C2110 *Operační systém UNIX a základy programování*

5. lekce

Programy vs skripty, algoritmizace, bash

Petr Kulhánek

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta
Masarykova univerzita, Kamenice 5, CZ-62500 Brno

Průběžný test I

Průběžný test I

➢ Test prostřednictvím odpovědníku v IS

Student – Odpovědníky – C2110 – Test 1a, 1b (dle seminární skupiny)

Délka 20 minut.

Je možné sestavit pouze jednu sadu otázek.

Používejte průběžné uložení.

Vyhodnocení je možné pouze jednou.

Je povoleno a doporučeno:

- Testovat příkazy v terminálu.
- Prohledávat manuálové stránky, svoje zápisky a prezentace předmětu.
- Při nejasnostech se přihlaste.

Není povoleno

- Komunikovat s další osobou mimo vyučujícího.

Obsah

- **Programy vs Skripty**
  - kompilované vs interpretované jazyky, příklady
- **Základy programování**
  - algoritmizace, algoritmus, zápisy algoritmů, datové struktury, operace, vstupně/výstupní operace
- **Bash**
  - interaktivní vs neinteraktivní režim, přímé a nepřímé spouštění skriptů

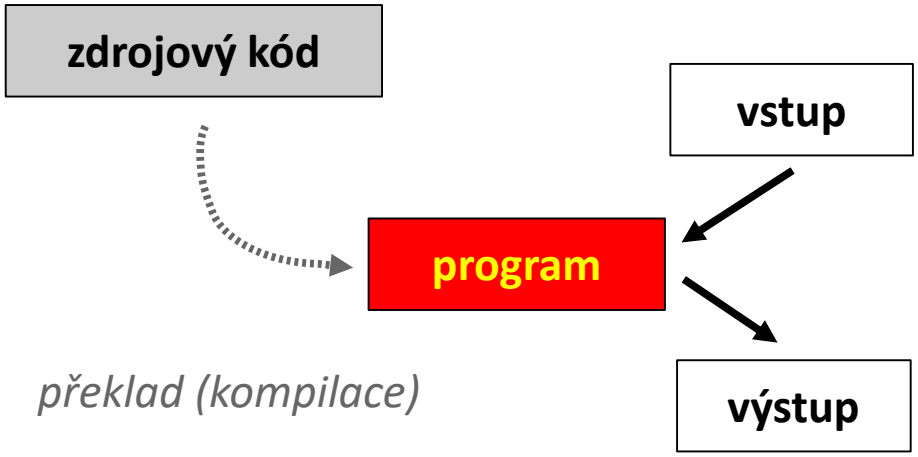
Programy vs skripty

Programy *vs* Skripty

Program je soubor strojových instrukcí zpracovávaných přímo procesorem. Program vzniká překladem zdrojového kódu programovacího jazyka.

Překládané jazyky:

C/C++
Fortran

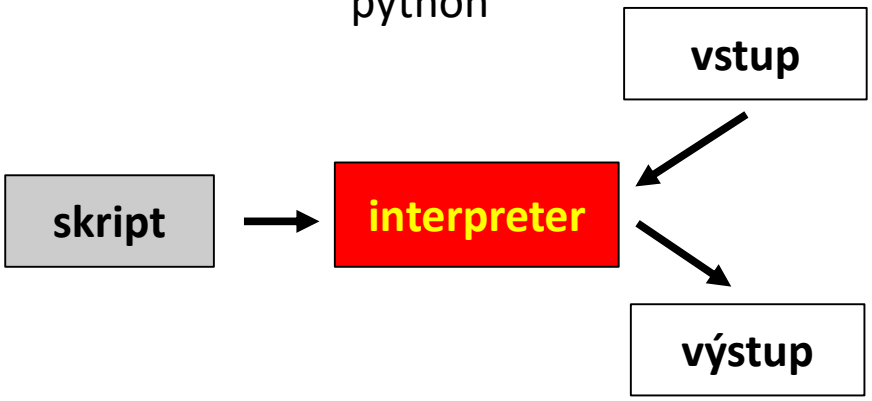


překlad (kompilace)

Skript je textový soubor obsahující příkazy a řídící sekvence, které jsou vykonávány interpretem použitého skriptovacího jazyka.

