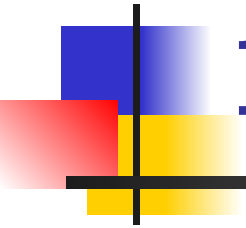


Úvod do programování
1. hodina

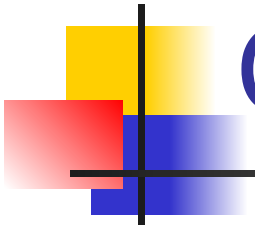
doc. RNDr. Jan Lánský, Ph.D.
Katedra informatiky a matematiky
Fakulta ekonomických studií
Vysoká škola finanční a správní
2015

Cíle předmětu

Programování je o zautomatizování základních postupů. Nutno prakticky trénovat. Ideálně hodinu každý den.

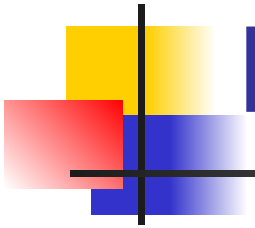
- Na předložený problém najít algoritmus
  - jednoznačný postup jeho řešení
- Tento algoritmus zapsat pomocí programu
  - Přednášky budou v C#
  - Lze pracovat s jiným jazykem
- Určit časovou a prostorovou složitost použitého algoritmu.
- První ze čtyř semestrů programování

C# je prostředek, ne cíl



Cíle hodiny

- Obecné informace o předmětu
- Ukázka: postup řešení problému v přirozeném jazyce a jeho převod do C#
- Instalace C#, vytvoření projektu, spuštění programu
- Syntax C#
  - Proměnné
  - Přiřazení, aritmetické operátory (zbytek po celočíselném dělení), výpis na obrazovku
  - Porovnací operátory, podmíněný příkaz (se zápornou větví, vnořený)



Literatura - C#

- Předmět učím v C# druhý rok, takže jsou k dispozici zatím jen slajdy
- Lze použít literaturu pro Pascal v kombinaci s libovolnou knihou pro C#
- Učebnice C#
  - Miroslav Virius: C# pro zelenáče, Neocortex, Praha, 2002

Literatura - Pascal

- P. Töpfer: Úvod do programování
  - IS ☐ studijní materiály ☐ UPg.pdf
  - Rozsahem odpovídá probíranému učivu
- P.Töpfer: Algoritmy a programovací techniky, Prometheus Praha 1995, 2. vydání 2007
- P.Töpfer: Základy programování v úlohách, Scientia Praha 1997

Příklady není nutné stihnout na hodině, postačí je vypracovat doma a na hodině prezentovat

V případě zájmu lze řešit příklady i na jiných mých cvičeních: B_PJC , B_Prg, B_OOP, B_AS – až do naplnění kapacity místnosti

Zápočet

- Splnění povinných příkladů z alespoň 7 z 12 cvičení (z toho splněna cvičení 3, 4, 5, 6 a 8)
  - IS ☐ studijní materiály ☐ cviceni.doc
  - Vyřešeným příkladům je nutno rozumět, na požádání je mírně modifikovat
  - Vyřešené příklady je nutno prezentovat osobně (**NE emailem**)
  - Doporučený termín odevzdání je do konce semestru, ale možné je i kdykoliv později (když mě zastihnete a budu mít čas)
  - **Prezentování cizích zdrojových kódů jako vlastních ☐ Disciplinární řízení ☐ Vyloučení ze studia**
  - **Source code plagiarism may lead to exclusion from the University**
  - **Плагиат исходного кода может привести к исключению из университета**

Vyplatí se studovat poctivě a průběžně

Šance na úspěch při studiu

obtížnost	1. semestr	2. semestr	3. semestr	4. semestr
Poctivec	střední	těžká	velmi těžká	střední
Podvodník 1	chycen: †			
1 %	1%	nightmare	nightmare	těžká
Podvodník 2	1 %	chycen: †		
0,01 %	1 %	1 %	nightmare+	nightmare
Podvodník 3	1 %	1 %	chycen: †	
0,0001 %	1 %	1 %	1 %	nightmare+
Podvodník 4	1 %	1 %	1 %	chycen: †
0,000001 %	1 %	1 %	1 %	1 %

Šance na výhru ve sportce 0.00001 % cca 10 x vyšší něž Podvodník 4

Zkouška

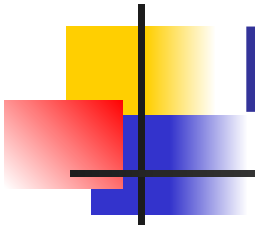
Zkoušku lze skládat kdykoliv během mojí výuky v semestru (v ZS i LS), není třeba se ohlašovat dopředu, stačí přijít.

