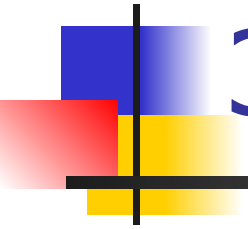
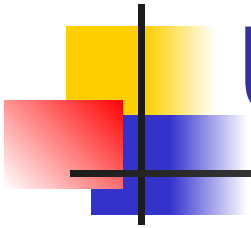


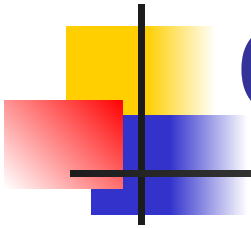
Úvod do programování
3. hodina

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



Umíme z minulé hodiny

- Syntax
  - Složený příkaz
  - Logické operátory
  - Čtení vstupu z klávesnice
  - Cykly while a for (continue, break), vnořené cykly
- Algoritmy
  - Prohození hodnot dvou proměnných
  - Ciferný součet, Euklidův algoritmus
  - Prvočíselný rozklad, test prvočíselnosti
- Základy časové složitosti algoritmů



Cíle hodiny

- Syntax
  - Funkce
  - Datové typy (celočíselné, reálné, bool, …)
  - Priorita a asociativita operátorů
  - Náhodná čísla Random
  - Výčtové typy enum
  - Vícenásobně větvení switch
  - Struktura struct - příklad datum
- Matematické funkce Math
- Testování správnosti programu

Funkce - motivace

Parametry funkce: občas se používá označení argumenty

- Funkce je ohraničený souvislý kus zdrojového kódu, který vykonává logicky vymezenou část programu.
  - Vstupy = parametry funkce (klidně i 0)
    - Případné načítání vstupů z klávesnice bude v Main
  - Výstup = Návratová hodnota (nemusí být)
- Jednou vytvoříme, můžeme (opakovaně) volat z různých částí zdrojového kódu
  - Případné změny a opravy chyb děláme na jednom místě, místo několika míst
- Přehlednější zdrojový kód

Funkce - implementace

Funkce je jednoznačně určena svým názvem, počtem a datovými typy parametrů

- Hlavička funkce
  - Klíčové slovo **static** následované typem návratové hodnoty (např. **int**) následované jménem funkce (např. **Abs**).
  - V kulatých závorkách **( )** je uzavřen seznam dvojic datových typů a názvů parametrů funkce (formální parametry), jednotlivé dvojice jsou odděleny čárkou.
- Tělo funkce
  - Ve složených závorkách **{ }** jsou uzavřeny příkazy
  - Funkce končí (a vrací hodnotu) použitím klíčového slova **return**

Funkce - volání

Předávání parametrů hodnotou: Při volání funkce se skutečný parametr zkopíruje. V těle volané funkce lze hodnoty parametru modifikovat, neprojeví se ve volající funkci

- Funkce je volána svým jménem následovaným v kulatých závorkách uzavřeným seznamem parametrů funkce (skutečné parametry).
  - Seznam může být prázdný
  - Počet a datové typy skutečných a formálních parametrů musí být shodné
- Návratová hodnota funkce je výraz, který lze dále použít
  - Lze ji také ignorovat

Funkce absolutní hodnota

Musí být static, nezkoumejme proč

Typ návratové hodnoty

Formální parametr hodnota

Vrácení návratové hodnoty a ukončení funkce

```
static int Abs(int hodnota)
{
    if (hodnota < 0) return -hodnota;
    else return hodnota;
}
```

Main: volající funkce, Abs: volaná funkce

Volání funkce Abs
Skutečný parametr -5

```
static void Main(string[] args)
{
    int x = -5, y = 4;
    int xAbs = Abs(x);
    Console.WriteLine("Absolutni hodnota {0} je {1}", x, xAbs);
    Console.WriteLine("Absolutni hodnota {0} je {1}", y, Abs(y));
    Console.ReadLine();
}
```