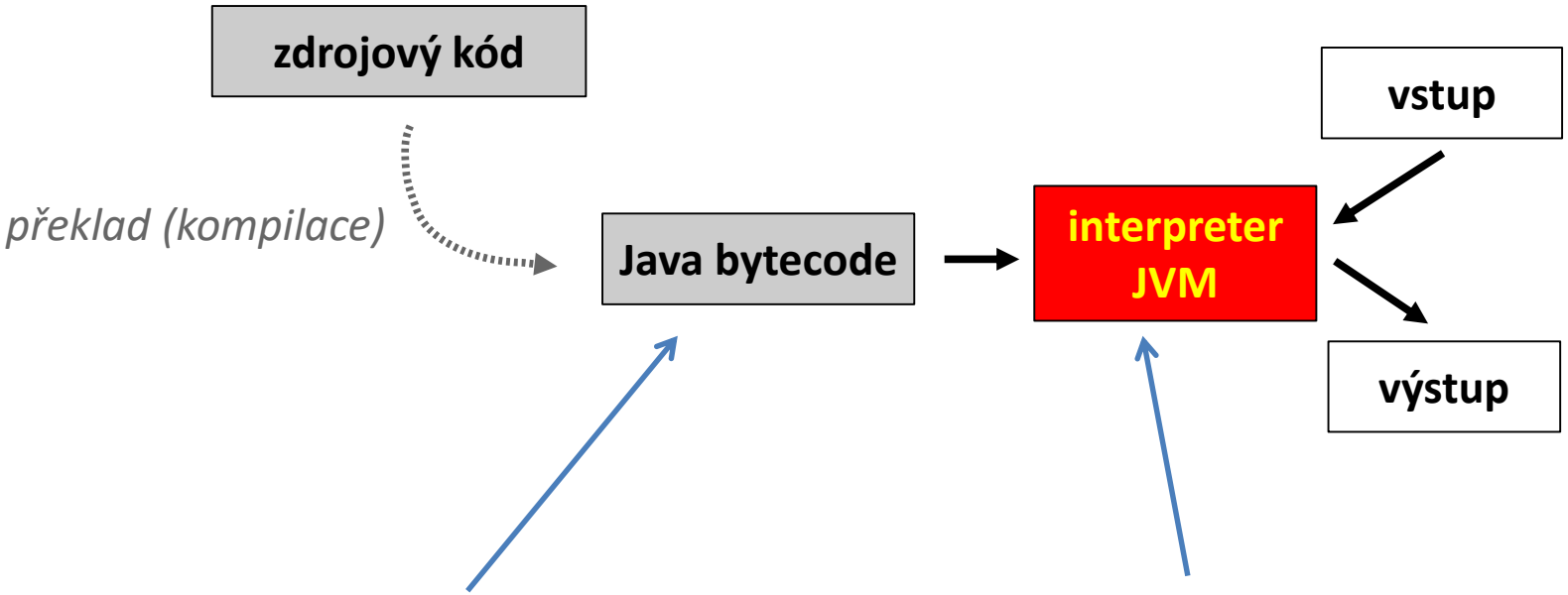
Skriptovací jazyky:

bash
gnuplot
awk
JavaScript
PHP
python



A co JAVA?

Existují i různé kombinace obou přístupů. Typickým příkladem je programovací jazyk Java.



Java bytecode
(nezávislý na platformě – OS a HW)

Java Virtual Machine
interpreter Java bytekódu
(program sestavený pro danou platformu – OS a HW)

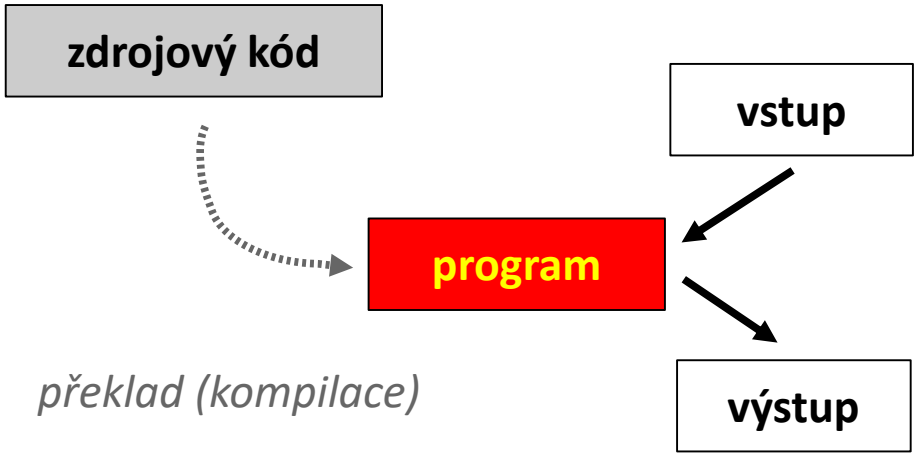
Programy *vs* Skripty, ...

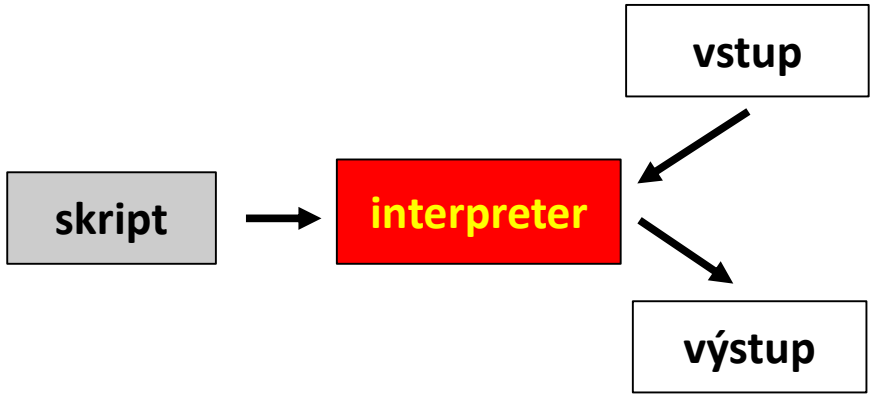
- snadná optimalizace
- rychlé vykonávání

