VŠB TECHNICKÁ UNIVERZITA OSTRAVA | HORNICKO GEOLOGICKÁ FAKULTA | KATEDRA GEOINFORMATIKY

Programování v GIS 1
1 - Algoritmus, programování

Michal Kačmařík

A924, tel.: 5512

e-mail: michal.kacmarik@vsb.cz

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

Obsah předmětu

- Cíl – získat základní znalosti a dovednosti z oblasti programování se zaměřením na úlohy nad prostorovými daty
- Rozsah 2+2
- Ukončení předmětu – zápočet + zkouška

Obsah předmětu

1, Algoritmus, programování, přehled programovacích jazyků, Python, kompilace

2, Datové typy a struktury

3, Struktury řízení chodu programu

4, Procvičování struktur řízení chodu programu

5, Práce s textovými a binárními soubory

6, Grafický zápis algoritmu (vývojové diagramy)

7, Algoritmy pro třízení a vyhledávání

8, Algoritmy nad vektorovými daty 1

9, Algoritmy nad vektorovými daty 2

10, Algoritmy nad rastrovými daty

11, Algoritmy v grafu, vyhledávání cesty

Požadavky na zápočet

- Vyhrazený počet bodů: 33
- Minimum je 17 bodů
- Rozdělení dle požadavků vedoucího cvičení

Požadavky na zkoušku

- Vyhrazený počet bodů: 67
- Minimum je 51 bodů při součtu bodů za zápočet a obě části zkoušky
- Písemná zkouška (max. 50 bodů) je psána v papírové formě, nejedná se o test s předdefinovanými odpověďmi. Nutné získat alespoň 26 bodů
- Ústní zkouška (max. 17 bodů)

Algoritmus

Algoritmem můžeme označit předpis, kterým popíšeme postup řešení nějaké úlohy či problému.

Nemusí být vůbec svázán s vytvořením počítačového programu

Potřebujeme:

- hodnoty vstupních dat,
- předepsané kroky řešení,
- požadovaný výsledek.

