Úvod

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZP3CS – Základy programování 3 (C#)

Doporučená literatura

- **John Sharp, Jon Jagger: Microsoft Visual C# .NET Krok za krokem.**
- Kolektiv autorů **:** C# Programujeme profesionálně.
- Eric Gunnerson : Začínáme programovat v C#.

Požadavky na zápočet

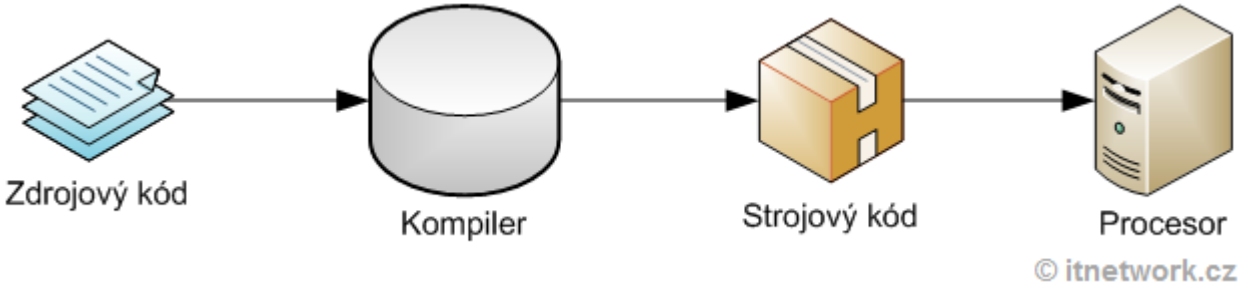
- Zápočet dostane student, který získá alespoň **12 bodů**.
- Body se získávají za:
  - **1 bod** za splnění úkolu na cvičení
  - **0 až 8 bodů** za samostatnou práci
- Všeobecné podmínky pro vypracování samostatných prací:
  - Za práci může student získat **0 až 5 bodů**.
  - Práci musí student odevzdat do začátku zápočtového týdne.
  - Body se strhávají v těchto případech:
    - Řešení je **nekompletní**; byla opomenuta nějaká část, mezní hodnoty vstupů apod.
    - Student odevzdával velmi **neodladěné** řešení.
    - Řešení je sice funkční, ale program vypadá obzvláště **odpudivě.**
  - Svoje řešení student prezentuje v posledních třech cvičeních. Za prezentaci může student získat **0 až 3 body**. Průběh prezentace (10 minut):
    - Předvedení fungujícího programu.
    - Ukázka klíčových částí kódu.

Konzultace

- v pracovně 5.044,
- každá středa (13.00 – 14.00) a pátek (9.45 – 10.45),
- jindy po vzájemné domluvě,
- email: jiri.zacpal@upol.cz,
- skupina v MS Teams **https://1url.cz/2zDKl**