- Libovolný procedurální jazyk
- Řešení příkladu na papír
  - Není důležitá přesná syntax, ale správné použití řídících konstrukcí
  - Povoleny jen psací potřeby
- [Nedoporučeno] Zkoušku lze skládat i na PC
  - zadání je obtížnější
  - Povolena jen vestavěná nápověda vývojového prostředí

Zkouška

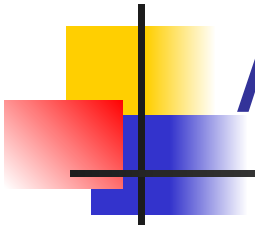
Na známku A je třeba znát i učivo potřebné na známky C a E.

- Známka E (dobře)
  - Podmínky, cykly, funkce, pole, třídění, řetězce, soubory
- Známka C (velmi dobře)
  - Dvojrozměrná pole, dlouhá aritmetika, rekurze, zásobník, fronta
- Známka A (výborně)
  - Backtracking, vlna, ořezávání, heuristiky



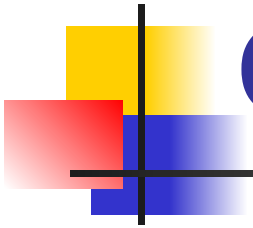
Problém a instance

- Problém - Obecná úloha
- Instance problému – konkrétní data
- Problém: Sčítání tří jednociferných čísel
  - Instance: 2+5+1, 1+7+9, 3+3+6, …
- Problém: Najít dopravní spojení mezi dvěma městy
  - Instance: Praha – Brno, Cheb – Ostrava, …



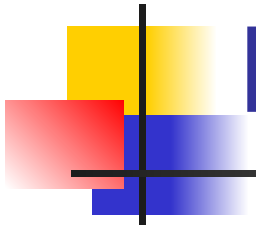
Algoritmus

- [Elementárnost] Konečná posloupnost elementárních operací. Každá elementární operace je jednoznačně a přesně definována.
- Algoritmus pro každou instanci problému [Obecnost] převede vstupní data na požadovaná výstupní data [Příčinnost] za konečný počet kroků [Konečnost].
- [Mechaničnost] Algoritmus v každém kroku jednoznačně určuje další postup. Algoritmus nevyžaduje vlastní iniciativu vykonavatele.
- [Opakovatelnost] Na stejných datech dosáhneme vždy stejného výsledku (výjimka použití náhodných čísel)



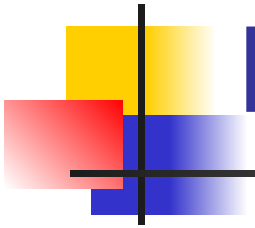
Co není algoritmus

- [Není elementárnost] Postavíme raketu, doletíme s na Mars a postavíme tam město.
- [Není obecnost] Postup řešící sečtení čísel 3 a 5
- [Není příčinnost] Vypsání textu bez reakce na vstupy
- [Není konečnost] Projdeme všechna celá čísla a …
- [Není mechaničnost] Přizpůsobíme jízdu stavu vozovky …
- [Není opakovatelnost] Podívám se na barvu a popíši ji slovy: "světle hnědá", "kaštanová"



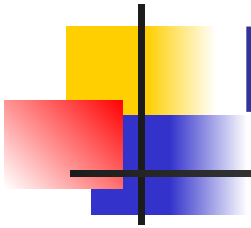
Program

- Zápis algoritmu v programovacím jazyce
- Jeden algoritmus může být zapsán více programy
  - Různé programovací jazyky
  - I v rámci jednoho programovacího jazyku.
- Množiny elementárních operací můžou být různé
  - Př.: pro výpočet mocniny je důležité, zda máme k dispozici sčítání, násobení, mocnění jako elementární operace



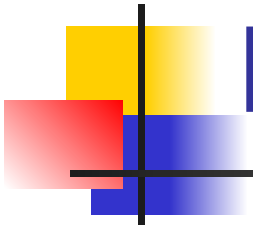
Problém: Součet tří čísel

- Na začátku nám budou nadiktována rychle za sebou tři celá čísla. Našim úkolem je tato čísla **sečíst**.
- Operace **zápis**: Na libovolné místo papíru si můžu zapsat libovolnou hodnotu a toto místo označit libovolným názvem. Po zápisu na papír zapomeneme vše co máme v hlavě.
- Operace **načtení**: Z papíru můžeme přečíst libovolnou hodnotu. V hlavě jsme schopni si najednou pamatovat pouze dvě hodnoty.
- Operace **sečtení**: Umíme atomicky **sečíst dvě hodnoty**, které máme v hlavě. Ale výpočet trvá dlouho a vyčerpává nás. Proto výslednou hodnotu musíme okamžitě zapsat na papír a odpočinout si.



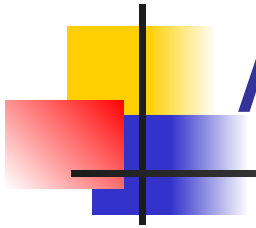
Problém: Součet tří čísel

- Neumíme
  - Vstupní čísla jsou nám diktována tak rychle, že je stihneme jen zapsat na papír, nedokážeme je sčítat průběžně (jsme pomalí)
  - Nedokážeme si v hlavě pamatovat výslednou hodnotu po sečtení dvou čísel (jsme příliš unaveni).
  - Neumíme si v hlavě pamatovat všechna tři čísla. (nemáme dobrou paměť)
  - Neumíme zapisovat na papír částečné výpočty.



