Labyrint programovacích jazyků

a ráj programátora

- Programovacích jazyků je mnoho a mají různé vlastnoti.
- Známe Python a C#, dnes si ukážeme další
- a všimneme si, že jim vlastně rozumíme.
- Chceme-li, aby nás poslouchaly báječné počítací stroje kolem nás, musíme rozumět mnoha jazykům.

Jazyky

najdeme třeba v ReCodExu

- Pascal,
- C, C++,
- Java,
- Javascript,
- Kotlin,
- Swift.
- Ukážeme si zpravidla podíl dvou čísel.

Pascal

osvědčený výukový jazyk u nás již nepoužívaný

```
procedure vydel(co,cim:integer);
begin
    if cim=0 then writeln('NELZE');
    else    writeln(co div cim);
end;
var prvni,druha:integer;
begin
    read(prvni);
    read(druha);
    vydel(prvni,druha);
end.
```

Pascal

a jeho vlastnosti

- Proměnné se napřed definují, potom použijí.
- Definuje se v sekci `var`, pořadí je obrácené oproti C# (napřed identifikátor, pak za dvojtečkou typ).
- Procedury jsou ideově oddělené od funkcí,
- mírně odlišné operatory (= pro porovnání, div pro celočíselné dělení),
- bloky se označují slovy `begin` a `end` namísto závorek či indentace.

C

ajajaj

```
#include <stdio.h>
#include <stdlib.h>
void vydel(int co, int cim)
{   if(!cim)    printf("NELZE\n");
    else    printf("%d\n",co/cim); }
int main(int argc, char*argv[])
{   char pom[100];
    gets(pom);
    int jeden=atoi(pom),druhej;
    gets(pom);
    druhej=atoi(pom);
    vydel(jeden,druhej);
    return 0; }
```

C

a poučení z něj

- Jazyk nízké úrovně,
- měli byste znát z Počítačových systémů (Arduino se programuje v podmnožině C++).
- Vychází z Pascalu, zakladatel dynastie, do které patří i C#.
- Nemá typ `bool` ani `string`. Boolean je reprezentován integerem (0 je false, cokoliv jiného true), string je reprezentován ukazatelem (pointerem) na typ char. Ukazuje se tam, kde string začíná.
- Odtud pochází reprezentace stringu jako pole znaků.
- Nemá garbage collector (zato má pointery), o paměť je nutné starat se ručně (allokovat/deallokovat).
- V příkladu využíváme typovou konverzi mezi polem charů a pointerem na char (reprezentujícím string).

C++

lze užívat stejně jako C, nebo také ne

```
#include <stdio.h>
using namespace std;
void vydel(int co, int cim)
{   if(!cim)    printf("NELZE\n");
    else    printf("%d\n",co/cim); }
int main(int argc, char*argv[])
{   int jeden,druhej;
    cin>>jeden;cin>>druhej;
    vydel(jeden,druhej);
    return 0;
}
```

C++

a jeho vlastnosti

- Lze v něj programovat podobně jako v C,
- oproti C obsahuje i objekty ovládané podobně jako v C# (s násobnou dědičností, místo generických typů obecnější šablony, přetěžování operátorů,...).
- V příkladu k načítání vstupu využíváme přetížené operátory bitového posunu (přetížené pro streamy).
- Pozor, jazyk C není objektový, objekty jsou až v C++.
- Fungují dva systémy allokace paměti (malloc/free z C a objektové new/delete – volá konstruktory a destruktory).

Java

příklad

```
import java.util.Scanner;
class x
{   public static void main(String[]args)
    {   Scanner s=new Scanner(System.in);
        int a=s.nextInt(),b=s.nextInt();
        if(b==0)
            System.out.println("NELZE");
        else    System.out.println(a/b);
    }
}
```

Java

mravní naučení

- Všiml si někdo, že to není C#?
- Rozdílů je velmi málo, neříká se `namespace`, String se píše s jiným s, vstup je řešen Scannerem, místo using říkáme import.
- Java nemá pointery (ani zamaskované), tudíž neexistují delegaty. Ovladače se řeší předáním objektu implementujícího správný interface. Tomu voláme příslušnou funkci jako ovladač.
- Správa paměti funguje jako v C# (garbage-collector),
- jazyk je povinně objektový.

Javascript

je divný jazyk

```
a=prompt("Dej cislo");
b=prompt("Dej druhe");
if(b==0) alert("NELZE!");
else     alert(Math.trunc(a/b));
```

Javascript

tady bude naučení mnoho

- Latentně objektový jazyk (vše je objekt, ale rituály nejsou vyžadovány).
- Proměnné se definují podobně jako v Pythonu, nedefinovaná proměnná formálně existuje a je typu undefined.
- Množství typových konverzí: "1"+1==11 (konverze do stringu), 1+1==2, "1"*1==1, "1"+1+1==111, 1+1+"1"=="21"...
- Do objektů se přistupuje operátorem tečky (jak jsme zvyklí).
- Paměť spravována garbage-collectorem.
- Každá funkce může být konstruktorem třídy: `a=new f();`...
- ... prvky objektu budou lokální proměnné funkce `f`.

Kotlin

budoucnost programování dle společnosti Google

```
fun main(args:  Array<String>){
    var a=readLine()
    var b=readLine()
    if(b!!.toInt()==0)
        println("NELZE")
    else    println((a!!.toInt()/b!!.toInt()).toInt()!!);
}
```

Kotlin

se podobá Swiftu kryptickými operátory otazníku a vikřičníku

- Jazyk se zjevně podobá Pythonu.
- Tam, kde může být objekt neinicializován, je třeba použít operátor otazníku (proveď dereferenci jen pokud je objekt nenullový) nebo dvou vikřičníků (proveď dereferenci v každém případě).
- Jazyk je zjevně objektový (viditelně voláme metody stringu).
- Syntax je nám stále intuitivně zřejmá.
- Někteří tvrdí, že Kotlin se podobá C#.

Swift

je jazyk navržený společností Apple

```
var a=Int(readLine())
var b=Int(readLine())
if(b==0)
{   print("NELZE")}
else { print(Int(a!/b!))}
```

Swift

jaký dojem na nás udělal

- Další jazyk podobný Pythonu – jako Kotlin.
- Za základními řídicími strukturami musí následovat blok (i když sestává z jednoho příkazu).
- Jazyk je také zjevně typový a proměnné se v něm definují podobně jako v Pascalu.
- Taktéž je osazen divnými operátory otazníku a vikřičníku (pro účely práce s neinicializovanými objekty).
- Tentokrát vikřičník stačí jeden! :-)

Závěr

z pohledu na labyrint jazyků

- Jazyků je mnoho a každý má nějaká specifika.
- Všechny jsou základním způsobem intuitivně srozumitelné,...
- ... od chvíle, kdy nás napadne, že do programovacího jazyku zapisujeme algoritmy, tedy záměrně navržené postupy směřující k vyřešení zadaného problému sestávající z jednotlivých kroků zvaných příkazy.
- Tyto příkazy jsou všude do značné míry stejné. Máme proměnné a práci s nimi, základní řídicí struktury, funkce, občas i objekty – a to je téměř vše.
- Při programování čteme vstup, vypisujeme výstup a zpracováváme (vstup na výstup). Vstup a výstup se mezi jazyky liší, zpracování ne.
- Přednášející nekecal, ve všech jazycích se programuje stejně.

Konec

zavíráme a jdeme domů – počkat, vždyť už jsme doma

Děkuji za pozornost a z pozice přednášejícího se tímto loučím.
Dotazy?
Mailem!