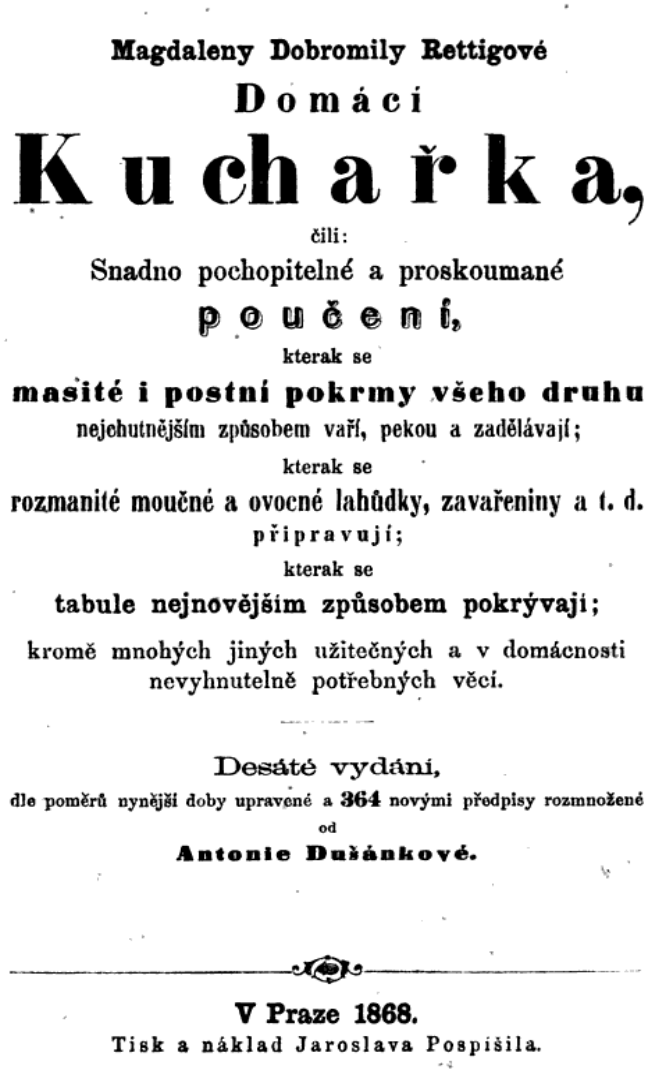
Vlastnosti algoritmu

- *Konečnost* – konečný počet kroků, požadovaný výsledek v rozumném čase, algoritmus se nesmí zacyklit, musí být jednoznačné, jak se dostane k cíli.
- *Determinovanost* – jednoznačný, přesný a srozumitelný, v každém okamžiku je jasné, co se má dělat a jaký bude následující krok.
- *Rezultativnost (korektnost)* – vrátí správný výsledek.
- *Hromadnost* – řešení skupiny podobných úloh (například algoritmus pro výpočet plochy polygonu bude fungovat pro různě veliké polygony, obsahující různý počet vrcholů, apod.).
- *Opakovatelnost* – při opakovaném spuštění se stejnými vstupními daty vrátí totožný výsledek.

Zápis algoritmu

- Pomocí přirozeného jazyka (slovní popis),
- pomocí grafického znázornění (např. vývojový diagram),
- pomocí speciálního jazyka (pseudojazyk),
- pomocí programovacího jazyka.

Zápis pomocí přirozeného jazyka



Magdaleny Dobromily Rettigové

Domácí

Kuchařka,

čili:

Snadno pochopitelné a proskoumané

poučení,

kterak se

masité i postní pokrmy všeho druhu

nejchutnějším způsobem vaří, pekou a zadělávají;

kterak se

rozmanité moučné a ovocné lahůdky, zavařeniny a t. d.

připravují;

kterak se

tabule nejnovějším způsobem pokrývají;

kromě mnohých jiných užitečných a v domácnosti nevyhnutelně potřebných věcí.

Desáté vydání,

dle poměrů nynější doby upravené a **364** novými předpisy rozmnožené

od

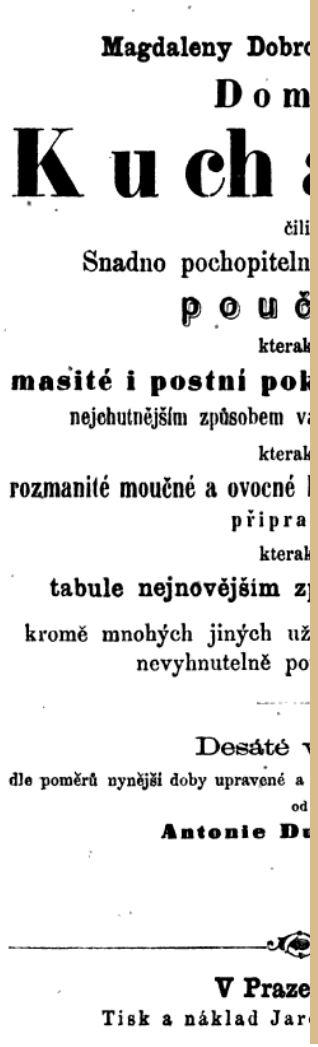
Antonie Dušánkové.

V Praze 1868.

Tisk a náklad Jaroslava Pospíšila.

http://www.scanzen.cz/knihy/domaci-kucharka

Zápis pomocí přirozeného jazyka



Magdaleny Dobr

D o m

K u ch

čili

Snadno pochopiteln

p o u č

kterak

masité i postní pok

nejchutnějším způsobem v

kterak

rozmanité moučné a ovocné

připra

kterak

tabule nejnovějším z

kromě mnohých jiných už

nevyhnutelně po

Desáté

dle poměrů nynější doby upravené a

od

Antonie Du

V Praze

Tisk a náklad Jar

30. Polívka se zemčaty.

Uvař ne příliš do měkka malá zemčata, oloupej a rozkrájej je na lístky, pak jich dej vrstvu na hluboký talíř, posyp je usekanou zeleninou, na to dej opět vrstvu zemčátek, pak opět uzeninu, a tak pokračuj, až je všecko vyrovnáno; navrch musí přijíti zemčátka; nyní to polej několika lžícemi dobré, tučné hovězí polívky, postav talíř na třínožku do trouby a nech to povrchu do zlatova upéci. Několik ovařených zemčátek oloupej a ustrouhej, dej na to procezené hovězí polívky, přidej k tomu trochu drobně usekané zelené petruželky a nech to chvíli vařit; potom vlej polívku do mísy a talíř s těmi opečenými zemčaty dej na stůl zvlášť, aby si mohl každý dle libosti posloužit.