Volání funkce

Funkce načtení a výpis čísla

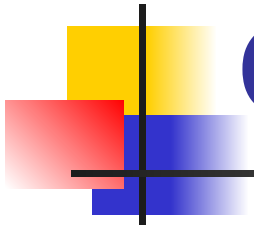
```
static int NactiCislo()
{
    int x;
    Console.Write("Zadejte Cislo: ");
    x = Convert.ToInt32(Console.ReadLine());
    return x;
}

static void VypisCislo(int x)
{
    Console.Write("Hodnota cisla je {0}", x);
    Console.ReadLine();
}

static void Main(string[] args)
{
    int a;
    a = NactiCislo();
    VypisCislo(a);
}
```

Bez parametrů

Bez návratové hodnoty

Pro přehlednost: Implementujte funkce v pořadí jejich použití

Mezi názvy formálních a skutečných parametrů není žádný vztah. Mohou se jmenovat stejně, různě



Celočíselné datové typy

- **long** - velikost 64 bitů (-9e18, +9e18)
  - Bezznaménkový **ulong** (0, 1e19)
- **int** – velikost 32 bitů, (-2mld, +2mld)
  - Bezznaménkový **uint** (0, 4 mld)
- **short** – velikost 16 bitů (-32768, 32767)
  - Bezznaménkový **ushort** (0, 65 535)
- **byte** – velikost 8 bitů (0 až 256)
  - Znaménkový **sbyte** (-128, 127)
- Naše defaultní volba int, menší typy jen pro pole

Reálné datové typy

- **double** – 64 bit – preferovaný typ
  - Přesnost 15-16 míst
  - ± 5e-324 až ± 1e308
- **float** – 32 bit
  - Přesnost 7 míst
  - -3e38 až +3e38
- **decimal** – 128 bit – finance
  - Přesnost 28-29 míst
  - -7e28 až 7e28
- Lze použít +, -, *, /
  - Nelze použít %, dělení je reálné, ne celočíselné

Zápis 5e-324 znamená 5*10^-324

Další datové typy

- Prázdný typ **void**
- Logický typ **bool** - hodnoty true, false
- **char** - velikost 16 bitů – UTF16 znak
  - Uzavřen v apostrofech
  - Př.: char x = 'a';
- **string** - proměnlivá délka, složen z charů
  - Uzavřen v uvozovkách
  - Př.: string a = "ahoj";

bool již známe

Práce s řetězci bude později

Přetečení datového typu

Překročení maximálního rozsahu datového typu

```
int i, souc = 1;
for (i = 0; i < 20; i++)
{
    souc = souc * 1000;
    Console.WriteLine(souc);
}
Console.ReadLine();
```

Řešení: Větší datový typ, třeba reálný

```
1000
1000000
1000000000
-727379968
-1530494976
-1486618624
-559939584
-1593835520
-402653184
1073741824
0
0
0
0
0
0
0
0
0
0
```

Přestože přetečení je deterministické, hodnoty získané přetečením bychom neměli jakkoliv využívat

Reálná čísla - Exponent

- **e** nebo **E** zkracuje zápisy čísel, které by obsahovaly mnoho nul
- Širší funkčnost než v matematice (základ nemusí být z intervalu [0,10)
- x**e**y nebo x**E**y je zkratka za zápis x * 10 na y
- Příklady:
  - -1256e3 = -1256000
  - 1756e-7 = 0.0001756
  - 25698.26e-1 = 2569.826

Zbytečné, ale možné použití

Nepřesnost reálné aritmetiky

```
int i,opak = 300;
double jedna = 1;
for (i = 0; i < opak; i++)
    jedna = jedna / 7;
for (i = 0; i < opak; i++)
    jedna = jedna * 7;

Console.WriteLine(jedna);
Console.ReadLine();
```

