VŠB TECHNICKÁ UNIVERZITA OSTRAVA | HORNICKO GEOLOGICKÁ FAKULTA | KATEDRA GEOINFORMATIKY

Programování v GIS 1
3 - Struktury řízení chodu programu

Michal Kačmařík

A924, tel.: 5512

e-mail: michal.kacmarik@vsb.cz

https://www.hgf.vsb.cz/548/cs/

Řídící struktury

Představují konstrukce programu s určitým významem. Dělíme je na jednoduché a strukturované.

Mezi jednoduché příkazy řadíme:

- prázdný příkaz,
- volání funkce (print(“Hello world“))

https://www.w3schools.com/python/trypython.asp?filename=demo_default

Řídící struktury

- Strukturované příkazy:
  - Sekvence, posloupnost
  - Selekce
  - Cyklus

Sekvence

Sekvence je tvořena posloupností jednoho nebo více příkazů, které se provádějí v pevně daném pořadí. Určitý příkaz se začne provádět až po ukončení předchozího příkazu.

příkaz*1*
příkaz*2*
...
příkaz*n*

Selekce (podmínka)

Umožňuje podmínit provedení následujícího příkazu splněním zadaného výrazu

Splnění podmínky určí, který z příkazů bude vykonán v závislosti na splnění či nesplnění podmínky.

„Pokud klient zaplatil, pošli mu zboží. Jinak ho vyzvi k platbě zasláním sms.“

Lze použít

- úplnou, nebo
- neúplnou selekci.

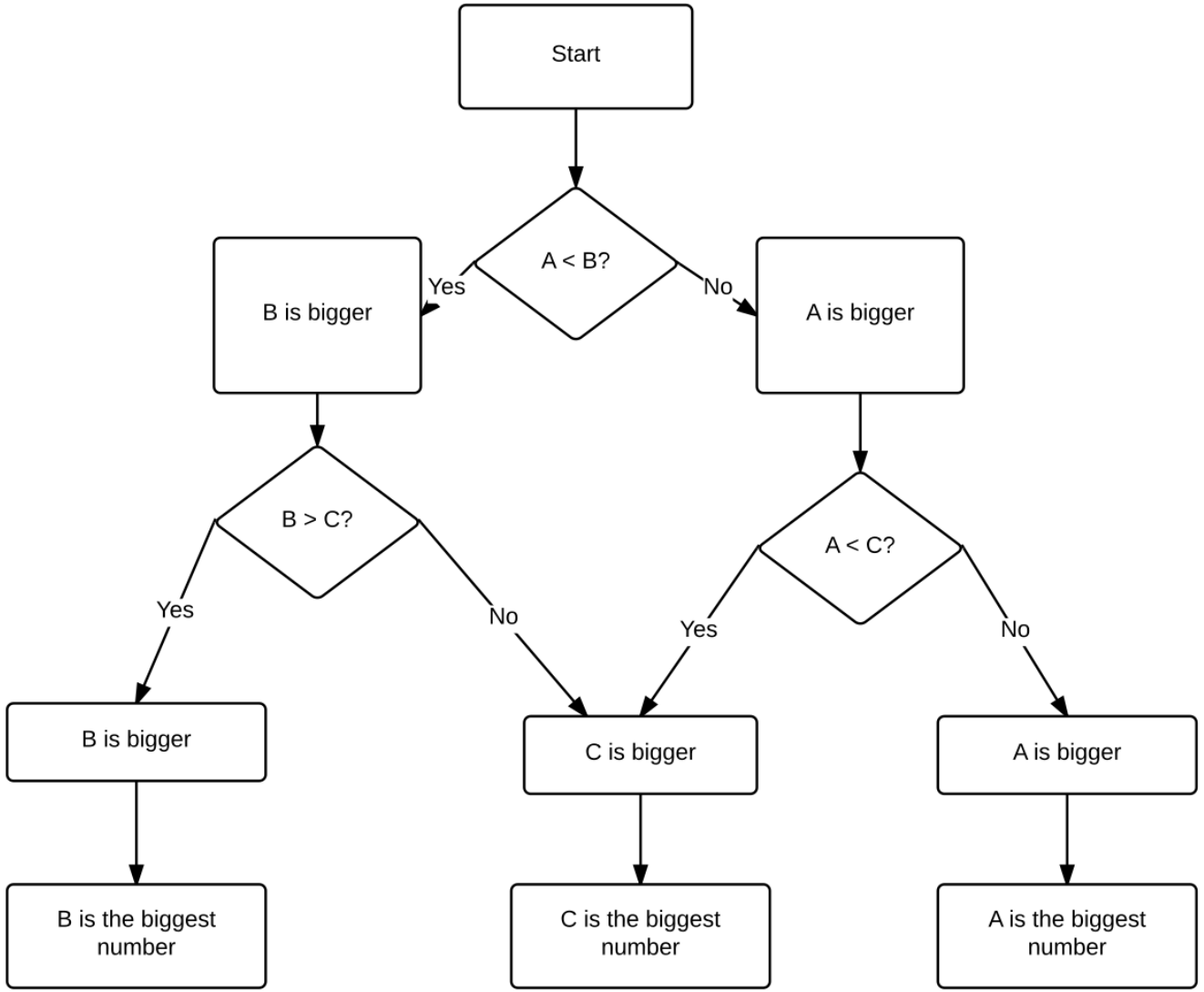
Neúplná selekce

```
if (podmínka)
{
 příkaz1
}
```