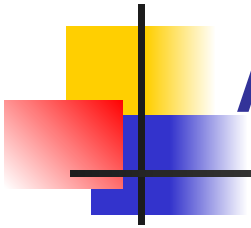
Motivace: Složitější problém

- Neumíte si omezení svých schopností (paměť, rychlost výpočtu) představit ?
- Složitější problému
  - Čísel místo 3 je 1000
  - Čísla jsou 5-ciferná (10-ciferná, …)
  - Místo sčítání je můžeme násobit, odečítat, dělit, …



Algoritmus: Součet tří čísel

A. Zapsání tří čísel **x**, **y** a **z** na papír
B. Sečtení **x** a **y** a zapsání součtu do **vysledek**
C. Sečtení **vysledek** a **z** a zapsání součtu do **vysledek**
D. Vystup algoritmu je v místě papíru označeném **vysledek**. Můžeme ho přečíst a nahlas říci tazateli.



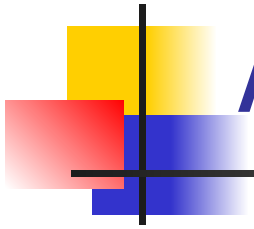
Algoritmus: Součet tří čísel

A. Zapsání tří čísel na papír
   1) Zapíšeme hodnotu prvního čísla na papír, místo kde je hodnota zapsána označíme **x**
   2) Zapíšeme hodnotu druhého čísla na papír, místo kde je hodnota zapsána označíme **y**
   3) Zapíšeme hodnotu třetího čísla na papír, místo kde je hodnota zapsána označíme **z**

Algoritmus: Součet tří čísel

B. Sečtení **x** a **y**

1) Z papíru přečteme hodnotu z místa označeného jako **x**.
2) Z papíru přečteme hodnotu z místa označeného jako **y**.
3) Sečteme dvě hodnoty čísel, která máme v hlavě, získanou hodnotu zapíšeme na papír do místa, které označíme jako **vysledek**



Algoritmus: Součet tří čísel

C. Sečtení **vysledek** a **z**

1) Z papíru přečteme hodnotu z místa označeného jako **vysledek**.
2) Z papíru přečteme hodnotu z místa označeného jako **z**.
3) Sečteme dvě hodnoty čísel, která máme v hlavě, získanou hodnotu zapíšeme na papír do místa označeného jako **vysledek**

Papír: Součet tří čísel

- **x** 5
- **y** 3
- **z** 9
- **vysledek** ~~8~~ 17

Microsoft Visual Studio

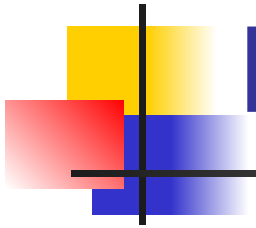
- https://www.visualstudio.com
- Downloads ☐ All Downloads ☐ Visual Studio 2015 ☐
- Community / Express for Desktop
  - Zdarma pro studenty, nutno se registrovat
  - Plně postačuje pro naše účely (i Express)
- Professional / Enterprise
  - Zdarma na vyzkoušení
  - Professional (45 USD/ měsíc)
  - Enterprise (250 USD / měsíc)
- Instalace návod
  - http://www.sallyx.org/sally/c/visual-studio-2013.php

Doporučený jazyk rozhraní AJ
Lépe se googlí pomoc

Visual Studio na VŠFS

- Virtuální server
- Start ☐ Spustit ☐ mstsc
  - Server: programator
  - Login: učo
  - heslo: jiné než pro přístup do IS, přidělí osobně na IT oddělení
- Doporučení: Pracovat na vlastním notebooku

Konzolová aplikace



- Microsoft Visual Studio 2013
- File ☐ New Project ☐ Visual C# ☐ ConsoleAplication
  - Name: pojmenujeme aplikaci
  - Location: kde bude aplikace uložená v PC
  - SolutionName: Např.: Hodina 01
- Editujeme soubor program.cs

Nová konzolová aplikace

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // sem vložíme zdrojový kód
        }
    }
}
```

Program: Součet tří čísel