V cyklu se nasčítá zaokrouhlovací chyba

Máme omezenou přesnost, v ní neprovedeme přesně

0,999999999999998

Řešení: netestovat přesnou shodu, ale testovat zda rozdíl hodnot v absolutní hodnotě je velmi malý

Konverze mezi celými a reálnými datovými typy

```
int i, x = 5, y = 7;
double d, a = 3.6;

d = x / y;
d = (double)x / y;
d = 5 / 7.0;
i = a + x;
```

Pravá strana se počítá celočíselně až výsledek se převede na double

Pravá strana se počítá reálně. Přetypování (double)

Pravá strana se počítá reálně, protože 7.0 je reálné

Syntax chyba: a je reálné, nelze automaticky převést na celé

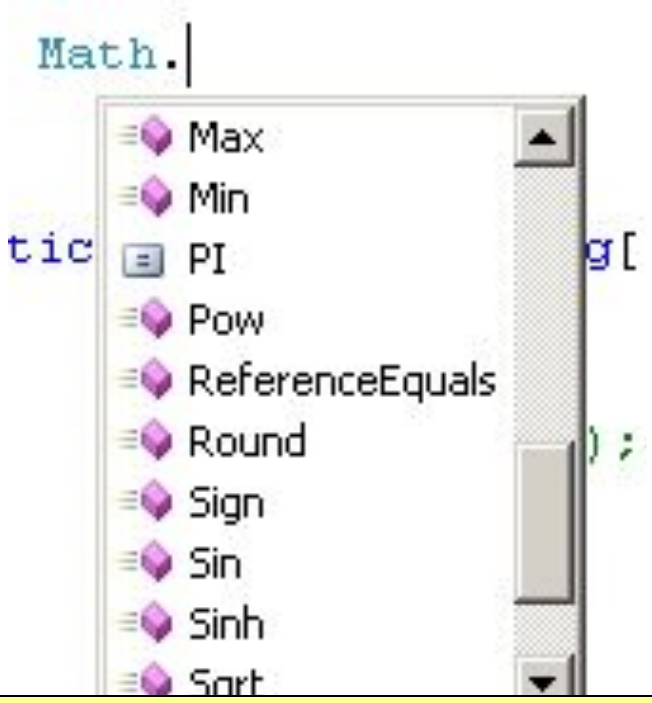
Menší číselný datový typ se automaticky konvertuje na větší, je-li nutné

Reálná čísla - funkce

```
double d = 5 + 2 / 7.0;
double x, y, z, o, p;

x = Math.Truncate(d);
y = Math.Ceiling(d);
z = Math.Round(d);

o = Math.Pow(5, 7);
p = Math.Log(o, 5);

Console.WriteLine("{0:NO}", o);
Console.WriteLine("{0:N3}", d);
```

Dolní celá část
Horní celá část
zaokrouhlení

Mocnina
Logaritmus



{0:N} číslo na dvě desetinná místa, mezery po 3 cifrách
{0:Nx} x desetinných míst

Napíšeme Math tečka a vybereme si:
PI, E, Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Sign, Min, Max, Abs, Sqrt, ...

Priorita a asociativita operátorů

Datový typ výrazu

Tečka – přístup k položce

1) . ++ -- typeof
2) ! (typ)
3) * / %
4) + -
5) < <= > >= == !=
6) &&
7) ||
8) =

Přetypování

a++ je a=a+1
a-- je a=a-1

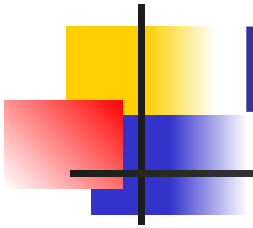
- Asociativita
  - Operátory se stejnou prioritou se vyhodnocují směrem zleva doprava
    - Př. x = 3 * 5 / 2
  - Výjimku tvoří =, to se vyhodnocuje zprava do leva
    - Př. x = y = 5;