31. Polívka se zemčaty na jiný způsob.

Pro 6 osob oloupej 12 prostředních zemčat, rozkrájej je na lístky a nech je buď na rozkrájeném hovězím loji nebo na čerstvém másle s rozkrájenou cibulkou hezky do zlatova usmažit; potom na ně nalej -½" litru dobré hovězí polívky a nech je do měkka rozvařit, pak je rozmačkej, rozmíchej, a proceď. Olouvej zase 12 malých, pěkných, stejných, kulatých zemčďt, nakrájej z nich na stéblo tenké koláčky, a opět je buď na loji nebo na novém másle do zlatova vysmaž, pak je dej do polívkové mísy, polívku okořeň květem a vlej ji na zemčata. Kdo tuto polívku trochu přihoustlejší míti chce, může zemčata před rozvařením ještě dvěma

Programování

Zápisem algoritmu do vybraného programovacího jazyka implementujeme algoritmus a vytváříme program

Program = sekvence příkazů (instrukcí), kterým počítač rozumí a má je vykonat k vyřešení daného problému

Základní pojmy

Datový typ určuje

- hodnoty, kterých může nabývat (datový) objekt,
- množinu přípustných operací nad tímto datovým typem.

- Číslo
- Znak
- Boolean
- …

Základní pojmy

Datovým objektem může být

- konstanta,
- proměnná,
- výraz,
- funkce.

Základní pojmy

- *Identifikátorem* označujeme **jméno**, které dáváme konstantám, proměnným, funkcím.
- Identifikátor je tvořen řetězcem znaků, kterými jsou písmena anglické abecedy, číslice, případně znak podtržítko.
- První znak řetězce je písmeno nebo znak podtržítko, pak může následovat libovolná sekvence písmen, číslic a znak podtržítko. Délka identifikátoru může být libovolná, je však obvykle omezena podle prostředí, ve kterém řešíme úlohu.
- _zemcata, Zemcata, zemcata

Základní pojmy

Konstanta je veličina, jejíž hodnota zůstává po dobu řešení problému stejná.
Může být použita dvěma způsoby:

- přímo: 63, 10^{-2}, ’ABC’ nebo
- pojmenováním – označena identifikátorem, (řecké písmeno PI používáme jako jméno konstanty 3,14…)

potato:=4, PI:=3.14

Základní pojmy

Proměnná – označujeme tímto pojmem takovou veličinu, která může měnit hodnotu během řešení problému.

Proměnná se zavádí definicí:

- pojmenováním proměnné a
- určením jejího datového typu.

```
int potato;           // deklarace proměnné potato
int strana, stred;    // deklarace proměnných strana, stred
float  pi=3.1428;     // deklarace a přiřazení hodnoty proměnné pi
```

Základní pojmy

Výraz je tvořen operátory, operandy a speciálními znaky.

Operandem může být:

- konstanta,
- proměnná,
- výraz,
- a volání funkce.

Základní pojmy

Příkazem (může být označen pojmem řídící struktura) rozumíme jednotlivé kroky algoritmu a návaznosti mezi nimi. Rozlišujeme jednoduché a strukturované příkazy.

Celý algoritmus lze chápat jako jeden příkaz.

Základní pojmy

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

příkaz*1*
příkaz*2*
...
příkaz*n*

Programovací jazyky

- Počítače (všechna digitální zařízení) umí vykonávat pouze velmi jednoduché operace (základní aritmetické a logické operace, čtení, psaní, zapisovaní)
- Všechny složitější operace (zobrazení webové stránky, zapsání textu do dokumentu, zobrazení fotografie) jsou proto tvořeny kombinací velkého množství jednoduchých instrukcí
- Počítač nerozumí přirozenému jazyku člověka („Zobraz fotografii“)
- Pro člověka by bylo nadmíru obtížné vytvářet programy pouze s použitím jednoduchých operací vykonatelných počítačem
- Proto vznikly programovací jazyky

Programovací jazyky

- Programovací jazyky umožňují implementaci algoritmů v lidem srozumitelné podobě, kterou je možné **překladačem (compiler)** přeložit do podoby programu (sekvence instrukcí) srozumitelného a zpracovatelného počítačem