```
static void Main(string[] args)
{
    int x = 5;   // A.1
    int y = 3;   // A.2
    int z = 9;   // A.3

    int soucet = x + y;   // B
    soucet = soucet + z; // C

    Console.WriteLine("{0} + {1} + {2} = {3}", x, y, z, soucet);
    Console.ReadLine();   // D
}
```

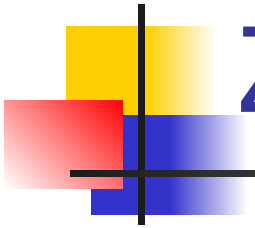
Nalevo od rovná se = zápis na papír

Napravo od rovná se = operace v hlavě

Výpis na obrazovku
5+3+9 = 17

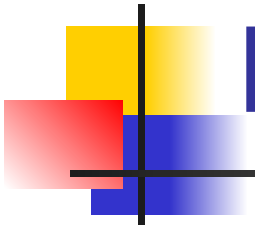
Standardní vstup (= klávesnice)
standardní výstup (= obrazovka)

Počkáme na zmáčknutí klávesy Enter



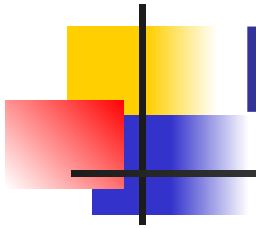
Zdrojové kódy

- Nepoužívat diakritiku (á, ž, …)
- Důležitá velikost písmen (case sensitive)
  - Dvě různé proměnné: vysledek vs. Vysledek
- Pozor na překlepy
- Automatické odsazování, respektovat
- Komentáře // moje poznamka



Práce s MVS

- Editujeme pouze obsah funkce Main
  - Obsah složených závorek
  - Zbytku programu si nevšímáme
- Klávesa F5 spuštění programu
- Chyby podtrženy vlnovkou
- Okno Output
  - Seznam chybových hlášení.
  - Po kliknutí na chybu nás nastaví do zdrojového kódu



Proměnná

Hodnotu proměnné mohu použít teprve poté, co jsem jí hodnotu přiřadil (inicializace)

- Proměnná reprezentuje přidělený kus paměti procesu. Paměť mohu číst a zapisovat do ní.
- Název – identifikátor programovacího jazyka umožňující přístup k paměti, kde je uložena hodnota proměnné.
  - Př. x, y, z, vysledek, hodnota1, p5d663rWQ
  - Obdoba označení čísla na papíře (viz. náš příklad)
- Datový typ – udává velikost paměti přidělené proměnné a její interpretaci:
  - Př. celé číslo, reálné číslo, řetězec, …
- Hodnota – interpretace obsahu paměti dle datového typu proměnné. Př. 17, "Ahoj", 5.569

Deklarace proměnné

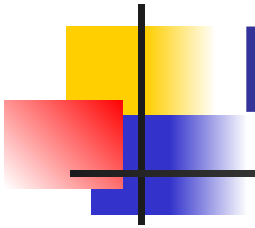
- syntax: název datového typu *mezera* název proměnné *středník*
- Celé číslo - int
- int x;
- int z = 5;
- int tmp1, tmp2;

Deklarace s inicializací

Více proměnných stejného datového typu

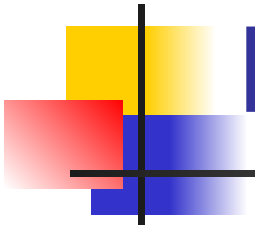
Binární operátor

- Syntax: levý operand *operátor* pravý operand
- aritmetické operátory: + - * / %
  - Celočíselné dělení /
  - Zbytek po celočíselném dělení %
  - 7/2 □ 3
  - 7%2 □ 7 – (7/2)*2 □ 7-3*2 = 1



Přiřazení

- Infixní binární operátor
- Syntax: levý operand = pravý operand
- Levý operand musí být proměnná
- Levý operand bude mít hodnotu pravého operandu.
- Odpovídá zapsání hodnoty na papír (viz. náš příklad)



Příkaz

- Výraz ukončený středníkem
  - Aritmetický výraz 2*(5-2+7/4)
- Přiřazovací příkaz
  - **vysledek** = **x** + 5;
  - **y** = **z**;
  - **M5x** = 156;
- Volání funkce (následující slajd)
- Podmínky, cykly, složený, … (později)

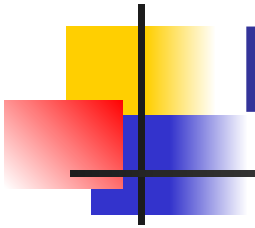
Kulaté závorky
Priorita operátorů

Výpis na obrazovku

- Console.Write(výraz);
  - Vypíše na obrazovku *výraz*
- Console.WriteLine(výraz);
  - Vypíše *výraz* a odřádkuje

