Programování

Úvod programování

Programování není jen proces psaní zdrojového kódu, jedná se o celý proces složený z několik aktivit (analýza problému, návrh řešení, generování algoritmů a tvorba kód). Doprovázející úkoly s programováním spojené je také testování, ladění a údržba zdrojového kód programu. Účelem programování je najít vhodný sled instrukcí a algoritmů, který povede ke stanovenému výsledku. Pro kódování se využívá programovací jazyk, nebo kombinace více programovacích jazyků, které se volí dle vhodnosti v dané problematice, namísto strojového kódu, který není zcela přívětiví pro programátory vzhledem k jejich složitosti zápisu. Proces programování často vyžaduje znalosti mnoho oblastí IT nebo matematiky.

Programovací paradigmata

Toto téma se věnuje způsobům a stylům přístupu k programování, tedy i postupu při řešení. Různé programovací jazyky tyto paradigmata umožňují/ kombinují , proto máme více druhů programovacích jazyků. Například C# je objektově orientovaný jazyk a je tedy možné využít v případě programování objektových aplikací, zároveň se jedná o imperativní jazyk.

 # Objektové programování

Výkonný kód je v objektovém programování přidružený k datům (metody jsou zapouzdřeny v objektech)

 # Imperativní programování

Popisuje výpočet pomocí posloupnosti příkazů a určuje přesný postup jak daný problém řešit. Vyhodnocuje se na základě podmínek a pomocí příkazů mění svůj stav.

 # Deklarativní programování

Je opakem imperativního programování neřeší se jak se to má udělat, pouze je řečeno co se má udělat, eliminuje se tak možnost vytvořit zbytečné chyby v předem jasně daných příkazech. Příkladem takového jazyka je SQL.

 # Funkcionální programování

Je částečně deklarativní programování, které je chápáno jako matematická funkce, která se postupně krátí a zjednodušuje. Taková jedna funkce, má jen jedno řešení a je ovlivňována pouze

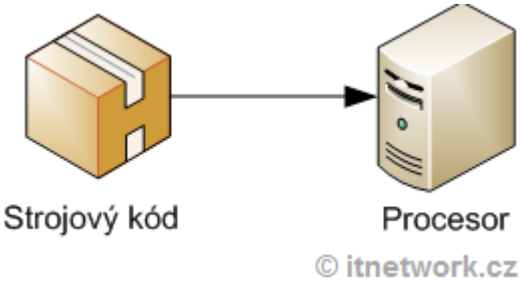
vstupními argumenty. Většina takových programovacích jazyků kombinuje mnoho paradigmat a jsou tedy hybridními.

 # Další:

Aspektově orientované programování, logické programování, generické programování,...

Strojový kód

Každý procesor má omezený počet instrukcí, které může vykonávat, tyto instrukce jsou na nejnižší úrovni, jsou to například instrukce součtu, nebo přepínat v adresách paměti, jsou interpretovány pomocí hexadecimálních čísel (FFFE), ale v nejnižší úrovni se jedná o jedničky a nuly. V konečném případě každý program napsaný v libovolném jazyce, musí být převeden do tohoto kódu a o to se starají různé nástroje. Nástavba nad strojovým kódem je jazyk Assembler, který je totožný se strojovým kódem, ale má slovní označení a tím se stává čitelnější. Kód tohoto jazyka se též překládá do strojového kódu.