- nutnost rekompilace
- nelze vytvářet samospustitelný kód

- nevyžaduje rekompilaci
- vytváření samospustitelného kódu

- špatná optimalizovatelnost
- pomalejší vykonávání





Program v jazyce C

Zdrojový kód

```
#include <stdio.h>

int main(int argc,char* argv[])
{
  printf("Tohle je program v jazyce C!\n");
  return(0);
}
```

Kompilace

```
$ gcc program.c -o program
```

kompiler jazyka C

název souboru s vytvořeným programem

Spuštění programu

```
$ ./program
```

soubor **program** musí mít práva **pro spuštění**

Skript v Bashi

Skript

```
#!/bin/bash

echo 'Toto je skript v interpretu Bash!'
```

Spuštění skriptu

```
$ bash skript.bash
```

soubor **skript.bash** **nemusí** mít práva **pro spuštění**

interpret Bash

Cvičení I

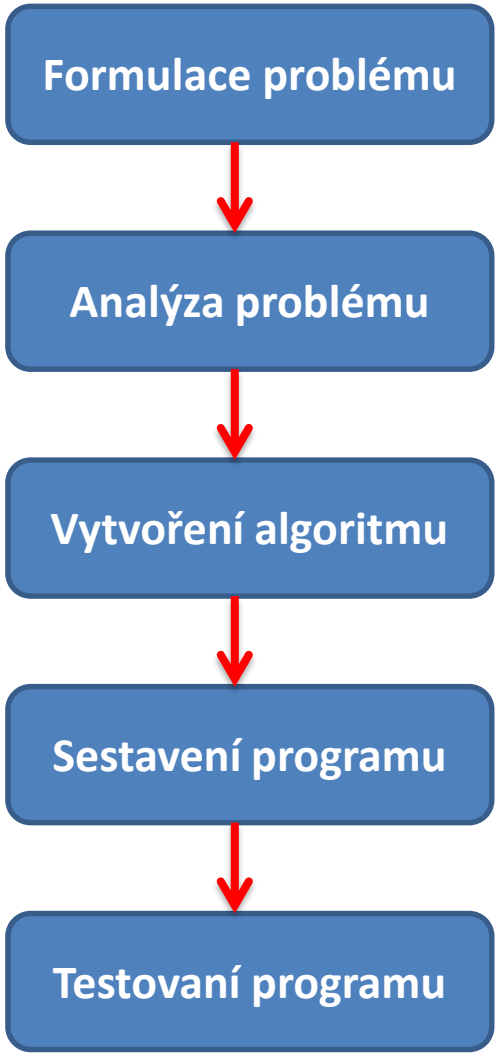
1. Vytvořte adresáře s názvy ukol01 a ukol02 ve vašem domovském adresáři.
2. Do jednotlivých adresářů uložte postupně soubory program.c a skript.bash z adresáře /home/kulhanek/Documents/C2110/Lesson05/programs
3. Zkompilujte zdrojové kódy programu napsaného v jazyce C. Ověřte, že vzniklý program lze spustit.
4. Jaká je velikost souboru obsahující výsledný program vzniklý kompilací zdrojového kódu v jazyce C. Otevřete vzniklý soubor v textovém editoru (gedit). Co soubor obsahuje?
5. Ověřte funkčnost skriptu skript.bash jeho spuštěním.
6. Změňte soubory program.c a skript.bash, tak aby výsledný program či skript vypisovaly jiný text.

Základy programování

http://www.spsemoh.cz/vyuka/algor/

https://cs.wikipedia.org/wiki/Algoritmus

Algoritmizace



Formulace problému
Je nutné přesně formulovat požadavky, určit výchozí data, požadované výsledky a přesnost řešení (u numerických úloh).

Analýza problému
Ověříme, že je úloha řešitelná pro očekávané vstupní data a podle charakteru úlohy navrhneme způsob nejvhodnějšího řešení.

Vytvoření algoritmu
Sestavíme jednoznačný sled operací, které je třeba provést, aby byla úloha správně vyřešena.

Sestavení programu
Na základě algoritmu vytvoříme zdrojový text programu ve zvoleném programovacím jazyce.

Testování programu
Nalezení syntaktických chyb ve zdrojovém kódu a logických chyb ve vlastním návrhu. Ověření funkčnosti programu na zadaných datech.

Algoritmus

Algoritmus je přesný návod či postup, kterým lze vyřešit daný typ úlohy. Pojem algoritmu se nejčastěji objevuje při programování, kdy se jím myslí teoretický princip řešení problému (oproti přesnému zápisu v konkrétním programovacím jazyce). Obecně se ale algoritmus může objevit v jakémkoli jiném vědeckém odvětví.