```
int x = 5;
Console.Write(x);
Console.Write(112);
```

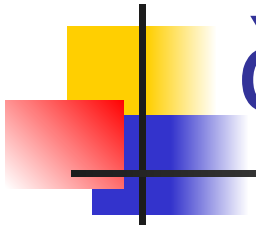
- Výraz = číslo, text, formátovací řetězec
- Console.Write("{2} +{0} = {1}", x, y, z);
  - Znaky uvedené v uvozovkách jsou brány jako text
  - Výjimka: {n} udává číslo parametru, který chceme vypsat. Číslujeme od nuly.



Problém: Nejmenší ze tří čísel

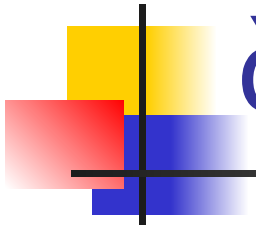
- Na začátku nám budou nadiktována rychle za sebou tři celá čísla. Našim úkolem je zjistit **nejmenší z těchto čísel**.
- Operace **zápis** a **načtení** (viz sečtení tří čísel)
- Operace **porovnání**: umíme atomicky určit, zda hodnoty **dvou čísel**, která máme v hlavě, **jsou nebo nejsou v požadovaném vzájemném vztahu** (<, >, = …). Výpočet trvá dlouho a vyčerpává nás. Výsledek výpočtu je ANO/NE.
- **Podmíněná operace**: Pokud porovnání skončilo odpovědí ANO, vykonáme podmíněné operace. Pokud porovnání skončilo odpovědi NE, podmíněné operace nevykonáme.

Algoritmus: Nejmenší ze tří čísel



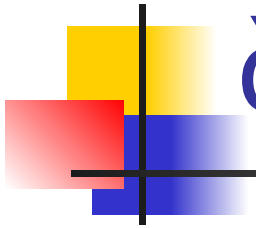
A. Zapsání tří čísel **x**, **y** a **z** na papír, zapsání **x** do **nejmensi**.
B. Porovnání **y** a **nejmensi**. Je-li **y** menší, zapsání **y** do **nejmensi**.
C. Porovnání **z** a **nejmensi**. Je-li **z** menší, zapsání **z** do **nejmensi**.
D. Vystup algoritmu je v místě papíru označeném **nejmensi**. Můžeme ho přečíst a nahlas říci tazateli.

Algoritmus: Nejmenší ze tří čísel



A. Zapsání **x**, **y**, **z** a **nejmensi** na papír
   1) Zapíšeme hodnotu prvního čísla na papír, místo kde je hodnota zapsána označíme **x**
   2) Zapíšeme hodnotu druhého čísla na papír, místo kde je hodnota zapsána označíme **y**