Programovací jazyky

1. Programovací jazyky vyžadující kompilaci
   - zdrojový kód musíme přeložit překladačem, vznikne samostatně spustitelný program (např. pro Windows *.exe)
   - kompilace probíhá pro celý zdrojový kód najednou
   - např. jazyk C, C++
   - výhody = rychlost běhu programu, snadné odhalení chyb ve zdrojovém kódu (kompilace neproběhne při chybě v programu), z pohledu tvůrce = distribuuje se výsledný program ve strojovém kódu, ne samotný zdrojový kód
   - nevýhody = závislost na platformě (zdrojový kód musí být samostatně kompilován pro každou vybranou platformu – Windows, Linux, Mac, Android, …), z pohledu uživatele = nemožnost editovat zdrojový kód

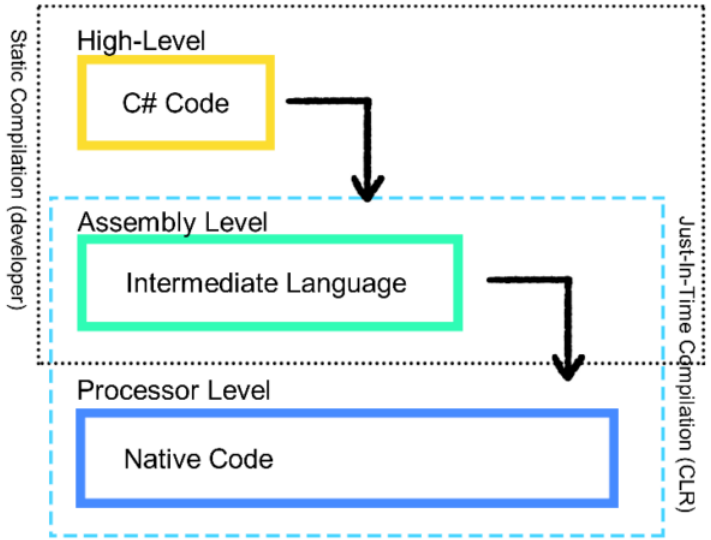
Programovací jazyky

2. Programovací jazyky vyžadující interpreta = skriptovací jazyky
   - pro spuštění zdrojového kódu (skriptu) je potřeba interpreter daného jazyka, kód se nepřekládá do podoby samostatně spustitelného programu
   - interpreter taktéž překládá zdrojový kód do strojového kódu, ale činí to postupně, po jednotlivých řádcích (řádek je přečten, přeložen, spuštěn, …)
   - tyto jazyky se typicky využívají ve webových aplikacích, při tvorbě rozšíření (pluginů) a automatizaci práce v existujících software, administraci systémů
   - např. Python, Perl, PHP, JavaScript, bash, Ruby
   - výhody = přenositelnost zdrojového kódu mezi platformami (stačí, aby existoval pro danou platformu interpret), jednoduchý vývoj, stabilita, jednoduchá editace řešení
   - nevýhody = při zpracování stejné úlohy jsou typicky pomalejší než programovací jazyky vyžadující kompilaci, obtížnější hledání chyb v kódu

