8. Program, programovací jazyky, příkaz, instrukce, druhy adresování, skoky

Program

Jedná se o posloupnost instrukcí, která popisuje realizaci úkolu počítačem.Zápisem algoritmu v programovacím jazyce vytváříme program. Zdrojový kód může být přeložen překladačem do strojového kódu, který je pak přímo vykonáván procesorem, nebo je zdrojový kód vykonáván interpretem.

Vývoj programování

1. Strojový kód

Jedná se program, jenž je souborem instrukcí, kde jsme nemáme žádnou možnost pojmenovávat proměnné nebo zadávat matematické výrazy.

2. Nestrukturované paradigma

Nestrukturovaný přístup je podobný assemblerům, jedná se o soubor instrukcí, který se vykonává odshora dolů. Zdrojový kód již nebyl závislý na hardwaru a byl lépe čitelný pro člověka, přístup na nějakou dobu umožnil vytvářet komplexnější programy.
Bylo tu však stále mnoho úskalí: Jediná možnost, jak udělat něco vícekrát nebo jak se v kódu větvit, byl příkaz GOTO .

3. Strukturované paradigma

Strukturované programování je první paradigma, které se udrželo delší dobu a opravdu chvíli postačovalo pro vývoj nových programů. Programujeme pomocí cyklů a větvení.

Program lze rozložit do funkcí. Problém se rozloží na několik podproblémů a každý podproblém potom řeší určitá funkce s parametry. Nevýhodou je, že funkce umí jen jednu činnost.
Neexistuje totiž způsob, jak vzít starý kód a jen trochu ho modifikovat, musíme psát znovu a znovu - vznikají zbytečné náklady a chyby. Tuto nevýhodu lze částečně obejít pomocí parametrizace funkcí nebo použitím globálních proměnných.

S globálními daty vzniká však nové nebezpečí - funkce mají přístup k datům ostatních.Celý program se potom skládá z nezapouzdřených bloků kódu a špatně se udržuje.

4. Objektově orientované programování

Jedná se o filozofii a způsob myšlení, designu a implementace, kde klademe důraz na znovupoužitel•nost. Přístup nalézá inspiraci v průmyslové revoluci - vynález základních komponent, které budeme dále využívat.

Poskládání programu z komponent je výhodnější a levnější. Můžeme mu věřit, je otestovaný , protože komponenty můžeme testovat odděleně. Pokud je někde chyba, stačí ji opravit na jednom místě. Jsme motivováni k psaní kódu přehledně a dobře, protože ho po nás používají druzí nebo my sami v dalších projektech (přiznejme si, že člověk je od přírody líný a kdyby nevěděl, že se jeho kód bude znovu využívat, nesnažil by se ho psát kvalitně).

Nejpoužívanější programovací jazyky

Python

Python je interpretovaný, vysoko úrovňový programovací jazyk, jenž je veřejně známý jakožto nejjednoduší a zároveň používaný i u velmi obtížných úloh.

Jeho filozofie si zadává za cíl jednoduchou orientaci v textu v koleraci s významným odsazováním.

Využívá objektově orientovaný přístup, aby pomáhal programátorům psát čistý, jednoduchý,logický kód na psaní malých a velkých programů.
Python program probíhá dynamicky a obsahuje garbage collector.

Jelikož je snad s těcho jazyků Python nejvýše, vyplňuje nesmyslně moc času zpracováváním jednoho primitivního příkazu a to se negativně jeví na chodu programu.

Javascript

S počátky už od roku 1995 jako skriptovací jazyk pro jeden z prvních webových prohlížečů Mosaic, se datuje Javascript.
Jedná se také interpetovaný jazyk, jenž je používaný na webových stránkách na stylizaci a efekty stránky.Jedná o tzv. Frontend.

Javascript je také možno používat velice dobře na backend zvaný Node.js na kterém je možné stavět webový server, REST API, atd.
Javascript potažmo Node.js klade důraz už sám na více vláknové programování a proto je rychlejší než python ve všech ohledech.

Většina prohlížečů má svůj engine na zpracování Javascriptu.

Java

Programovací jazyk, který si zakládá na třídách, a objektově-orientovanému programování.