   3) Zapíšeme hodnotu třetího čísla na papír, místo kde je hodnota zapsána označíme **z**
   4) Z papíru přečteme hodnotu z místa označeného jako **x**.
   5) Zapíšeme hodnotu čísla v hlavě na papír, místo kde je hodnota zapsána označíme **nejmenší**

Algoritmus: Nejmenší ze tří čísel



B. Vyhodnocení **y** jako kandidáta na nejmenší číslo

1) Z papíru přečteme hodnotu z místa označeného jako **y**.
2) Z papíru přečteme hodnotu z místa označeného jako **nejmensi**.
3) Porovnáme hodnoty čísel, která máme v hlavě. Pokud je hodnota prvního číslo **y** menší než hodnota druhého čísla **nejmensi** provedeme operace B.4 a B.5.
4) [platí-li B.3] Z papíru přečteme hodnotu z místa označeného jako **y**.
5) [platí-li B.3] Zapíšeme hodnotu čísla v hlavě na papír do místa označeného jako **nejmenší**

Algoritmus: Nejmenší ze tří čísel

C. Vyhodnocení **z** jako kandidáta na nejmenší číslo

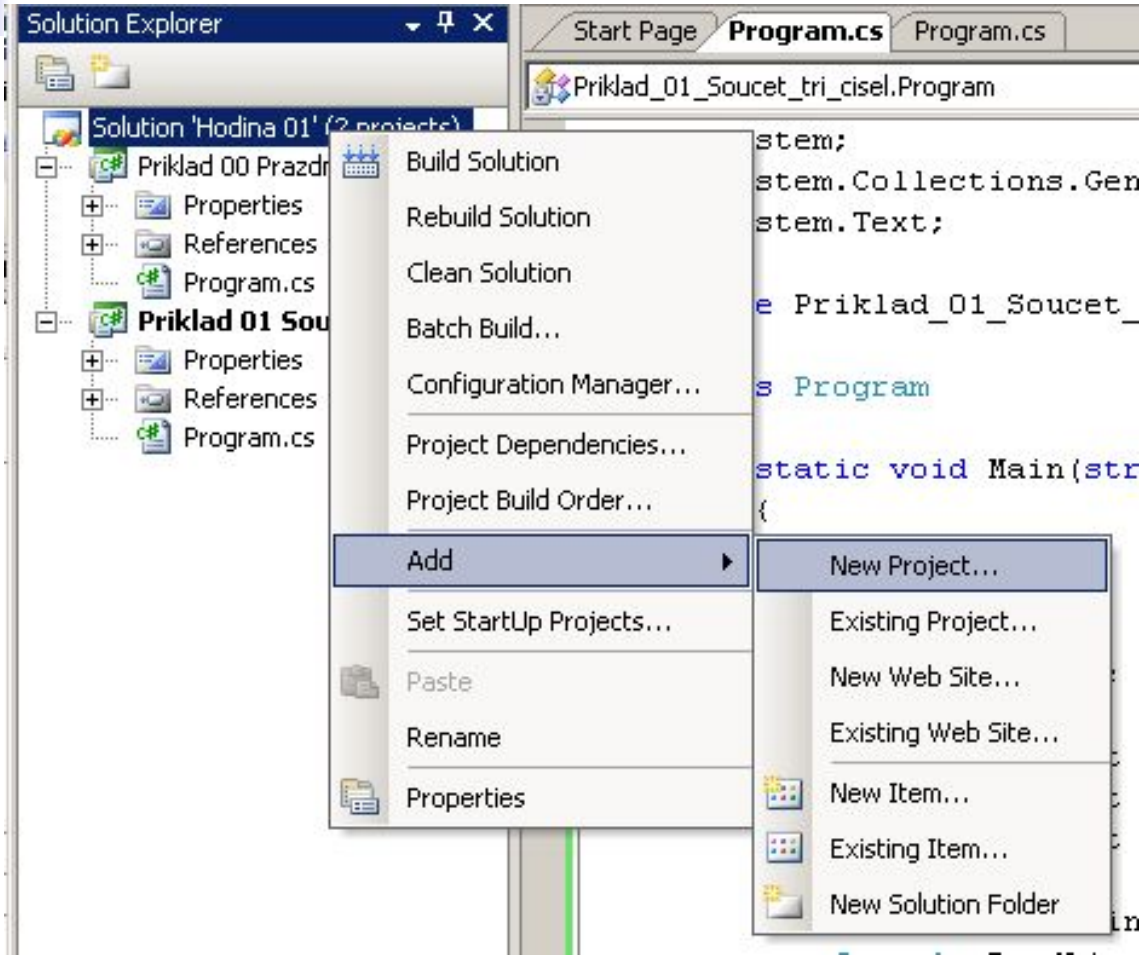
1) Z papíru přečteme hodnotu z místa označeného jako **z.**
2) Z papíru přečteme hodnotu z místa označeného jako **nejmensi**.
3) Porovnáme hodnoty čísel, která máme v hlavě. Pokud je hodnota prvního číslo **z** menší než hodnota druhého čísla **nejmensi** provedeme operace C.4 a C.5.
4) [platí-li C.3] Z papíru přečteme hodnotu z místa označeného jako **z.**
5) [platí-li C.3] Zapíšeme hodnotu čísla v hlavě na papír do místa označeného jako **nejmenší**

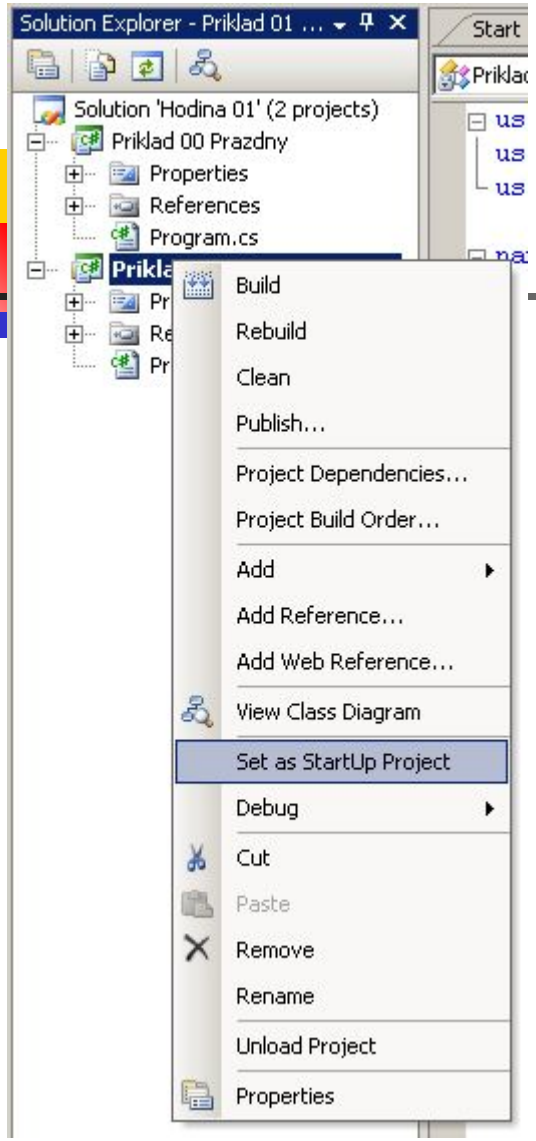
Stejné jako B, ale místo y je z

Papír: Nejmenší ze tří čísel

- **x** 8
- **y** 4
- **z** 6
- **nejmensi** ~~8~~ 4

Přidání nového projektu





Aktivní projekt

- Solution obsahuje více projektů
- Aktivní vždy jen 1 projekt
- Častá chyba: spouštíme předchozí projekt místo aktuálního

Program: Nejmenší ze tří čísel