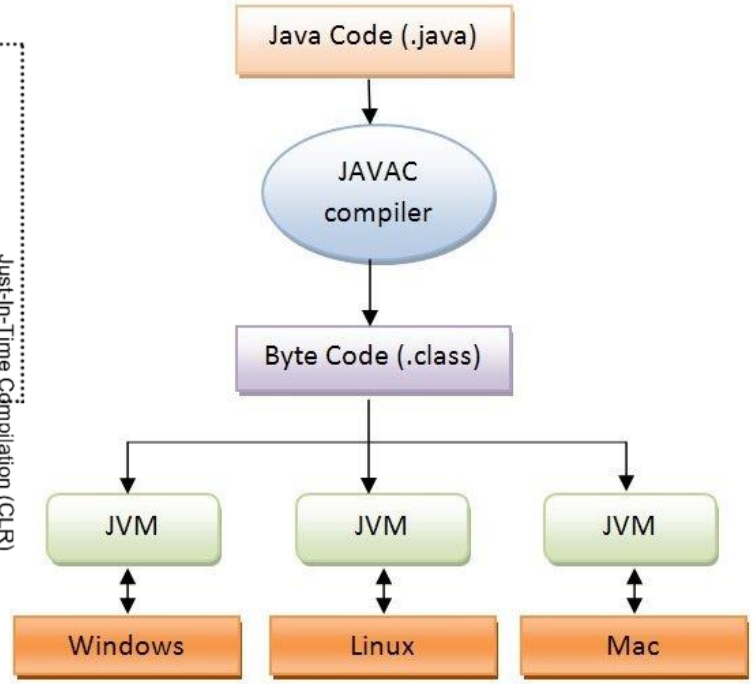
Programovací jazyky

3. Programovací jazyky využívající virtuální stroj

- kombinace přístupu 1. a 2.
- programovací jazyk využívá tzv. virtuální stroj pro zajištění možnosti využití vytvořeného programu na různých platformách a spojení výhod přístupu 1. a 2.
- zdrojový kód je nejprve přeložen do tzv. mezikódu nazývaného CIL (Common Intermediate Language)
- ten je následně díky své jednoduchosti relativně rychle interpretovatelný tzv. virtuálním strojem (tedy interpretem)
- např. Java, C#



https://freecontent.manning.com/how-is-c-compiled/



Programovací jazyky

- Značkovací jazyky (markup languages) – nejsou programovacími jazyky, slouží primárně k vytváření struktury pro ukládání dat či k definici vizuální podoby dat při jejich prezentaci
  - nepřevádí žádný algoritmus do podoby programu
  - např.: XML, HTML

Vybrané programovací jazyky

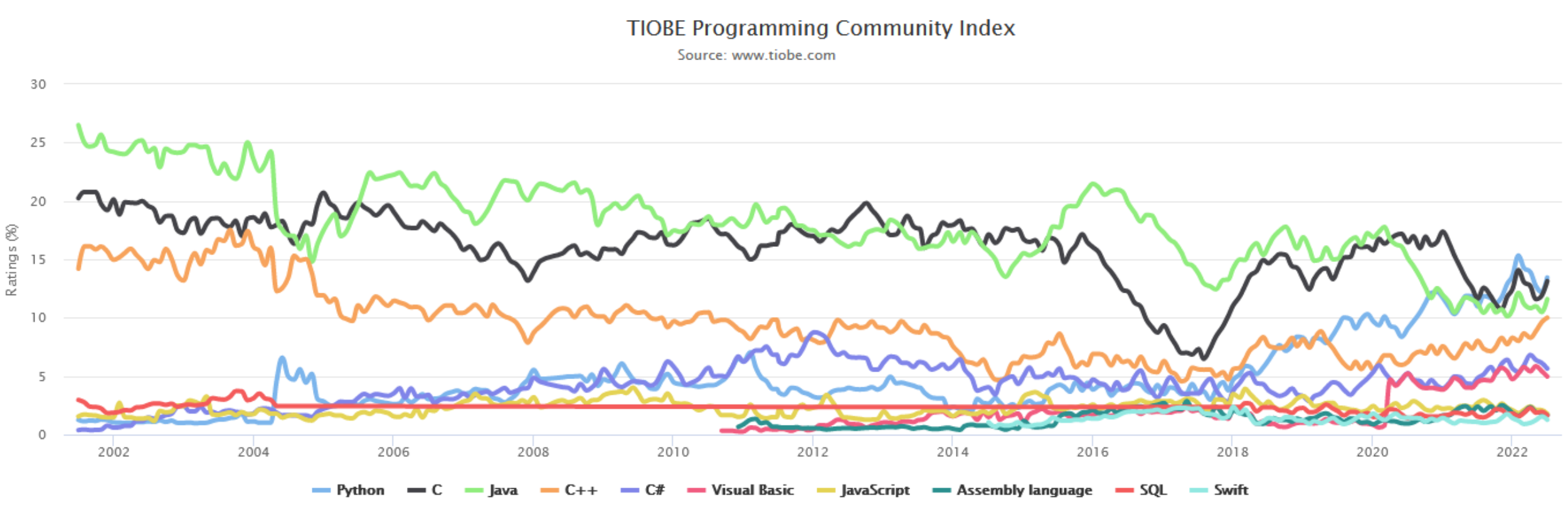
- **FORTRAN**: vyvíjen 1954 až 1963, kladen důraz na rychlost a efektivnost zejména vědecko-technických výpočtů, nevýhodou nepřehlednost kódu, ve své době velmi rozšířený, dnes omezeně využíván ve vědeckém prostředí
- **C**: vytvořen 1972, univerzální procedurální jazyk (dodnes tvoří základ celé řady software včetně operačních systémů a ovládání různého hardware), omezené množství vestavěných funkcí, ovlivnil velké množství následně vzniklých jazyků
- **C++**: představen 1983, vznikl rozšířením jazyka C o objektově-orientovaný přístup a další funkcionalitu
- **Java**: představen 1995, objektově-orientovaný jazyk, univerzální jazyk – vytvořené řešení je nezávislé na platformě díky použití virtuálního stroje (viz předcházející snímek), aktuálně stále velmi rozšířený, syntaxe do značné míry odvozena z C++