Úplná selekce

```
if (podmínka)
{
 příkaz1
}
else
{
 příkaz2
}
```

Použití selekce



Selekce v jazyce Python

Klíčová slova:

- ***if***: definuje začátek selekce, následuje logický výraz. POVINNÉ, vyskytuje se 1x
- ***elif***: řeší, co se má provést, pokud nebyla splněna předchozí podmínka. Následuje logický výraz. Nepovinné, může se vyskytnout vícekrát. Zkráceně „else if“.
- ***else***: řeší, co se má provést, pokud nebyla splněna žádná z předchozích podmínek. Nepovinné, může se vyskytnout jen 1x.

https://www.w3schools.com/python/python_conditions.asp

Selekce v jazyce Python

POZOR NA ODSAZOVÁNÍ ŘÁDKŮ UVNITŘ PODMÍNEK!

Příklad kódu pro nalezení vyššího ze dvou zadaných čísel:

```
x = 15
y = 10

if x > y:
    print (x)
elif x == y:
    print ("cisla jsou si rovna")
else:
    print (y)
```

Zkrácená varianta zápisu:

```
print(x) if x > y else print("cisla jsou si rovna") if x == y else print(y)
```

Selekce v jazyce Python

Logické operátory

Operátor	Význam	Příklad
==	rovná se	a == b
!=	nerovná se	a != b
>	větší než	a > b
<	menší než	a < b
>=	větší než, nebo rovná se	a >= b
<=	menší než, nebo rovná se	a <= b

Pozor: 1x rovná se (=) slouží k přiřazení hodnoty do proměnné, 2x rovná se za sebou (==) slouží k porovnání dvou proměnných

Selekce v jazyce Python

- Podmínku **if** je možno použít i k zjištění, zdali se nějaká hodnota či výraz (ne)nachází v zadané proměnné
- Klíčové slovo **in**, případně negace **not in**
- Příklady:

```
seznam = ["rohlik", "chleba", "houska", "koblizek"]
if "rohlik" in seznam:
    print("rohlik je k dispozici")
else:
    print("rohlik neni k dispozici")

jmeno = "Michal Kacmarik"
if "s" not in jmeno:
    print("Pismeno neni ve jmene")
```

Selekce v jazyce Python

- v jednom příkaze můžeme kombinovat více podmínek (jejich počet de facto není omezen)
- používají se klíčová slova (logické operátory):
  - **and**: má význam „a zároveň“, obě podmínky musí být splněny, aby platil celý výraz
  - **or**: má význam „nebo“, postačuje, aby jedna z podmínek byla splněna a bude platit celý výraz

```
if hmotnost > 36 or vyska > 150: #hmotnost v kg, vyska v cm
    print("nemusis pouzit autosedacku pro jizdu v aute")
```

Cyklus

Cyklus je část algoritmu, která je opakovaně prováděna za splnění řídící podmínky. Opakující se příkaz (sekvenci příkazů) nazýváme tělo cyklu.

Rozlišujeme dva typy cyklů:

- *indukční* – řídící podmínka cyklu určuje, zda bude provedena posloupnost příkazů, která tvoří tělo cyklu, nebo dojde k předání řízení za tělo cyklu,
- *iterační* – počet opakování těla cyklu závisí na hodnotě řídící proměnné.

Druhy cyklů

- Cyklus s podmínkou před vykonáním těla cyklu
- Cyklus s podmínkou za tělem cyklu
- Cyklus s pevným počtem opakování

Cyklus s podmínkou před vykonáním těla cyklu

U tohoto cyklu dochází k jeho ukončení v případě, že podmínka není splněna. Tělo cyklu se tedy nemusí vykonat ani jednou.

while (výraz s podmínkou)
{
příkaz
}

Cyklus while v Pythonu

POZOR NA ODSAZOVÁNÍ ŘÁDKŮ V TĚLE CYKLU!

Ukázka kódu pro vytištění celých čísel od 1 do 10:

```
cislo = 1
while cislo <= 10:
    print(cislo)
    cislo = cislo + 1
```

```
cislo = 1
while cislo <= 10:
    print(cislo)
    cislo += 1
```

Zavádíme řídící proměnnou (*cislo*) – udává, v jaké iteraci cyklu se zrovna nacházíme. V každé iteraci cyklu ji v uvedeném případě zvyšujeme hodnotu o +1.

