C2110 *Operační systém UNIX a základy programování*

5. lekce / modul 2

PS/2020 Distanční forma výuky: Rev1

Petr Kulhánek

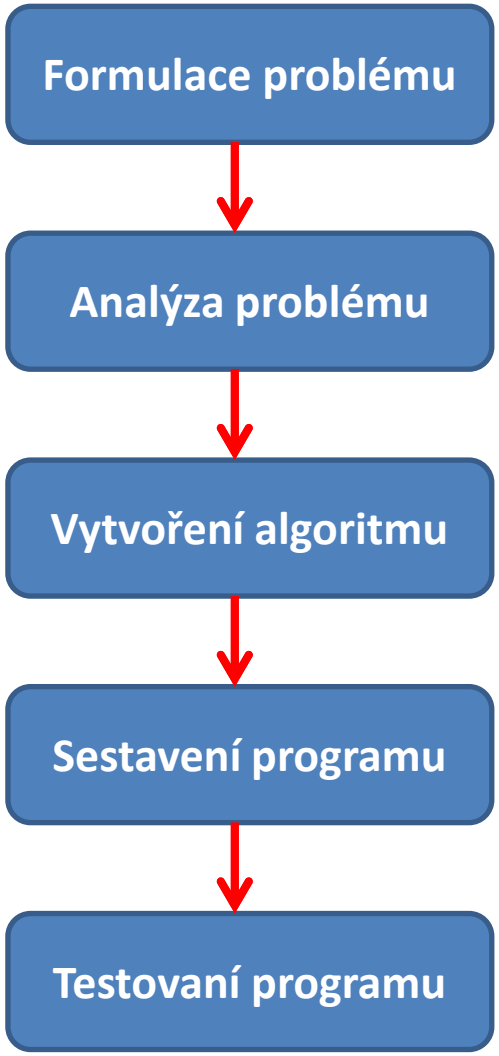
kulhanek@chemi.muni.cz

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

Základy programování

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:

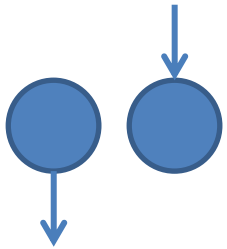
- **Determinovanost** - algoritmus musí být přesný, srozumitelný a jednoznačný, tj. v každém místě je jednoznačně určen další krok a pro stejná vstupní data musí poskytovat stále stejné výsledky. (Činnost algoritmu nesmí záviset na libovůli osoby ani na vlastnostech zařízení, které ho realizují).
- **Hromadnost** - 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:

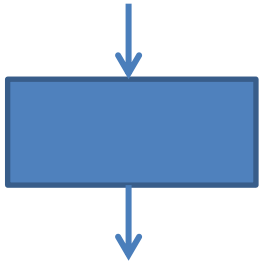
- slovně
- pseudokód
- graficky (vývojový diagram, apod.)

Vývojový diagram

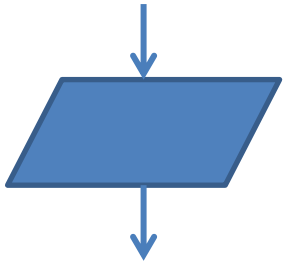
Vývojový diagram je grafické vyjádření algoritmu. Diagram se skládá ze značek (bloků), které se vykonávají v pořadí od začátku do konce diagramu ve směru šipek.



spojka (začátek, konec)

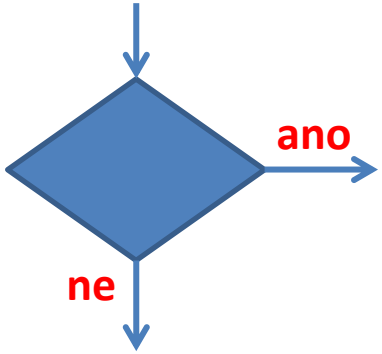


blok zpracování



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.