Požadované vlastnosti:

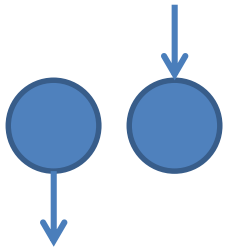
- **Determinovanost** - algoritmus musí být přesný, srozumitelný a jednoznačný, tj. v každém místě je jednoznačně určen další krok a pro stejná vstupní data musí poskytovat stále stejné výsledky. (Činnost algoritmu nesmí záviset na libovůli osoby ani na vlastnostech zařízení, které ho realizují).
- **Hromadnost** - algoritmus neslouží k řešení jen jedné úlohy, ale je řešením celé skupiny úloh, které se od sebe liší jen vstupními údaji. Vstupní údaje se mohou měnit v určitých mezích.
- **Konečnost** - hledané výsledky musíme získat po konečném počtu kroků, algoritmus musí po konečném počtu kroků skončit.

Zápis algoritmů:

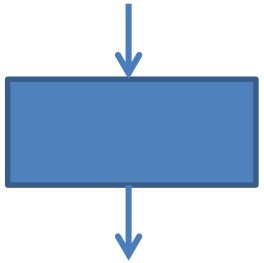
- slovně
- pseudokód
- graficky (vývojový diagram, apod.)

Vývojový diagram

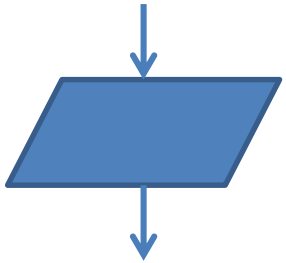
Vývojový diagram je grafické vyjádření algoritmu. Diagram se skládá ze značek (bloků), které se vykonávají v pořadí od začátku do konce diagramu ve směru šipek.



spojka (začátek, konec)

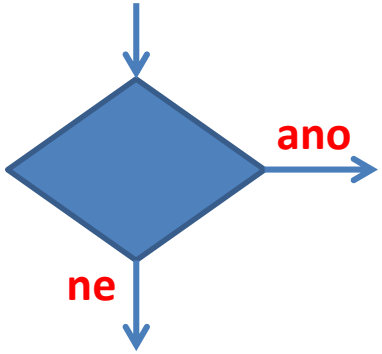


blok zpracování



blok vstupu/výstupu

Značky se kombinují tak, aby diagram popsal jednoznačný sled operací, které je třeba provést, aby byla úloha správně vyřešena. Do jednotlivých značek se vpisují operace nebo skupiny operací, které se mají provést.

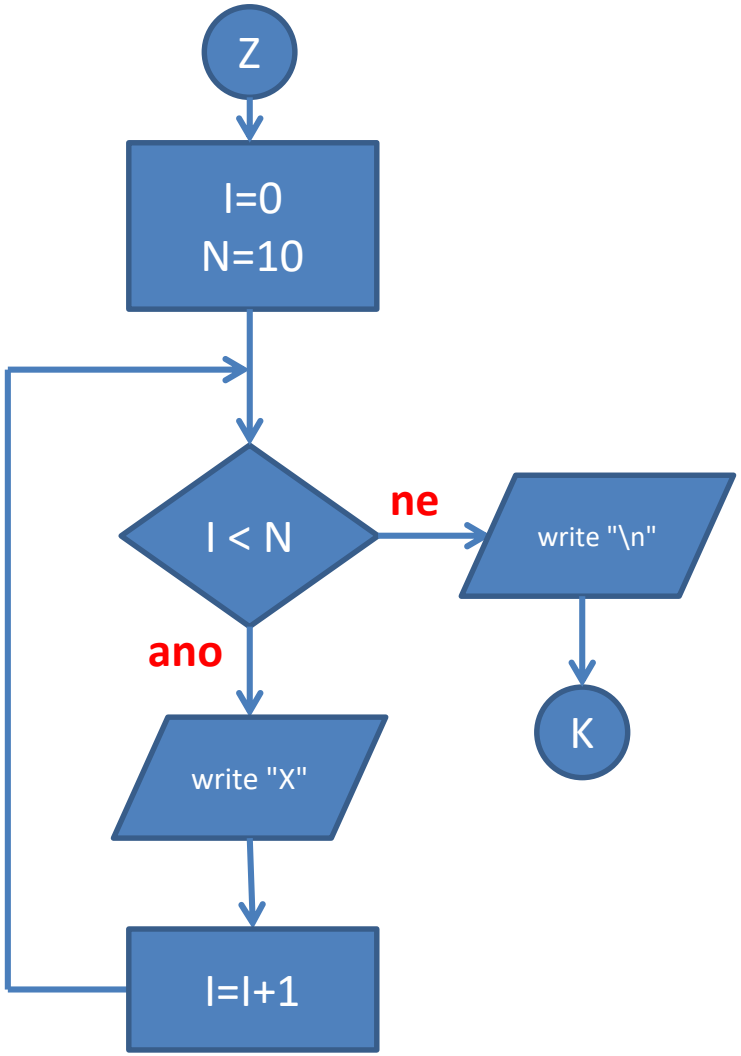


blok rozhodování

Existují i jiné značky, které však zatím nebudeme používat.