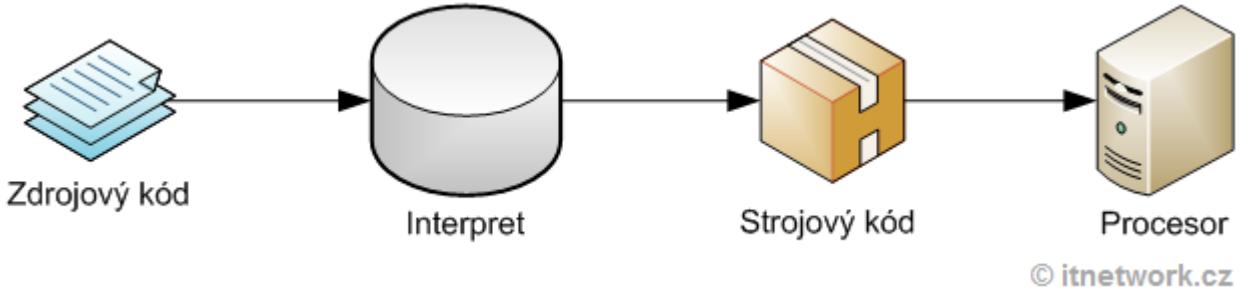
Programovací jazyky

Kompilované jazyky



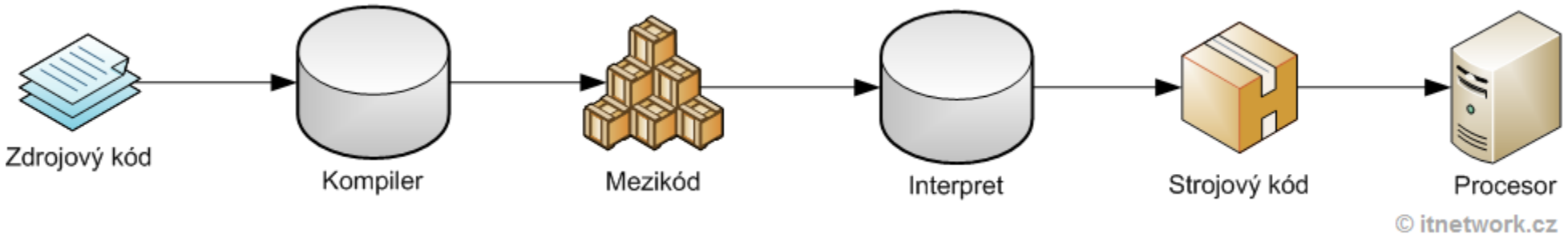
- **Výhody**:
  - **Rychlost**.
  - **Nepřístupnost zdroj. kódu** - Program se šíří již zkompilovaný, není jej možné jednoduše modifikovat pokud zároveň nevlastníte jeho zdroj. kód.
  - **Snadné odhalení chyb ve zdroj. kódu** - Pokud zdrojový kód obsahuje chybu, celý proces kompilace spadne a programátor je s chybou seznámen. To značně zjednodušuje vývoj.
- **Nevýhody:**
  - **Závislost na platformě.**
  - **Nemožnost editace.**
  - **Memory management** - Vzhledem k tomu, že počítač danému programu nerozumí a jen mechanicky vykonává instrukce, můžeme se někdy setkat s velmi nepříjemnými chybami s přetečením paměti. Kompilované jazyky obvykle nemají automatickou správu paměti a jsou to jazyky nižší (s nižším komfortem pro programátora). Běhové chyby způsobené zejména špatnou správou paměti se kompilací neodhalí.
- Příklad: C, C++.

Interpretované jazyky



- **Výhody:**
  - **Přenositelnost**.
  - **Jednodušší vývoj** - Ve vyšších jazycích jsme odstíněni od správy paměti, kterou za nás dělá tzv. garbage collector (řekneme si o něm v seriálu více). Často také nemusíme ani zadávat datové typy a máme k dispozici vysoce komfortní kolekce a další struktury.
  - **Stabilita** - Díky tomu, že interpret kódu rozumí, předejde chybám, které by zkompilovaný program jinak klidně vykonal. Běh interpretovaných programů je tedy určitě bezpečnější, dále umožňuje zajímavou vlastnost, tzv. reflexi, kdy program za běhu zkoumá sám sebe, ale o tom později.
  - **Jednoduchá editace.**
- **Nevýhody**:
  - **Rychlost**.
  - **Často obtížné hledání chyb** - Díky kompilaci za běhu se chyby v kódu objeví až v tu chvíli, kdy je kód spuštěn. To může být někdy velmi nepříjemné.
  - **Zranitelnost** - Protože se program šíří v podobě zdrojového kódu, každý do něj může zasahovat nebo krást jeho části.
- Příklad: PHP.

