basic (Excel)

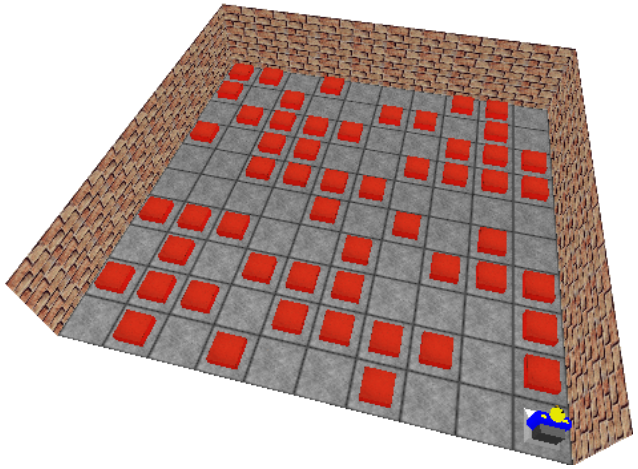


# Jazyky pro výuku programování

- Př. Karel

```
BEGINNING-OF-PROGRAM

 DEFINE turnright AS
 BEGIN
   turnleft
   turnleft
   turnleft
 END

 BEGINNING-OF-EXECUTION
   ITERATE 3 TIMES
     turnright

   turnoff
 END-OF-EXECUTION

END-OF-PROGRAM
```

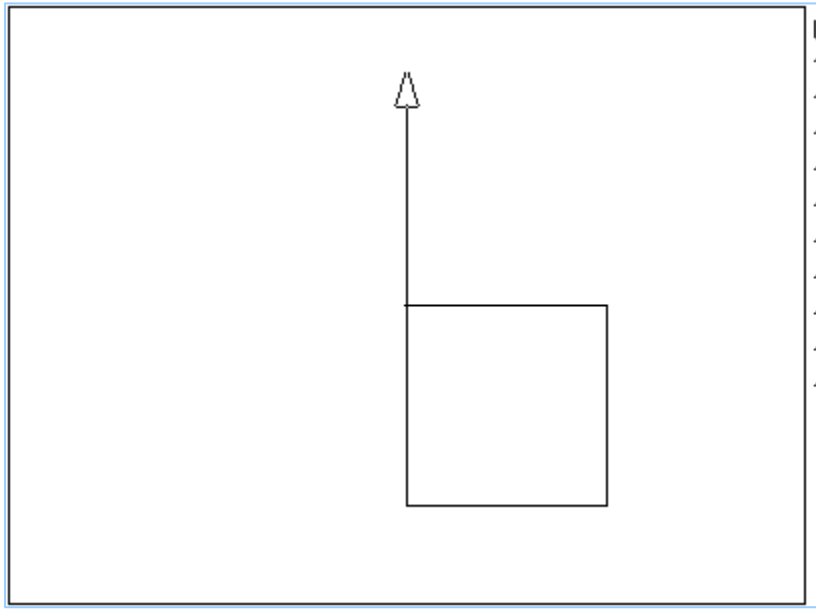


# Jazyky pro výuku programování

- Př. Logo

```
PŘÍKAZ ČTVEREC
  OPAKUJ 4 [
    DOPŘEDU 100
    VPRAVO 90
  ]
KONEC
```



```
Welcome to Berkeley Logo version 5.5
? right 90
? forward 100
? right 90
? forward 100
? right 90
? forward 100
? right 90
? forward 100
? forward 100
?
```

# Příští přednáška

- Následuje cvičení v A104 v 14:00
- Příští (poslední) přednáška 13.12. v B410 v 8:00