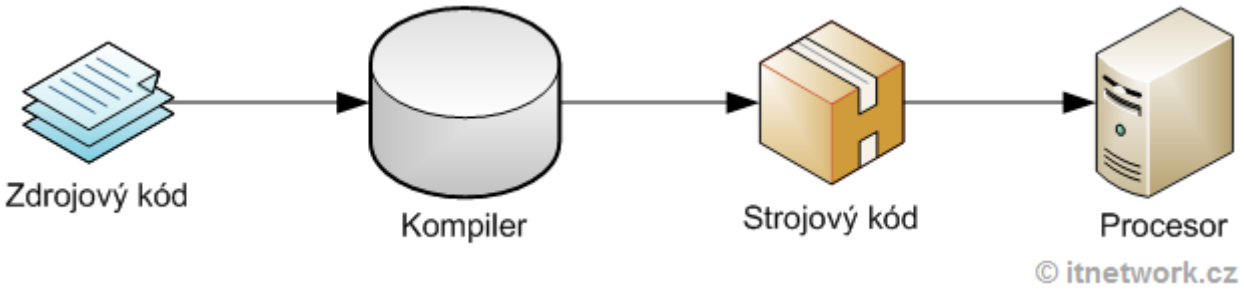
ukázka z programu pro smazání souboru s názvem "soubor.txt"

```
00b5a80 6878 5a61 3176 5f35 645f 6c67 7469 7265
00b5a90 6c61 3632 3532 464d 415a 6179 5f00 3344
00b5aa0 7473 3864 6164 6574 6974 656d 3431 6579
00b5ab0 7261 7349 654c 7061 6559 7261 4e46 4e61
00b5ac0 6962 625a 5f00 3344 7473 3564 7473 6964
00b5ad0 346f 6946 656c 4237 4379 7568 6b6e 6535
00b5ae0 706d 7974 784d 4e46 5a64 0062 445f 6334
00b5af0 726f 3665 6874 6572 6461 3431 6946 6562
00b5b00 4572 6378 7065 6974 6e6f 5f37 435f 616c
00b5b10 7373 005a 445f 7333 6474 6334 6e6f 3176
00b5b20 5f39 545f 7434 7865 5474 7941 5461 5477
00b5b30 7941 5a61 7434 7865 4674 7941 7761 7941
00b5b40 5a61 7941 0061 445f 7333 6474 6438 7461
00b5b50 7465 6d69 3965 6954 656d 664f 6144 3679
00b5b60 5f5f 7463 726f 464d 614e 6969 5a69 3353
00b5b70 7473 3864 6164 6574 6974 656d 5439 6d69
```

```
00b5b80 4f65 4466 7961 5f00 3344 7473 3564 6172
00b5b90 676e 3565 5f33 545f 7235 7465 6f72 4154
```

Kompilované jazyky

Programovací jazyky kompilované jsou jazyky, které jsou psané v jazyce, který je pro programátory snadno čitelný, ne však zcela přívětivý. Je tedy nutné, aby tento zdrojový kód byl přeložený do strojového kódu, k tomu se využívá kompiler, neboli překladač.



Výhody tohoto řešení je rychlost takového programu, protože k překladu dochází jen v jednu chvíli. A tak se stává podobně rychlý jako je strojový kód. Další výhodou je jejich šíření a to především v bezpečnosti, protože zdrojový kód je nepřístupný a není snadné takový kód získat. Další významnou výhodou kompilovaných jazyků je snadné odhalení chyb, které se projeví ve chvíli, kdy dochází ke kompilaci a pokud se v programu chyba nachází kompilace neproběhne.

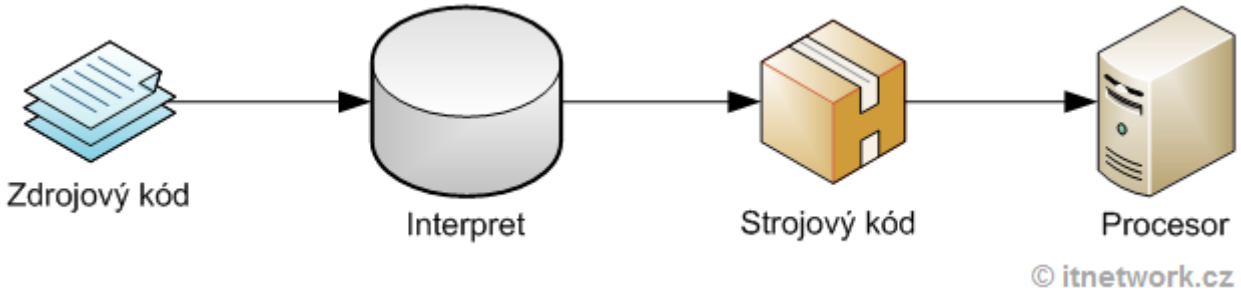
Nevýhodou těchto jazyků je častý problém s nekompatibilitou a závislostí na platformě, kde musí vždy dojít znovu ke kompilaci, například při změně operačního systému. Též není možné zkompilované programy snadno editovat, musí dojít ke znovu kompilaci. Vzhledem k tomu, že při spuštění programu psané v takovém jazyce počítač nerozumí programu a pouze provádí instrukce, může dojít k problémům s pamětí a jejímu přetečení, takovým chybám se však dá zabránit správně napsaným programem. Jelikož se nejedná o chyby v instrukci, ale v systému, kompilátor takovou chybu neodhalí.