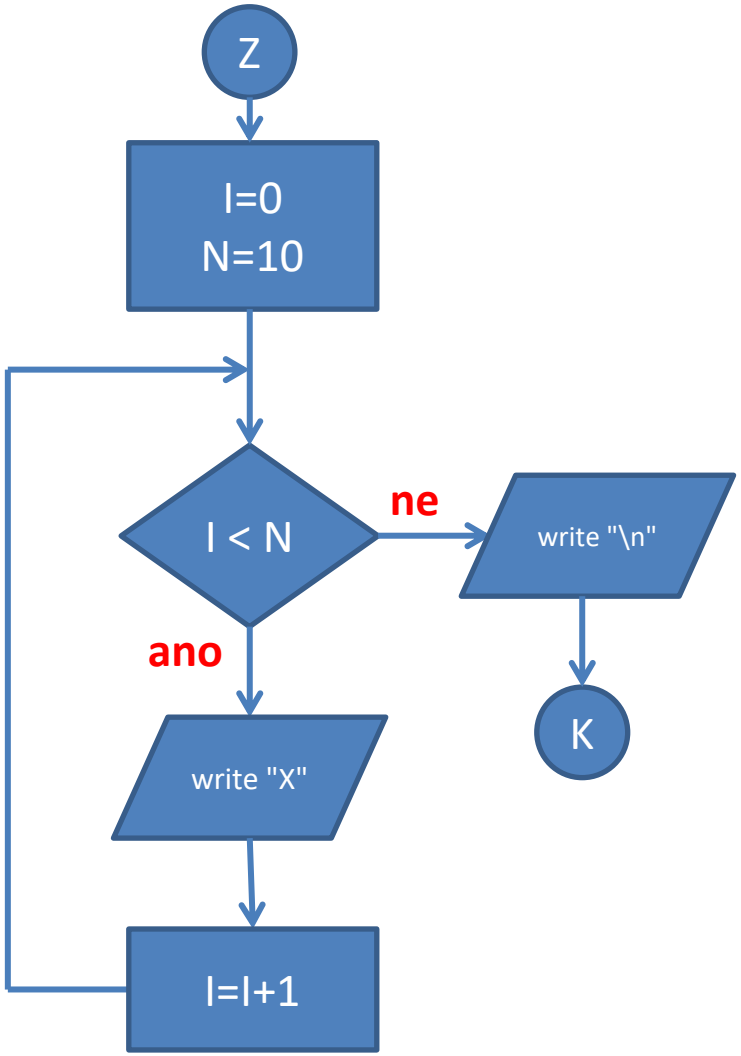


blok rozhodování

Existují i jiné značky, které však zatím nebudeme používat.

Příklady

Vývojový diagram:

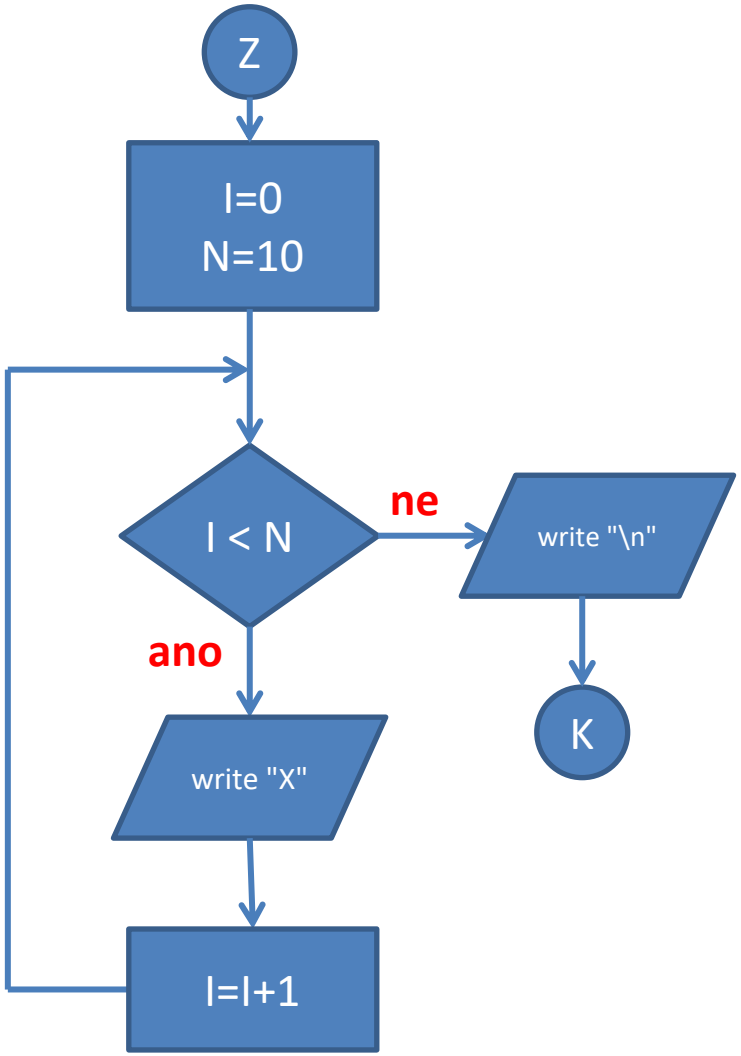


Slovní popis:

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:

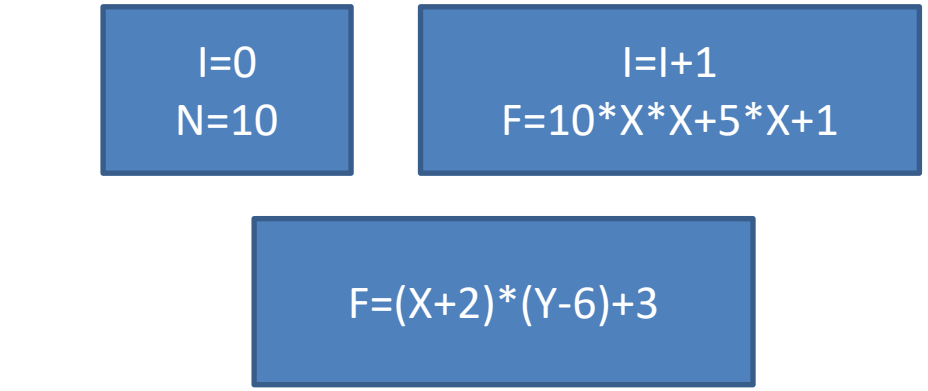
`A="pokusna hodnota"`	text (řetězec)
`B=5`	celé číslo
`C=10.458`	reálné číslo (číslo v pohyblivé řádové čárce)
`D=B+C`	
~~`E=A+B`~~	

Operace

Základní operace:

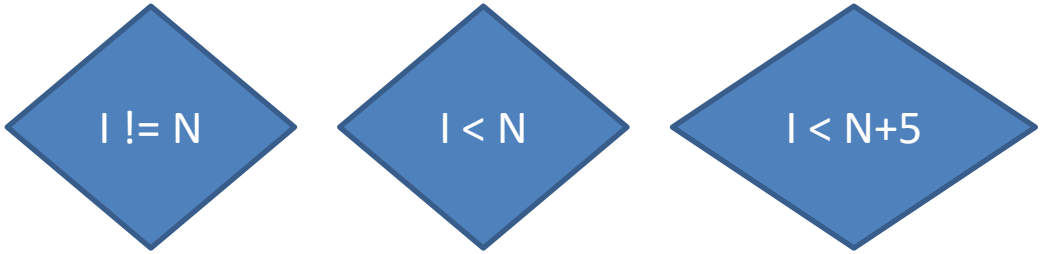
=	přiřazení
+	sčítání
-	odčítání
*	násobení
/	dělení

blok zpracování



Logické operace:

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



blok rozhodování

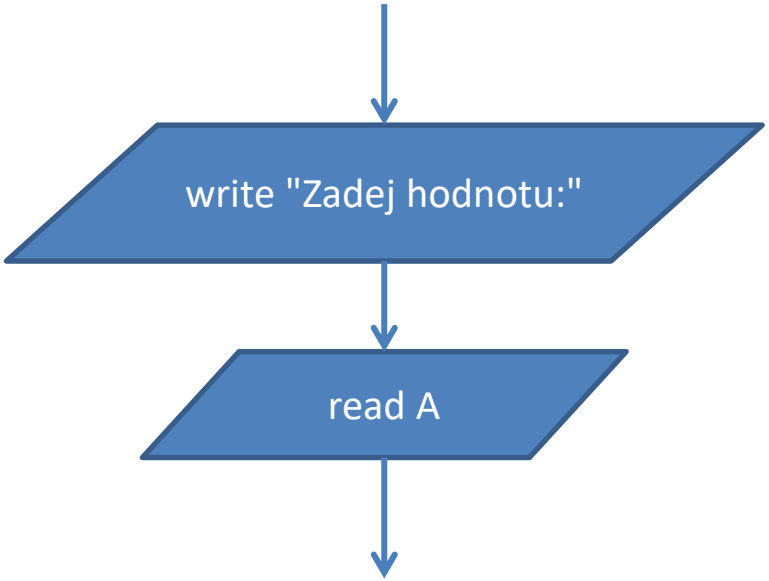
Vstup

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.

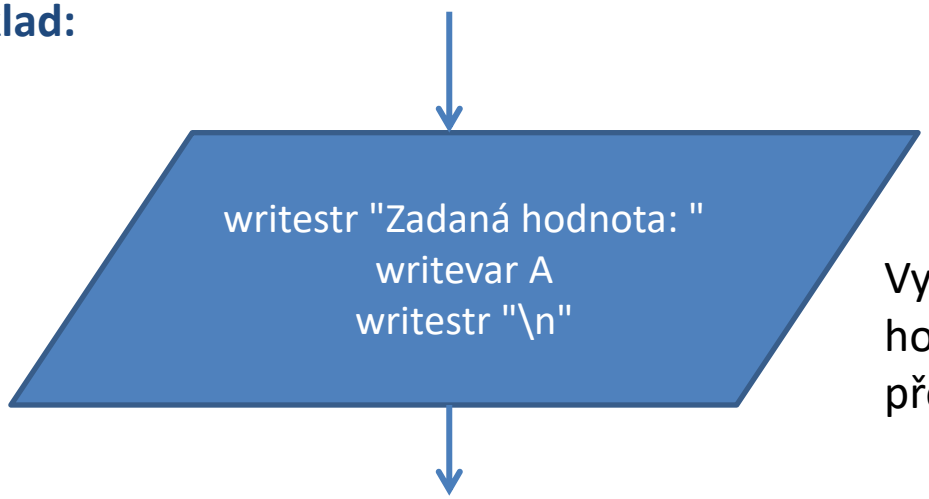
Výstup

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.

Domácí úkol

- **Algoritmizace**

Domácí úkol - pokyny

1. **Vytvořte vývojový diagram pro jedno z následujících zadání**. 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**, který je dostupný na klastru WOLF (popř. si jej můžete nainstalovat do VM s Ubuntu: $ sudo apt-get install dia)
3. Výsledné diagramy vložte do odevzdávárny **Algoritmus** v pdf formátu. Jméno souboru bude v následujícím formátu:

   PrijmeniL05X.pdf

   kde x je označení úkolu. Soubory, které budou pojmenovány jiným způsobem, nebudou kontrolovány (automatické změny jména provedené ISem jsou povoleny).
4. Termín pro odevzdání je **13. listopadu 2020 23:59**.

Doporučení: U cyklů používejte blok rozhodování před blokem zpracování.

Úkol A

Ve formě vývojového diagram popište algoritmus, který do výstupního zařízení typu terminál (tiskárna) vypíše 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
```

Úkol B

Ve formě vývojového diagram popište algoritmus, který do výstupního zařízení typu terminál (tiskárna) vypíše 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 C

Ve formě vývojového diagram popište algoritmus, který do výstupního zařízení typu terminál (tiskárna) vypíše 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
```