Vývíjen společností Oracle, vytvořen ve stejnou dobu jako Javascript.
S myšlenkou napsat jednou program a umožnit jeho běh na jakémkoliv zařízení.
Zkompilovaný kód může běžet na jaké koliv platformě, která Javu podporuje bez jakékoliv rekompilace.
Kód býva typicky v bytekódu a ten poběží na jakémkoliv Java virtual machine s irelevantostí na jaké počítačové architektůře poběží.

Java runtime umožňuje dynamicky odkazovat a upravovat běžící kód.
Dle Githubu byla Java za rok 2019 nejvíce používaný jazyk.

C++

S úmyslem vytvořit dodatek k C vznikl jazyk C++.
Moderní C++ nabízí OOP, generika a funkcové možnosti, ale stále nabízí výhody C,jakožto manipulaci s pamětí a uklízení si paměti.

C++ je široce používaný jazyk a je s ním psáno ve všech odvětvích.
Hry, serverové aplikace, desktopové aplikace, a výkonostní-kritické aplikace.Nevyužívá garbage collector a je kompilován kompilátorem.

Je hojně používán u mikrořadičů jako STM32 a Arduino, které k tomu využívá knihovnu Wiring.

C

Jazyk C je nejnižší jazyk s čitelností dnešních programovacích jazyků.Vyvinut v roce 1972 Dennisem Ritchie, jenž je také otcem Unixu, se tento jazyk stále používá u Linuxu, kernelů, a mikrokontrolérů.

Je využíván jak zařízeními se 8bit architekturou tak Superpočítači.
Napsat program v C je dosti složíté, jelikož musí programátor udělat většinu práce sám.V C je napsán Linux díky Linusu Torvaldosi.

Tento jazyk má také spousty kompilátorů nabízených např. IBM, Microsoft atd.Tento jazyk nemá konkurenci v exekučním čase.
Na algoritmizaci a zpracování dat neexistuje lepší konkurence, pokud si můžeme dovolit strávit většinu času psaním programu.

Proces

Jedná se o instanci spuštěného programu.Proces je umístěn v operační paměti počítače v podobě sledu strojových instrukcí vykonávaných procesorem.

Proces obsahuje nejen kód vykonávaného programu, ale i dynamicky se měnící se data, která proces zpracovává.

Moderní operační systémy umožňují spustit více procesů, což nazýváme multitasking - multiprocessing. Při přepínání procesů je důležité uchovat všechny informace potřebné pro opětovné obnovení procesu v místě a ve stavu, kde bylo jeho vykonávání přerušeno tak, aby běžící proces tuto změnu nepoznal.

Tyto informace se pro každý blok ukládají do tabulky PCB (Process Control Block), která je spravována jádrem operačního systému.

Změna kontextu je poměrně náročná operace, a proto byla zavedena vlákna, která mají režii na přepnutí nižší.Uspoří se zejména na tom, že thready sdílejí jeden paměťový prostor. To jim též umožňuje velmi rychlou a efektivní vzájemnou komunikaci.

Řízení procesoru pomocí programu

Každý jazyk nám umožní do jiné hloubky fungovat v operačním systému.
V C/C++ se dostaneme k paměti a můžeme na ni odkazovat, ale musíme uklízet po sobě.U vyšších jazyků máme buď interpreter nebo kompilátor.V kompilovaných jazycích už musí vše v čase kompilace být napsané.

U mikrořadičů se dostaneme přímo ke kódu v jazyce C kde určuje co na jaký registr půjde.Jsme schopni definovat každou operaci a optimalizovat ji.

Programové konstrukce

```java
package test;

public class Main{

    public static void main(String[] args){
        System.out.println("Hello, World!");
    }
}
```

package

nám deklaruje namespace ve kterém jsou třídy uloženy.Package nám organizuje třídy založené na jejich funkcionalitě.

public class Main

public znamená veřejný přístup k této třídě.Říká to kompilátoru jak k ní má přistupovat.
class je na deklaraci třídy a poté název.

public static void main(String[] args)

Jedná se o funkci pojmenovanou main s polem stringů jako vstupní parametry.
Každý Java program začíná v nějaké třídě funkcí main.

System.out.println("Hello, World!")

System třída se skládá z několika metod a stará se o input,output,error proudy.
Nemůžeme z ní udělat instanci.Voláme statický PrintStream a jeho metodu out do níž posíláme String.