Příklady

Vývojový diagram:

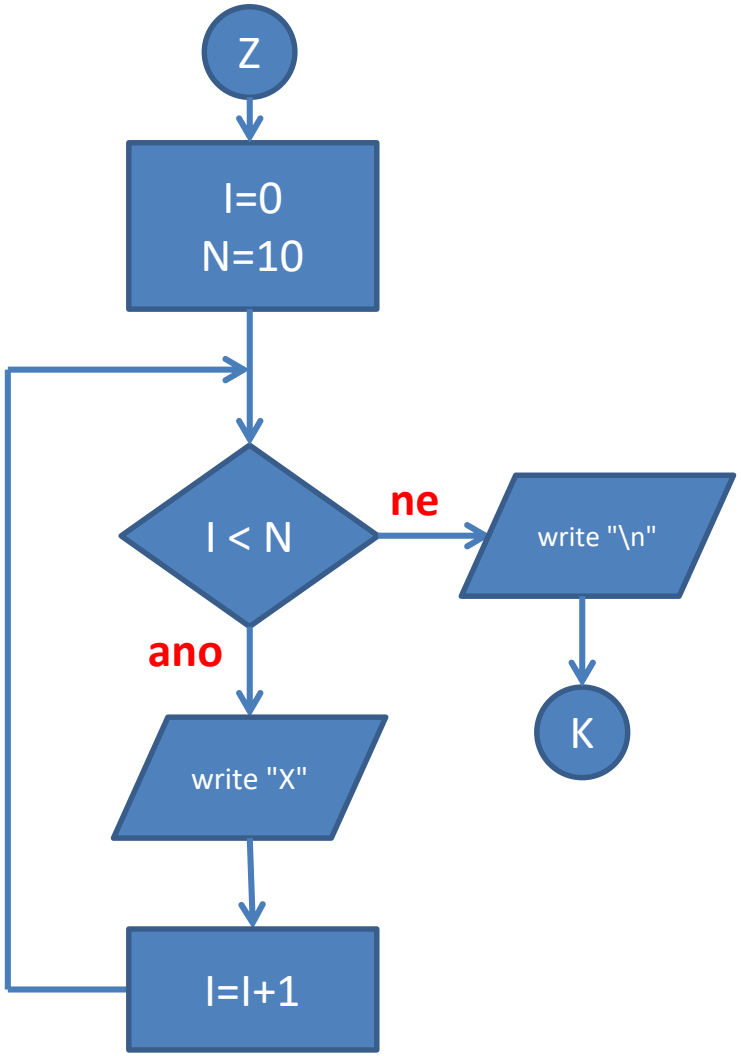


Slovní popis:

1. Vlož do proměnné I hodnotu 0
2. Vlož do proměnné N hodnotu 10
3. Je hodnota proměnné I menší než N?
   ANO – pokračuj bodem 4
   NE – pokračuj bodem 7
4. Vytiskni znak X.
5. Zvětši hodnotu proměnné I o jedničku.
6. Pokračuj v bodě 3.
7. Vytiskni konec řádku.
8. Konec

Příklady

Algoritmus



Skript v Bashi

```
#!/bin/bash

I=0
N=10

while [ $I -lt $N ]; do
    echo -n "X"
    ((I=I+1))
done
echo ""
```

Výsledek:

```
$ ./my_skript
XXXXXXXXXX
$
```

Datové struktury

Datové struktury slouží pro ukládání dat. Mezi základní datové struktury patří:

a) **proměnná**
b) pole
c) záznam
d) objekt

Proměnná je **pojmenované umístění** v paměti, které **obsahuje hodnotu**. Každá proměnná je určitého **typu**, který omezuje možné operace nad proměnnou. Typ proměnné může být určen při vytváření proměnné (explicitní typ) nebo může být určen až při použití proměnné (implicitní typ). Typ proměnné má vliv na způsob uložení dat v paměti počítače.

Příklady:

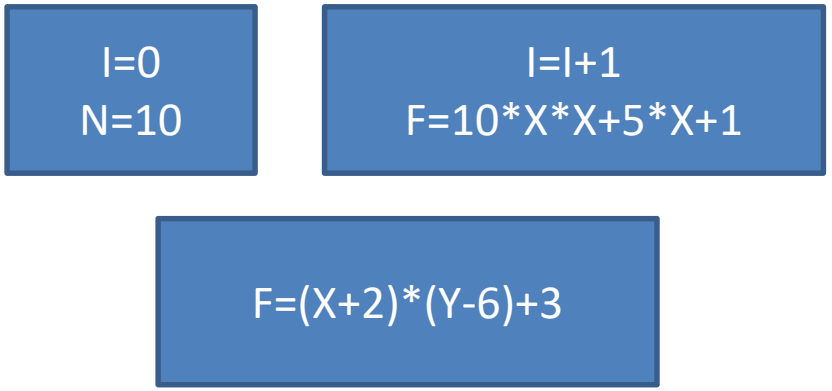
`A="pokusna hodnota"`	text (řetězec)
`B=5`	celé číslo
`C=10.458`	reálné číslo (číslo v pohyblivé řádové čárce)
`D=B+C`	
~~`E=A+B`~~	

Operace

Základní operace:

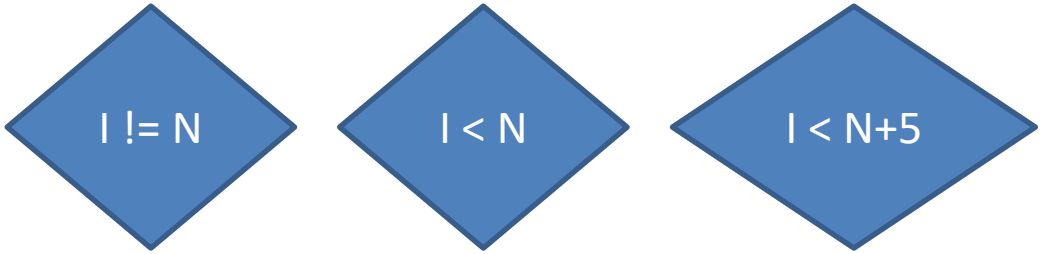
=	přiřazení
+	sčítání
-	odčítání
*	násobení
/	dělení

blok zpracování



Logické operace:

==	rovná se
!=	nerovná se
<	menší
<=	menší nebo rovno
>	větší
>=	větší nebo rovno



blok rozhodování

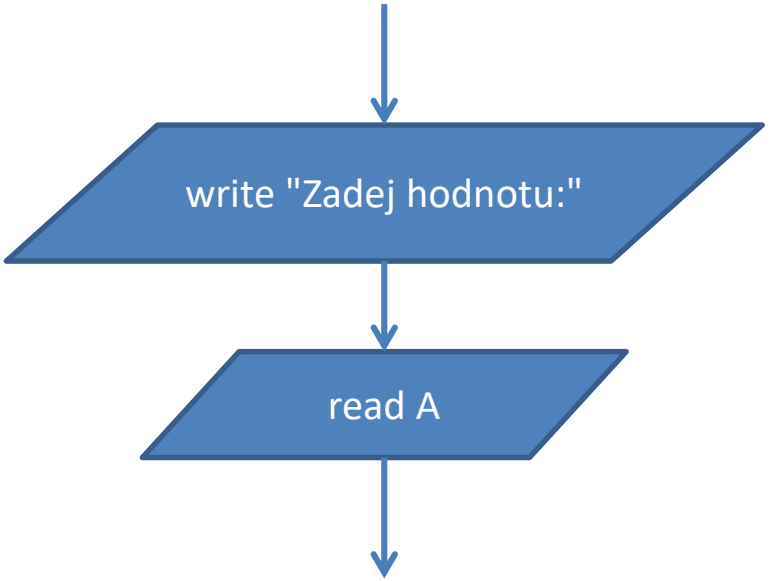
Vstup

Vstupem pro program mohou být informace zadané uživatelem z terminálu, přesměrované ze souboru, nebo z jiného programu pomocí roury.

Základní operace (pseudokód):

`read var` načti hodnotu do proměnné *var*

Příklad:



Program vypíše dotaz pro uživatele a očekává vstup, který je po zadání uživatelem načten do proměnné A.

Takto definovaný vstup odráží základní možnosti jazyka bash. Jiné možnosti vstupu v této fázi nedoporučuji používat.

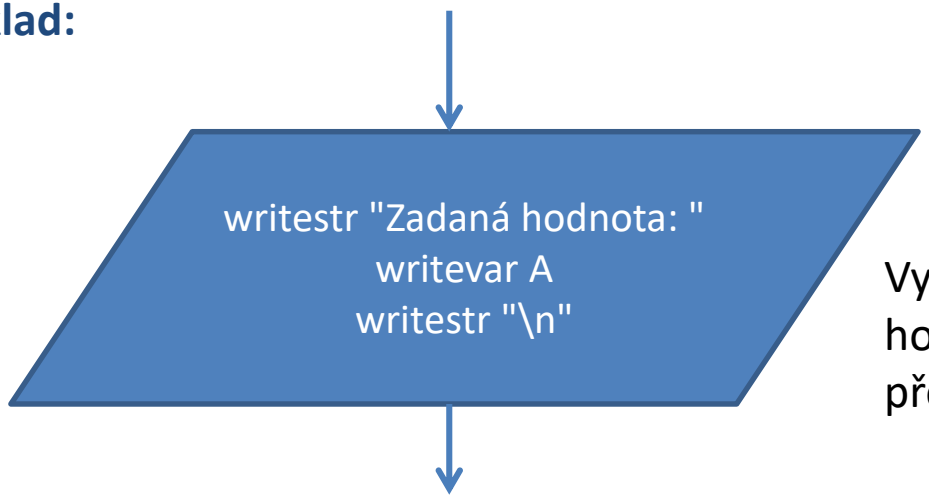
Výstup

Výstupem je terminál. Terminál se chová jako tiskárna, ve které nelze vzít zpět již jednou vytištěný znak. Kromě toho umí tiskárna přejít na začátek nového řádku tak, že do terminálu zapíšeme znak \n.

Základní operace (pseudokód):

`writestr "retezec"`	vytiskne znaky (řetězec, string) uvedené v uvozovkách
`writevar var`	vytiskne hodnotu proměnné var

Příklad:



Vypíše text "Zadaná hodnota: " následovaný hodnotou proměnné A. Kurzor bude přesunut na nový řádek.

Takto definovaný výstup odráží základní možnosti jazyka bash. Jiné možnosti výstupu v této fázi nedoporučuji používat.