"Ukázka výpisu věty "Hello, world!"

```
int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

Interpretované jazyky

Interpretované jazyky místo kompilátoru používají interpreta, který překládá jen části které jsou v danou chvíli potřebné, prochází řádek po řádku a když narazí na nějakou instrukci, zavolá podprogram, který provede to co mu instrukce určuje.



Psaní v takovém jazyce je programátorsky přívětivější, bohužel si za to vybírá svou daň v podobě větší náročnosti na paměť a procesor. Díky tomu, že nejdou do procesoru přímo, fungují na všech procesorech kde pro ně byl jednou zkompilován interpreter. Snadná je také údržba takového programu, protože je možné kód zobrazit a upravit prakticky za běhu.

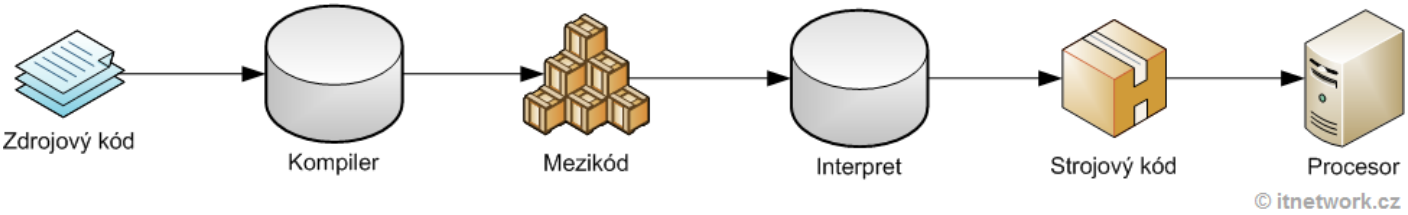
Vedle mnohých výhod mají tyto jazyky také významné nevýhody, jako je obtížné hledání chyb, chybu nikdo neohlásí a tak část takového kódu je "nefunkční", jelikož se tyto programy šíří jako zdrojové kódy, je snadné takový kód krást, případně analýzou zjistit zranitelná místa.

Ukázka výpis věty "Hello, world!"

```
<?php echo "Hello, world!"; ?>
```

Jazyky s virtuálním strojem

Jazyky tohoto typu jsou nejmodernějším a kombinují výhody interpretovaných a kompilovaných jazyků. Princip těchto jazyků je trochu komplikovanější než u předchozích dvou. Zdrojový kód takového programu se nejprve kompiluje do mezikódu u Microsoftu **CIL**, ten je podobný strojovému kódu, mezikód je pak interpretovaný virtuálním strojme, který vytváří strojový kód pro počítač.



Díky chytré kombinaci principům překladu na strojový kód z předchozích dvou typů jazyků, se eliminovali jejich nevýhody, sčítá tak výhody jako jsou, snadné odhalení chyb ve zdrojovém kódu díky kompilaci, interpret zajišťuje stabilitu programu, protože zdrojovému kódu rozumí a zabraňuje provedení nebezpečné instrukce, rychlost takového programu je též na slušné úrovni, program se šíří jako zdrojový kód v CIL je kód hůře čitelný a tím bezpečnější (je možné ho dekompilovat pomocí různých nástrojů), programy jsou také díky CIL snadno přenositelné, protože je lze spustit na všech stanicích, na kterých se nachází virtuální stroj. Je zde také úplně nová výhoda a tím je nezávislost

na programovacím jazyce, některé části programu, mohou být psány například v C# a jiný třeba v C++, protože vše bude přeloženo do CIL.

Ukázka výpisu věty "Hello world!"

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

Backend

Backend je serverová stránka webu. Ukládá a uspořádává data a také zajišťuje, že vše na straně klienta na webu funguje dobře. Je to část webu, se kterou nemůžete vidět a pracovat s ní. Je to část softwaru, která nepřichází do přímého kontaktu s uživatelem. Díly a vlastnosti vyvinuté návrháři backendu jsou nepřímo přístupné uživateli prostřednictvím frontend aplikace. Součástí backendu jsou také činnosti, jako je psaní API, vytváření knihoven a práce se systémovými komponentami.

Frontend

Je část webu, se kterou uživatel přímo komunikuje, se nazývá frontend, nazývá se také „klientská strana“ aplikace. Zahrnuje vše, co uživatelé přímo zažívají: barvy a styly textu, obrázky, grafy a tabulky, tlačítka, barvy a navigační menu. HTML, CSS a Javascript jsou jazyky používané pro vývoj rozhraní frontend. Struktura, design, chování a obsah všeho, co je vidět na obrazovce prohlížeče při otevírání webů, webových aplikací nebo mobilních aplikací, implementují vývojáři frontend. Vývojář musí zajistit, aby web reagoval, tj. aby se zobrazoval správně na zařízeních všech velikostí. Žádná část webu by se neměla chovat neobvykle bez ohledu na velikost obrazovky.

Revision #1
Created 2025-05-21 13:42:26 UTC by Magdalena Dobešová
Updated 2025-05-28 06:20:46 UTC by Magdalena Dobešová