Vybrané programovací jazyky

- **C#** (čti „sí šárp“): zveřejněn Microsoftem v roce 2002 společně s platformou .NET (ta poskytuje širokou škálu prostředků pro programy), objektově-orientovaný jazyk, založen na C++ a Java, primárně pro platformu Windows, ale využívá virtuálního stroje k umožnění nasazení i na jiných platformách
- **Python**: zveřejněn 1991, skriptovací objektově-orientovaný jazyk, jeho popularita prudce roste v posledních cca 5 letech, nezávislý na platformě, open-source
- **PHP**: zveřejněn 1995, skriptovací jazyk zejména pro webové aplikace, open-source
- **JavaScript**: zveřejněn 1995, skriptovací jazyk zejména pro webové aplikace (včetně publikování prostorových dat na webu)

Vybrané programovací jazyky

TIOBE = index popularity programovacích jazyků



https://www.tiobe.com/tiobe-index/

Proč Python?

- univerzálnost
- celosvětová rozšířenost a popularita
- nepřeberné množství (volně) dostupných modulů
- velmi dobrá komunitní podpora
- primární skriptovací jazyk pro řadu software v geoinformatice (ArcGIS, QGIS)
- jednoduchá syntaxe
- https://www.w3schools.com/python/default.asp

Syntaxe programovacího jazyka

- Každý (*programovací*) jazyk má svá **pravidla**, která se musí dodržovat, aby bylo možné vytvořenému textu rozumět (*aby mohl počítač zdrojový kód zpracovat*)

C++

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string name;
    cin >> name;
    cout << "Hello, " << name << endl;
    return 0;
}
```

Python

```python
name = input()
print("Hello, " + name)
```

Komentáře

- komentáře jsou části zdrojového kódu, které nejsou zpracovávány při běhu programu
- slouží k:
  - K popisu části kódu, vysvětlení jeho účelu a principu fungování
  - Ke zlepšení orientace v celém kódu
  - K dočasnému odstranění či zablokování části kódu (např. při hledání chyby v části kódů)
  - K zachování původní verze části zdrojového kódu
- je dobrou praxí a v některých případech povinností naučit se své zdrojové kódy dobře komentovat (usnadní to pochopení kódu druhou osobou, ale i námi samotnými když se k němu třeba po delší době vrátíme k dalšímu úpravám).

Komentáře

```
#-----------------------------------------------------------------------------
#function for meteo RINEX files conversion to own format, converts all files in given folder
#structure of own MET format = station_name, date, time, pressure, temp
def fMeteoRINEX_myMETEO(Met_RIN,Met_MET):
    print '-------------------------------------------------------------------------------------------------'
    print "conversion of meteo RINEX files to own MET format was started - function "+fMeteoRINEX_myMETEO.__name__
    print '-------------------------------------------------------------------------------------------------'

    flag_error = 0
    if len(os.listdir(Met_RIN)) > 0:
        for file in os.listdir(Met_RIN):
            # read content of file
            f = open(Met_RIN+'/'+file, 'r',)
            myLines = f.readlines()
            f.close()

            myMeteoParam=[]
            soubor=[]

            for myLine in myLines:
                # find station name
                if not re.search('MARKER NAME', myLine) == None:
                  myLine = myLine.strip('\n')
                  myMarkerName = (re.split('\s+',myLine.strip(' '))[0][0:4]).lower()
                  #print myMarkerName

                # find type of included meteo data and their order on data lines
                if not re.search('TYPES OF OBSERV', myLine) == None:
                  myLine = myLine.strip('\n')
                  myHeaderLine = re.split('\s+',myLine.strip(' '))
```

Komentáře v Pythonu = použití znaku #

Děkuji za pozornost

Michal Kačmařík

michal.kacmarik@vsb.cz

www.vsb.cz