Bash

https://www.gnu.org/software/bash/

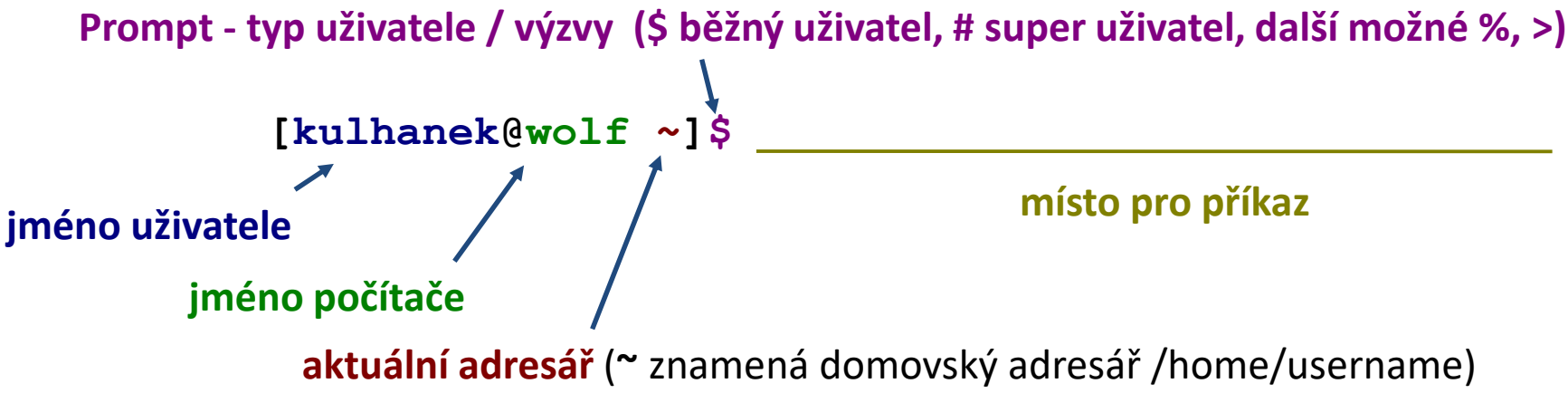
Bash - přehled

Unixový shell (též příkazový procesor, v doslovném překladu „unixová skořápka“) je název **textového uživatelského rozhraní**, které je předchůdcem grafického uživatelského rozhraní.
Jedním z unixových shellů je **Bash**.

Bash je POSIX shell s řadou rozšíření. Je koncipován pro operační systémy založené na projektu GNU a je možné ho spustit na většině unixových operačních systémů. Je používán jako implicitní příkazový interpret v systémech postavených na linuxovém jádře, stejně jako i v Mac OS X nebo v systému Darvin. Je možné ho použít i v systému Microsoft Windows za použití subsystému pro unixové aplikace (SUA), nebo emulace POSIX pomocí softwaru **Cygwin** a MSYS.

https://cs.wikipedia.org

Interaktivní režim



Příkaz se vykoná zmáčknutím klávesy **Enter**.

Historie: pomocí kurzorových šipek nahoru a dolů lze procházet seznamem již zadaných příkazů. Příkaz z historie lze znovu použít nebo upravit a upravený použít. Historie je přístupná i příkazem **history**.

Automatické doplňování: zmáčknutím klávesy Tab (tabulátor) se interpret příkazové řádky snaží dokončit rozepsané slovo. Doplňují se jména příkazů, cesty a jména souborů (pokud jeden stisk nic nevyvolá, existuje více možností doplnění, opakovaný stisk je zobrazí).

Shell interpretuje (expanduje) **divoké znaky a jiné speciální znaky**, před vlastním spuštění příkazu. V interaktivním režimu je možné **spouštět řídící struktury** jazyka bash.

Interaktivní režim se ukončuje příkazem **exit**.

Neinteraktivní režim - skripty

1) Nepřímé spouštění

Spouštíme interpreter jazyka a jako argument uvádíme jméno skriptu.

```
$ bash muj_skript_v_bashi
```

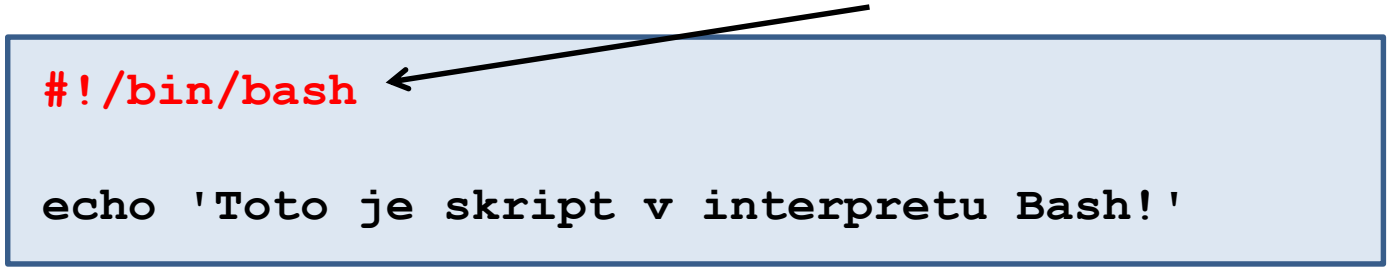
Skripty **nemusí** mít nastaven příznak x (executable).