Jazyky s virtuálním strojem

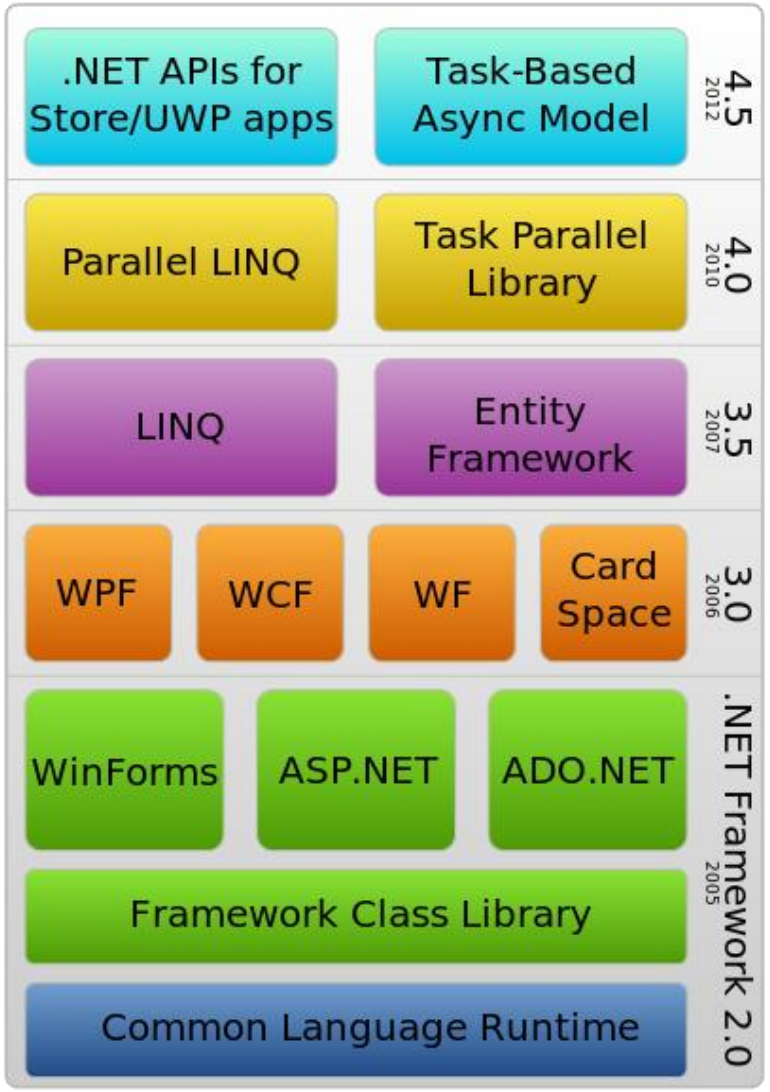


- **Výhody:**
  - **Odhalení chyb ve zdrojovém kódu** - Díky kompilaci do CIL (Common Intermediate Language) jednoduše odhalíme chyby ve zdrojovém kódu.
  - **Stabilita** - Díky tomu, že interpret kódu rozumí, zastaví nás před vykonáním nebezpečné operace a na chybu upozorní. Můžeme také provádět reflexi (i když pro CIL, ale od toho jsme většinou odstíněni).
  - **Jednoduchý vývoj** - Máme k dispozici hitech datové struktury a knihovny, správu paměti za nás provádí garbage collector.
  - **Slušná rychlost** - Rychlost se u virtuálního stroje pohybuje mezi interpretem a kompilerem. Virtuální stroj již výsledky své práce po použití nezahazuje, ale dokáže je cachovat, sám se tedy optimalizuje při četnějších výpočtech a může dosahovat až rychlosti kompileru (Just In time Compilator). Start programu bývá pomalejší, protože stroj překládá společně využívané knihovny.
  - **Málo zranitelný kód** - Aplikace se šíří jako zdrojový kód v CIL, není tedy úplně jednoduše lidsky čitelná.
  - **Přenositelnost** - Asi je jasné, že hotový program poběží na každém železe, na kterém se nachází virtuální stroj. To ale není vše, my jsme dokonce nezávislí i na samotném jazyce. Na jednom projektu může dělat více lidí, jeden v C#, druhý ve Visual Basic a třetí v C++. Zdrojové kódy se poté vždy přeloží do CILu.
- Příklad: Java, C#.

Jazyk C#

.NET framework

- Je tvořen:
  - Jazyk – několik jazyků, jedním z nich je C#.
  - Visual Studio - IDE (Integrated Development Environment), prostředí, ve kterém píšeme zdrojový kód a které nám také pomáhá s vývojem.
  - Virtuální stroj - CLR (Common Language Runtime) je virtuální stroj, který interpretuje CIL do instrukcí fyzického procesoru.
  - Knihovny - Microsoft nám v podstatě dodává kompletní sadu knihoven, ve které máme předpřipravenou řadu struktur a komponent, např. pro práci s konzolí, databázemi, formulářovými prvky a podobně.



Charakteristiky jazyka C#

- vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework,
- později schválený standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270),
- Microsoft založil C# na jazycích C++ a Java (a je tedy nepřímým potomkem jazyka C, ze kterého čerpá syntaxi),
- C# lze využít k tvorbě
  - databázových programů,
  - webových aplikací a stránek, webových služeb,
  - formulářových aplikací ve Windows,
  - softwaru pro mobilní zařízení (PDA a mobilní telefony) atd.,
- komponentně orientovaný.