7 vs 6

Závorky

```
int a = 2, b = 3, c = 4;
bool x;

x = a + b * (c + a) < b / a && a * c > b + a;
x = ((a + b * (c + a)) < (b / a)) && ((a * c) > (b + a));
```

- Závorky umožňují změnit pořadí operací
- Lze je použít i pro zvýšení přehlednosti když chceme původní pořadí zachovat
  - Nebo když nevíme, jaké pořadí je



Náhodná čísla

- Důležitý pomocník při testování programů
- Hodnotu náhodného čísla nelze předpovědět
- V PC používáme pseudonáhodná čísla
  - Hazardní hry se snaží o co největší náhodnost
- Random r = new Random();
  - Oznámení, že budeme pracovat s náhodnými čísly
- r.Next(min,max)
  - Vrátí náhodné celé číslo z intervalu [min, max)
- r.NextDouble
  - Vrátí náhodné reálné číslo z intervalu (0,1)

Hádání čísla

Kvůli první iteraci cyklu

Hrozba nekonečného cyklu. Jak program upravit?

```
int x = -1;
Random r = new Random();
int kostka = r.Next(1,7);
int pokusu = 0;

Console.WriteLine("Uhodnete cislo, ktere padlo na kostce");
while (x != kostka)
{
    Console.Write("Zadejte hledane cislo (1 az 6): ");
    x = Convert.ToInt32(Console.ReadLine());
    pokusu++;
}

Console.WriteLine("Cislo {0} uhodnuto na {1} pokusu", x, pokusu);
Console.ReadLine();
```

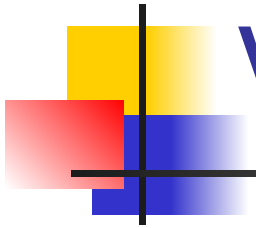
Náhodné číslo (1 až 6)

Zatím jsme neuhodli

Náš tip

Počet pokusů

Výčtové typy - motivace



- Malá pevně daná množina hodnot, které je rozumné vyjádřit textově.
- Textovému vyjádření hodnoty odpovídá i číselné vyjádření
  - S textovým vyjádřením se lépe pracuje, zvyšuje přehlednost zdrojového kódu
- Př.: Dny v týdnu, měsíce, barvy, karty, šachové figurky, (bool)

Reálně je bool slabší, nelze a++

Někdy je lepší měsíc číselný

Výčtové typy - definice

```
enum Den
{
      Pondeli = 1,
      Utery,
      Streda,
      Ctvrtek,
      Patek,
      Sobota,
      Nedele
}
```

- Definujeme vně funkce
- Hodnoty výčtového typu jsou identifikátory oddělené čárkou
- Identifikátorům můžeme přiřadit číselnou hodnotu
  - Pokud to neuděláme, přiřadí se automaticky. První hodnota = 0, další hodnota = hodnota (předchozí) +1

Výčtový typ - načtení

- Převod textového zápisu hodnoty z klávesnice na identifikátor výčtového typu

```
static Den NactiDen()
{
    Den d = 0;
    Console.Write("Zadejte den v tydnu: ");
    d = (Den) Enum.Parse(typeof(Den), Console.ReadLine(), true);
    return d;

}
```

Ignorování velikosti písmen

Nutný operátor typeof pro zjištění výčtového typu

Přetypovat

Vrátí číselnou hodnotu identifikátoru výčtového typu (1. parametr). Jemuž odpovídá řetězec (2. parametr)

Výčtový typ - ukázka

```
Den d;

for (d = Den.Pondeli; d <= Den.Nedele; d++)
   Console.Write("{0} ", d);
Console.WriteLine();

d = NactiDen();
Console.WriteLine("Den {0} je {1}. dnem v tydnu", d, (int) d);
```