2) Přímé spouštění

Spouštíme přímo skript (shell automaticky spustí interpreter).

```
$ chmod u+x muj_skript_v_bashi
$./muj_skript_v_bashi
```

Skripty **musí** mít nastaven příznak **x** (executable) a **interpreter** (součást skriptu).



Určení interpretru

Specifikace interpretru (první řádek skriptu):

```
#!/absolutní/cesta/k/interpretru/skriptu
```

Skript v bashi

```
#!/bin/bash

echo "Toto je skript v bashi!"
```

Skript v gnuplotu

```
#!/usr/bin/gnuplot

set xrange[0:6]

plot sin(x)

pause -1
```

- Pokud není interpreter skriptu při jeho přímém spuštění uveden, použije se interpreter systémového shellu (bash).
- Interpreter uvedený ve skriptu se ignoruje při nepřímém spuštění.
- Interpretr je vhodné do skriptu vždy uvádět, protože je použit textovými editory pro zvýrazňování syntaxe.

Určení interpretru, II

Pokud se absolutní cesta k interpretru mění (např. při použití softwarových modulů), lze použít následující konstrukci:

```
#!/usr/bin/env interpreter
```

Interpreter musí být v některém adresáři určeném systémovou proměnnou PATH.

Skript v bashi

```
#!/usr/bin/env bash

echo "Toto je skript v bashi!"
```

Skript v gnuplotu

```
#!/usr/bin/env gnuplot

set xrange[0:6]

plot sin(x)

pause -1
```

Cvičení II

1. V terminálu spusťte příkaz bash. Co se stalo? Druhé sezení ukončete příkazem exit.
2. Příkazem ps zjistěte číslo procesu, který interpretuje příkazovou řádku.
3. Proces ukončete příkazem kill (kill -9 pid, kde pid je číslo procesu). Co se stane?
4. Skript skript.bash z adresáře ukol2 spusťte přímo.

Závěr

Závěr

- **Program** je binární soubor vykonávaný přímo procesorem. **Skript** je textový soubor, který je vykonáván interpretem.
- **Algoritmizace** je způsob transformace úlohy do popisu (algoritmu), který je snadno přepsatelný do zdrojového kódu programovacího jazyka nebo skriptu. Základním principem je **rozložení úlohy na elementární části** a uvedení jejich přesného postupu provádění.
- **Bash** je unixový shell, který interpretuje příkazovou řádku a zároveň obsahuje podporu pro spouštění skriptů.

Domácí úkoly

- **Algoritmizace**

Domácí úkoly - pokyny

1. Vytvořte vývojové diagramy pro následující úlohy. Ve vývojových diagramech použijte pouze značky a operace včetně vstupně-výstupních, které jsou uvedeny v této prezentaci.
2. Diagramy kreslete ve vhodném software:
   - např. program **dia** (sudo apt-get install dia)
   - online nástroje, např. **www.draw.io**
3. Výsledné diagramy vložte do odevzdávárny L05 v pdf formátu. Jméno souboru bude v následujícím formátu:

   PrijmeniL05Ux.pdf

   kde x je číslo úkolu. Soubory, které budou pojmenovány jiným způsobem, nebudou kontrolovány (automatické změny jména provedené ISem jsou povoleny).
4. Termín pro odevzdání je **5. listopad 2017 23:59**.

Doporučení: U cyklů používejte blok rozhodování před blokem zpracování.

Úkol 1

Do terminálu vytiskněte čtverec se znaků **X**. Délku strany čtverce zadá uživatel.

```
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.

Úkol 2

Do terminálu vytiskněte pravoúhlý trojúhelník se znaků **X**, tak aby jedna odvěsna byla umístěna nahoře a druhá na levé straně. Délku odvěsny zadá uživatel.

```
x x x x x x x x x x
x x x x x x x x x
x x x x x x x x
x x x x x x x
x x x x x x
x x x x x
x x x x
x x x
x x
x
```

Úkol 3

Do terminálu vytiskněte pravoúhlý trojúhelník se znaků **X**, tak aby jedna odvěsna byla umístěna dole a druhá na levé straně. Délku odvěsny zadá uživatel.

```
x
x x
x x x
x x x x
x x x x x
x x x x x x
x x x x x x x
x x x x x x x x
x x x x x x x x x
x x x x x x x x x x
```

Úkol 4

Do terminálu vytiskněte obrys čtverce se znaků **X**. Délku strany čtverce zadá uživatel.

```
X X X X X X X X X X
X                   X
X                   X
X                   X
X                   X
X                   X
X                   X
X                   X
X                   X
X X X X X X X X X X
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.

Úkol 5

Do terminálu vytiskněte obrys čtverce a jeho uhlopříčky se znaků **X**. Délku strany čtverce zadá uživatel.

```
X X X X X X X X X X
X X             X X
X   X         X   X
X     X     X     X
X       X X       X
X       X X       X
X     X     X     X
X   X         X   X
X X             X X
X X X X X X X X X X
```

To, že se nejedná vzhledově o čtverec, ignorujte. Počet znaků **X** na řádku a počet řádků však musí být stejný.