Vlastnosti jazyka C#

- V C# neexistuje vícenásobná dědičnost – to znamená, že každá třída může být potomkem pouze jedné třídy.
- Neexistují žádné globální proměnné a metody, všechny musí být deklarovány uvnitř tříd. Náhradou za globální proměnné a metody jsou statické metody a proměnné veřejných tříd.
- V objektově orientovaném programování se z důvodu dodržení principu zapouzdření často používá vzor, kdy k datovým atributům třídy lze zvenčí přistupovat pouze nepřímo, a to pomocí dvou metod: metody *get* (accessor) a metody *set* (mutator).
- C# je typově bezpečnější než C++. Jediné předdefinované implicitní konverze jsou takové, které jsou považovány za bezpečné. Příkladem budiž rozšiřování celočíselných typů (např. z 32bitového na 64bitový) nebo konverze z odvozeného typu na typ rodičovský. Neexistuje však implicitní konverze z celočíselných typů na boolean ani implicitní konverze mezi výčtovými a celočíselnými typy.
- C# nepotřebuje a ani neobsahuje dopřednou deklaraci – pořadí deklarace metod není důležité.
- Jazyk C# je case sensitive – rozlišuje mezi velkými a malými písmeny.

MS Visual Studio 2017

MS Visual Studio 2017

- zdarma pro studenty
- vytvoření projektu
  - popis souborů, které tvoří projekt
  - soubory s příponou cs
- vkládání existujících souborů do projektů
- kompilace, spuštění, debugging (F5xCtrl F5)
- „odvšivení“
  - spouštění krok po kroku (F10, F11)
  - breakpoint
  - sledování stavu proměnných

Konzolová aplikace

Příklad 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace zp3cs_uvod_konzole
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Ahoj světe!");
        }
    }
}
```

Obory názvů

- zjednodušuje použití identifikátorů
- používá kontejnery pro identifikátory
- identifikátory musí být unikátní v rámci jednoho kontejneru
- definice:

```
namespace jmeno_oboru_nazvu
{
...
}
```

- použití:

```
jmeno_oboru_nazvu.identifikator
```

- zjednodušení syntaxe pomocí příkazu `using`

```
using meno_oboru_nazvu;
```

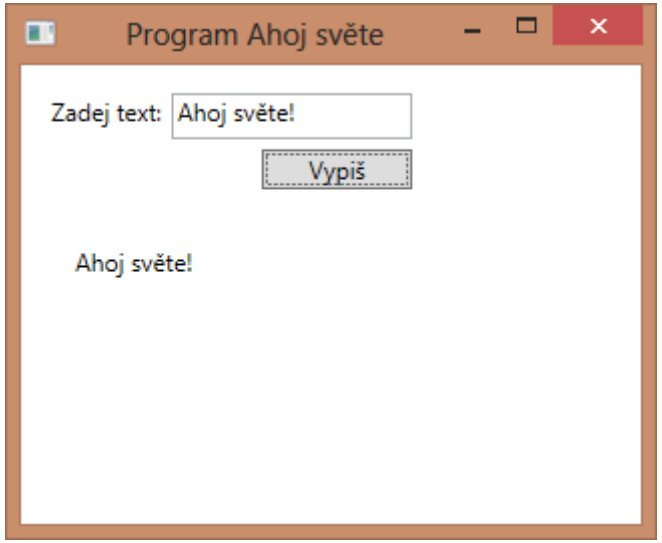
- místo `jmeno_oboru_nazvu.identifikator` stačí použít pouze `identifikator`

Grafická aplikace

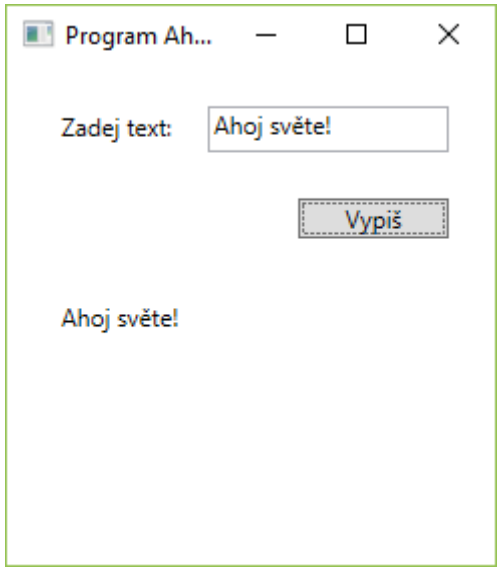
Grafická aplikace

- vytvoření projektu
- Windows Forms
  - klasická tvorba formulářů
- Formulář WPF
  - popis v jazyce XAML
- vytvoření uživatelského rozhraní
- psaní kódu

Příklad 2



Příklad 3



Úkol

- Vytvořte konzolovou aplikaci, která vypíše vaše jméno a adresu.
- Totéž vytvořte v grafické aplikaci.