```
static void Main(string[] args)
{
    int x = 8;  // A.1
    int y = 4;  // A.2
    int z = 6;  // A.3
    int nejmensi = x; // A.4 a A.5

    if (y < nejmensi)
        nejmensi = y; // B
    if (z < nejmensi)
        nejmensi = z; // C

    Console.WriteLine("Nejmensi z cisel {0}, {1} a {2} je {3}", x, y, z, nejmensi);
    Console.ReadLine();  // D
}
```

Podmíněný příkaz **if**: Pokud je splněna podmínka v závorce, vykoná se příkaz

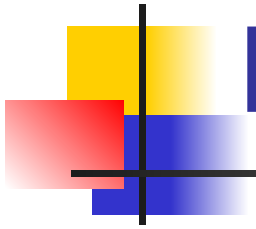
z < min { x,y } ☐ z je nejmenší

Porovnávací operátory

- Infixní binární operátor
  - Syntax: levý operand [porovnání] pravý operand
- < menší, <= menší nebo rovno
- > větší, >= větší nebo rovno
  - < (alt + 60), > (alt + 62)
- == rovno, != nerovno
- Výsledek typu bool
  - hodnoty true / false

Přiřazení =
Porovnání ==

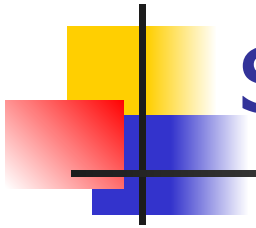
Podmíněný příkaz



if klíčové slovo
zdrojový kód modře

- if (výraz) příkaz;
- Výraz reprezentující podmínku
  - Musí být v závorce
  - Výraz typu bool
    - Hodnota výrazu TRUE ☐ podmínka splněna,
    - Hodnota výrazu FALSE ☐ podmínka nesplněna
- Příkaz
  - Pokud **je splněna** podmínka, **vykoná** se příkaz
  - Pokud **není splněná** podmínka, příkaz se **nevykoná**

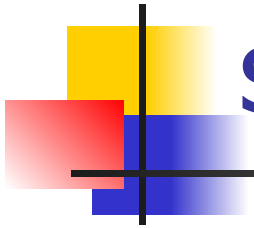
Problém: Počet lichých a sudých čísel



- Na začátku nám budou nadiktována rychle za sebou tři celá čísla. Našim úkolem je zjistit **kolik z těchto čísel je lichých a kolik sudých**.
- Operace **zápis**, **načtení**, **aritmetické operace**, **porovnávací operace**
- **Podmíněná operace se zápornou větví**: Pokud porovnání skončilo odpovědí ANO, vykonáme podmíněné operace. Pokud porovnání skončilo odpovědi NE, vykonáme jiné podmíněné operace.

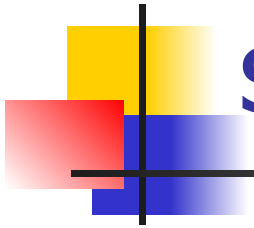
Nepoužijeme, že sudých = 3 - lichých

Algoritmus: Počet lichých a sudých čísel



A. Zapsání tří čísel **x**, **y** a **z** na papír, zapsání **0** do **lichych** a **0** do **sudych**.
B. Zjištění zda x je liché nebo sudé, navýšení příslušného čítače (lichych nebo sudych) o jedna.
C. Zjištění zda y je liché nebo sudé, navýšení příslušného čítače (lichych nebo sudych) o jedna.
D. Zjištění zda z je liché nebo sudé, navýšení příslušného čítače (lichych nebo sudych) o jedna.
E. Vystup algoritmu je v místech papíru označených **lichych** a **sudych**. Můžeme ho přečíst a nahlas říci tazateli.

Algoritmus: Počet lichých a sudých čísel



A. Zapsání **x**, **y**, **z**, **lichych**, **sudych** na papír
   1) Zapíšeme hodnotu prvního čísla na papír, místo kde je hodnota zapsána označíme **x**
   2) Zapíšeme hodnotu druhého čísla na papír, místo kde je hodnota zapsána označíme **y**
   3) Zapíšeme hodnotu třetího čísla na papír, místo kde je hodnota zapsána označíme **z**
   4) Zapíšeme hodnotu 0 na papír, místo kde je hodnota zapsána označíme **lichych**
   5) Zapíšeme hodnotu 0 na papír, místo kde je hodnota zapsána označíme **sudych**

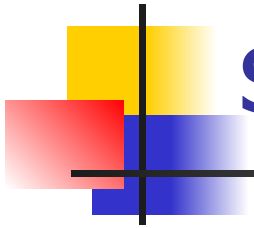
Algoritmus: Počet lichých a sudých čísel

B. Vyhodnocení zda **x** je liché nebo sudé

1) Z papíru přečteme hodnotu z místa označeného jako **x.**
2) Vypočteme zbytek po celočíselném dělení hodnoty čísla v hlavě hodnotou **2.** Výslednou hodnotu zapíšeme na papír na místo, které označíme **tmp**
3) Z papíru přečteme hodnotu z místa označeného jako **tmp**.
4) Porovnáme hodnotu čísla, které máme v hlavě s hodnotou **1**. Pokud se hodnoty čísel rovnají, provedeme operace B.5 a B.6. Pokud se hodnoty čísel nerovnají, provedeme operace B.7 a B.8

tmp pomocná proměnná, v C# ji nevidíme

Algoritmus: Počet lichých a sudých čísel



B. Vyhodnocení zda **x** je liché nebo sudé

5) [platí-li B.4] Z papíru přečteme hodnotu z místa označeného jako **lichych**.

6) [platí-li B.4] Vypočteme součet hodnoty čísla v hlavě a hodnoty 1. Výslednou hodnotu zapíšeme na papír do místa označeného jako **lichych**

7) [neplatí-li B.4] Z papíru přečteme hodnotu z místa označeného jako **sudých**.

8) [neplatí-li B.4] Vypočteme součet hodnoty čísla v hlavě a hodnoty 1. Výslednou hodnotu zapíšeme na papír do místa označeného jako **sudých**

Papír: Počet lichých a sudých čísel

- **x** 5
- **y** 2
- **z** 9
- **lichych** 0 1 2
- **sudych** 0 1
- **tmp** 1 0 1

Červená – krok B
Modrá – krok C
Zelená krok D

Program: Počet lichých a sudých čísel