Výčtový typ může být řídící proměnnou cyklu

Naše funkce na načtení hodnoty z klávesnice

Pokud chci použít číselnou hodnotu výčtového typu, musím ho přetypovat

Vícenásobně větvení switch

- Syntax switch (podmínka)
  - { case (hodnota1): příkazy1; break; …;
  - case hodnota1: příkazy1; break;
  - case hodnota2: příkazy2; break;
  - …
  - default: příkazy; }
- Pro různé hodnoty jednoho výrazu (typicky proměnné) provedeme různé příkazy
  - Hodnoty jsou uvozeny návěštím case
- Ekvivalent postupně zanořených podmíněných příkazů
  - Ty se vyhodnocují pomaleji než switch, ale pro nás není tento rozdíl zásadní

Ukončení příkazů pro dané návěští break

Příklad - switch

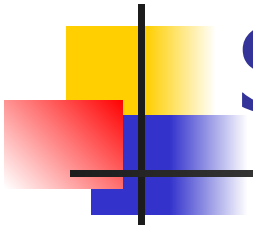
```
switch (d)
{
    case Den.Sobota:
    case Den.Nedele:
        Console.WriteLine("Volno");
        break;
    case Den.Pondeli:
        Console.WriteLine("Začátek pracovního týdne");
        break;
    case Den.Patek:
        Console.WriteLine("Konec pracovního týdne");
        break;
    default:
        Console.WriteLine("Prostředek pracovního týdne");
        break;
}
```

Obvykle int, ale může být i enum,
V C# i string (v C, C++ ne)

Pro Sobotu a Neděli se provedou stejné příkazy

Nezapomínat break

Nepovinně: ostatní nevyjmenované hodnoty



Struktury - motivace

- Struktura je složený datový typ.
  - Obsahuje položky: položka je dvojice identifikátor a datový typ
  - Datové typy položek mohou být navzájem různé, mohou to být i struktury
  - Ke struktuře lze přistupovat jako k celku (parametr funkce, návratová hodnota) nebo k jednotlivým položkám (přístup k hodnotě položky)
  - Položky struktury by měly spolu mít nějaký logický vztah
- Př.: Osoba, datum, komplexní číslo

Struktury syntax

- struct [Název] {
- public [datový typ1] [identifikátor1];
- ...
- public [datový typN] [identifikátorN]; }

Musí být public, nezkoumejme proč

Po klíčovém slovu struct následuje název struktury a ve složených závorkách uzavřený seznam položek. Jednotlivé položky jsou odděleny středníkem. Položka obsahuje klíčové slovo public, datový typ a svůj název.

Struktura datum

```
struct Datum
{
    public int den;
    public int mesic;
    public int rok;
}
```

- V C# existuje datový typ DateTime
- Měsíc by mohl být výčtový typ
- V C# bývá zvykem položky struktur začínat velkými písmeny
  - My budeme malými (neobjektový přístup)

Struktura datum - použití

```
static void Main(string[] args)
{
    Datum dat, nyni;
    int dni;

    dat.den = 31;
    dat.mesic = 10;
    dat.rok = 2008;

    nyni.rok = DateTime.Now.Year;
    nyni.mesic = DateTime.Now.Month;
    nyni.den = DateTime.Now.Day;

    dni = DatumNaCislo(nyni) - DatumNaCislo(dat);

    Console.Write("Cim je z hlediska informatiky vyznamne datum ");
    Console.WriteLine("{0}. {1}. {2} ?", dat.den, dat.mesic, dat.rok);
    Console.WriteLine("Od tohoto data uplynulo {0} dni.", dni);
    Console.ReadLine();
}
```

Proměnné typu Datum: dat a nyni

Přístup k položce struktury: tečka

DateTime.Now Aktuální datum a čas

Datum na číslo – častý postup

Zná někdo odpověď ?