Datové konstrukce

```
    public class Item {
        private int identCislo;
        private String nazev;

        public Item(String nazev,int id){
            this.nazev=nazev;
            this.id=id;
        }
        public int getNazev(){
            return identCislo();
        }
    }
```

Jako u minulého příkladu zde máme třídu Item,která má dva vnitřní datové typy a konstruktor pro vytvoření objektu třídy Item a metodu getNazev.

Příkazy, srovnání s instrukcemi

Příkaz vyjadřuje činnost, jenž má být provedena.Program v imperativním jazyce je obvykle sada podprogramů tvořených posloupností příkazů.
Programovací jazyky obvykle rozlišují:
- výraz
- příkaz
  - jednoduchý příkaz
  - složený příkaz(blok)
- deklaraci

Výraz

```
Jedná se o kombinaci jedné nebo více explicitních literálů, konstant, proměnných, operátorů a funkcí, která programovací jazyk interpretuje.
Například 2+3 je aritmetický výraz, který lze v programu vyhodnotit již při překladu jako číslo 5
Jméno proměnné je výraz, protože označuje hodnotu v paměti, takže y+6 je výraz.
Příkladem relačního výrazu je 4≠4, který se vyhodnotí jako nepravda.
```

Jednoduchý příkaz

```
- přiřazení
- příkaz skoku
- návrat z funkce
- volání funkce
```

Složený příkaz

```
- blok
- cyklus
- podmíněný příkaz
```

Deklarace

```
Deklarace je zápis, který zavádí identifikátor a zpravidla určuje jeho datový dyp a další aspekty pro proměnné, funkce, konstanty.
Překladač je informován o příslušném objektu.
```

Instrukční sada

```
viz. 7. otázka
```

Symbolická instrukce

```
Strojové instrukce si nelze zapamatovat a jsou nahrazené symbolickými instrukcemi.Činnost je angl. zkratka.
```

Způsoby adresování v instrukci

Adresování "přímá data"

```
Je založené na kopírování kostanty do registru.
Konstanta "přímá data" má počet bitů jako registr:
- 8 bitový  => 8b  konstantu
- 16 bitový => 16b konstantu
přesun konstanty             MOV R1,2B Hex          2B hex -> kopíruje ->
<R1>
maskování konstantou         AND R4, 00001100 Bin   <R4> 00001100 Bin ->
<R4>
<R1> obsah v reg. R1        operand,cílový,zdrojový operand
pozn. konstanta jen do místa zdrojového operandu        OZ       Ač
                                                      [[   ] [  konstanta ]]
```

Přímá adresa

```
samostatné proměnné jsou deklarované pseudo příkazem
např. str_A DB 00 hex nebo plocha DW 0000 hex       DB = Definuj Byte
symbolická adresa pro programátora se přeloží       DW = Definuj Word
        str_A   plocha                              pro proměnné 0-255 Dec
na přímé číselné adresy(ve spuštěném programu jsou adresy konst.)0-65500
Dec
      0117 hex  0060 hex        OZ [[  ][ 0060]] AČ     OZ [[  ][0117]] AČ
```

Nepřímé adresování

```
Používá, když např. příkazem cyklus for procházíme polem 1R, ukazujeme si
na hodnoty v poli pomocí (adresového) ukazatele.
Ukazatel je realizovaný registrem v procesoru, který má velikost v bitech
jako je adresa OP pro data nebo adresa části OP zvaná segment.
V ukazateli je uložená počáteční hodnota adresy pole 1R. Tato počáteční
hodnota adresy se při průchodu cyklem mění. Velmi často se operace s členy
pole posunou na sousední adresu = zvětší se hodnota ukazatele o +1.
Registr ukazatele je např.
Samostatný registr SI o velikosti 16b (OP data 64KB)
        (SI... indexový registr)
Samostatný registr BP o velikosti 16b
        (BP... bázový registr)
Spojené registry R7 a R6 o velikosti 2x8b
                 X  a Y  o velikosti 2x8b
```

Nepodmíněný a podmíněný skok

```
Skok je instrukce, která narušuje normální způsob provádění počítačového
programu instrukce po instrukci. Zatímco po provedení jakékoli jiné
instrukce se pokračuje prováděním instrukce následující, po provedení skoku
se pokračuje instrukcí na jiné určené adrese.
```