```
static void Main(string[] args)
{
    int x = 5;  // A.1
    int y = 2;  // A.2
    int z = 9;  // A.3
    int lichych = 0; // A.4
    int sudych = 0; // A.5

    if (x % 2 == 1) lichych = lichych + 1; // B
    else sudych = sudych + 1;
    if (y % 2 != 0) lichych++; // C
    else sudych++;
    if (z % 2 == 0) sudych++; // D
    else lichych++;

    Console.Write("Z cisel {0}, {1} a {2} ", x, y, z);
    Console.WriteLine("je lichych {0} a sudych {1}.", lichych, sudych);
    Console.ReadLine();  // E
}
```

První zápis funguje jen pro kladná čísla

Třikrát jiný zápis podmínky - stejná funkčnost pro x, y, z

Pozor na prioritu
Aritmetická operace a porovnání v jednom kroku
(V algoritmu musely být dva kroky)

x = x + 1
Odpovídá x++

Podmíněný příkaz se zápornou větví

- if (výraz) příkaz1;
- else příkaz2;
- Výraz reprezentující podmínku
  - v závorce, typ bool
- Příkaz
  - Pokud **je splněna** podmínka daná výrazem, **vykoná** se příkaz1 (a příkaz2 se nevykoná)
  - Pokud **není splněná** podmínka daná výrazem, **vykoná** se příkaz2 (a příkaz1 se nevykoná)

If, else klíčová slova zdrojový kód modře else je nepovinné

Vnořený podmíněný příkaz

- if (výraz1) příkaz1;
- else if (výraz2) příkaz2;
- else příkaz3;
  - Pokud **je splněna** podmínka daná výrazem1, **vykoná** se příkaz1 (a zpracování podmínky končí)
  - Pokud **není splněná** podmínka daná výrazem1, testuje se podmínka daná výrazem2
    - Pokud **je splněna** podmínka daná výrazem2, **vykoná** se příkaz2
    - Pokud **není splněná** podmínka daná výrazem2 **vykoná** se příkaz3

Postupně zanořený else nepovinný

Vnořený podmíněný příkaz

- if (výraz1)
  - if (výraz2) příkaz1;
  - else příkaz2;
- else příkaz3
  - Pokud **je splněna** podmínka daná výrazem1, otestuje se podmínka daná výrazem2
    - Pokud **je splněna** podmínka daná výrazem2, **vykoná** se příkaz1
    - Pokud **není splněná** podmínka daná výrazem2 **vykoná** se příkaz2
  - Pokud **není splněná** podmínka daná výrazem1, provede se příkaz3

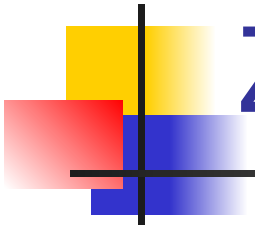
Dvojitě vnořený else patří k nejvíce zanořenému cyklu

Vnořený podmíněný příkaz

Dva různé zápisy. Postupné zanoření (1. zápis) vhodnější než dvojité zanoření (2. zápis)

```
if (x == 0) Console.WriteLine("Nula");
else if (x > 0) Console.WriteLine("Kladne");
else Console.WriteLine("Zaporne");

if (x != 0)
    if (x > 0) Console.WriteLine("Kladne");
    else Console.WriteLine("Zaporne");
else Console.WriteLine("Nula");
```



Zpětná vazba

- Objevili jste ve slajdech chyby?
  - Včetně pravopisných
- Nechápete nějaký slajd?
  - Je příliš obtížný, nesrozumitelný?
- Máte nějaký nápad na vylepšení?
- Anonymní formulář
  - Odeslání za pár vteřin
- http://goo.gl/forms/WxkZqBsZLs