Cyklus while v Pythonu

Zakomponování klíčového slova **else** do cyklu **while**:

```
cislo = 1
while cislo <= 10:
    print(cislo)
    cislo += 1
else:
    print("hodnota je vetsi nez 10")
```

Identicky lze **else** použít i v cyklu **for** (viz dále)

Cyklus s podmínkou za tělem cyklu

Tělo cyklu se provede minimálně jednou, protože k prvnímu testování podmínky dojde až po prvním průchodu tělem cyklu.

do
{
příkaz
}
while (výraz s podmínkou)

Jazyk Python nemá implementován cyklus do – while!
je možné ho však napodobit kombinací cyklu while a klíčových slov true, break

Cyklus s pevným počtem opakování

Cyklu nedefinujeme podmínku, ale přesně stanovený počet opakování.

```
for (výraz1; výraz2; výraz3)
{
příkaz
}
```

Cyklus for v Pythonu

POZOR NA ODSAZOVÁNÍ ŘÁDKŮ V TĚLE CYKLU!

Ideální pro procházení jednotlivých prvků v nějaké datové struktuře (textový řetězec, seznam, slovník, apod.)

Cyklus for v Pythonu

Ukázka kódu pro vytištění celých čísel od 0 do 10:

```
for i in range(0,11):
    print(i)
```

```
for i in range(11):
  print(i)
```

```
for i in range(0,11,1):
    print(i)
```

Zavádíme řídící proměnnou (*i*), do závorky specifikujeme její:

- **počáteční** hodnotu (pokud není zadána uživatelem, použije se 0)
- **koncovou** hodnotu (cyklus se zastaví jeden krok před jejím dovršením!)
- volitelně jako třetí parametr **inkrement**, o který se zvyšuje řídící proměnná v každé iteraci cyklu; pokud není zadána uživatelem, použije se 1

Cyklus for v Pythonu

Ukázka použití cyklu **for** pro postupné načítání a tištění hodnot ze seznamu

seznam = [“rohlik“, “chleba“, “houska“, “koblizek“]

kontrolní otázka: co vrátí příkaz *print(seznam)*?

Cyklus for v Pythonu

Klasická forma – použiji pomocnou proměnnou *i*, v každé iteraci proto s jistotou vím, kde v seznamu se aktuálně nacházím (na prvku s jakým indexem)

```
for i in range(0,len(seznam)):
    print(seznam[i])
```

Zkrácená forma – nemám přímou informaci o tom, na jakém místě seznamu se aktuálně nacházím, ale nemusím řešit pomocnou proměnnou

```
for prvek in seznam:
    print prvek
```

Kontrolní otázka: co bude výsledkem cyklu zadaného příkazy

```
for prvek in seznam[0:3]:
    print prvek
```

Break, continue

Klíčová slova využitelná v Pythonu v cyklech **while** i **for**:

- **break**: ukončí procházení cyklem i když podmínka je stále splněna
- **continue**: ukončí aktuální iteraci cyklu a přesune se do další iterace (zbytek těla cyklu za příkazem continue se v dané iteraci tedy již neprovede)

- Příklad **break**: cyklus prochází písmena abecedy a zastaví se před zadaným písmenem (k) – vytisknou se pouze písmena a až j

```
abeceda = "abcdefghijklmnopqrstuvwxyz"
for pismeno in abeceda:
    if pismeno == "k":
        break
    print(pismeno)
```

Break, continue

- Příklad **continue**: cyklus prochází písmena abecedy a přeskočí příkaz pro vytištění zadaného písmene (k) – písmeno k se tedy jako jediné ze všech nevytiskne

```
abeceda = "abcdefghijklmnopqrstuvwxyz"
for pismeno in abeceda:
    if pismeno == "k":
        continue
    print(pismeno)
```

Zásady pro řízení cyklů

- Před zahájením cyklu musí řídící proměnné nabývat smysluplných hodnot, umožňujících jeho ukončení.
- Tělo indukčního cyklu musí zajistit změnu řídících proměnných cyklu.

Vnořené cykly

- Jeden cyklus běží uvnitř jiného cyklu
- Často používané
- Například potřebujeme postupně projít všechny buňky rastru
- Ukázka v jazyce Python:

```
for radek in rastr:
    for bunka in radek:
        print(bunka)
```

```
for x in range(0, len(rastr)):
    for y in range (0, len(rastr[x])):
        print(rastr[x][y])
```

Kontrolní otázka: jakým způsobem (s využitím jakých datových struktur) by musel být zapsán rastr, aby uvedené ukázky kódu fungovaly a dávaly požadovaný výstup?

Python – funkce iter()

- Python má vestavěnu funkci **iter()**, která umožňuje postupně procházet prvky v seznamu (i ve variantě tuple), nebo ve slovníku
- Může být alternativou k využití cyklu **for**
- Příklad:

```
seznam = ["rohlik", "chleba", "houska", "koblizek"]
iteruj_seznam = iter(seznam)

#dalsi prvek v poradi volame prikazem next()
print(next(iteruj_seznam))
print(next(iteruj_seznam))
```

Děkuji za pozornost

Michal Kačmařík

michal.kacmarik@vsb.cz

www.vsb.cz