Přestupný rok

```
static bool PrestupnyRok(int rok)
{
    if (rok % 4 != 0) return false;
    if (rok % 100 != 0) return true;
    if (rok % 400 != 0) return false;
    return true;
}
```

Nejsou nutné else, protože je tu return

- Při přestupném roce se přidává den 29.2.
- Pokud je rok dělitelný 4 je přestupný
  - Výjimka, je-li dělitelný 100 není přestupný
    - Výjimka z výjimky, je-li dělitelný 400 je přestupný (problém Y2K)

Převod data na číslo I. část

Počítáme počet dní, které uplynuly od bazického data (vybral jsme 1.1.1900) do zadaného data

```
static int DatumNaCislo(Datum d)
{
    int i, vys = 0;
    vys = vys + d.den;

    for (i = 1900; i < d.rok; i++)
    {
        vys = vys + 365;
        if (PrestupnyRok(i)) vys++;
    }
```

Počet dní uplynulých z posledního měsíce

Procházíme celé roky a za každý započítáme počet dní které měl

Převod data na číslo II. část

```
for (i = 1; i < d.mesic; i++)
{
    switch (i)
    {
        case 4:
        case 6:
        case 9:
        case 11:
            vys = vys + 30;
            break;
        case 2:
            vys = vys + 28;
            if (PrestupnyRok(d.rok)) vys++;
            break;
        default:
            vys = vys + 31;
            break;
    }
}
return vys;
```

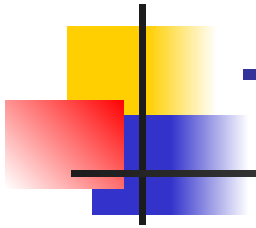
Procházíme celé měsíce (posledního rok není celý) a za každý započítáme kolik měl dní

Měsíce co mají 30 dní jsou jen 4, rozumnější pro ně case

Únor je specialita

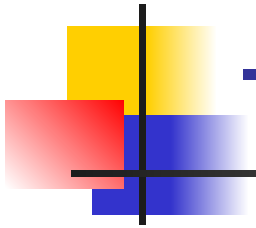
Měsíců co mají 31 dní jsou je 7, rozumnější pro ně default

Testování správnosti programu - teoreticky



- Důkaz správnosti algoritmu + důkaz, že program implementuje algoritmus
  - Nad schopnosti průměrného studenta
- Vyzkoušením všech kombinací vstupních hodnot a kontrola získaných výsledků
  - Často bývá kombinací nekonečno
- V praxi: kritické systémy (letectví, jaderná elektrárna)

Testování správnosti programu - prakticky



Námi vybraný vstup nebývá ošklivý

- Program jde zkompilovat
- Program po spuštění nespadne
- Program pro zadaný vstup nevrací naprostý nesmysl
- Otestujeme mnoho náhodných vstupů, ideálně pomocí Random
- Otestujeme mezní hodnoty (kraje intervalů vstupních hodnot
- Kontrola správnosti výstupů proti externímu programu, který řeší stejný problém jako my

Není zaručena správnost programu, snažíme se jen odhalit co nejvíce chyb

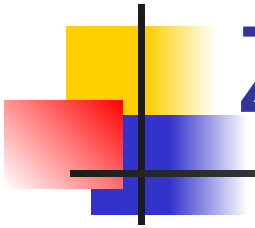
Čínský test prvočíselnosti

- N je prvočíslo, pokud dělí číslo $2^N - 2$
- Př. 7 dělí $2^7 - 2 = 126 = 7 * 18$
- Př. 12 nedělí $2^{12} - 2 = 4094$ 341,1666666
- Algoritmus selže pro číslo 341
  - $341 = 31 * 11$ a 341 dělí $2^{341} - 2$
  - Praktická ukázka, že vyzkoušení mnoha vstupních hodnot nezaručuje správnost programu.

Kde kontrolovat výsledky

- Google [název programu] + online
- Matematický online software
  - http://www.wolframalpha.com/
- Databáze posloupností celých čísel
  - https://oeis.org/
- Databáze textů
  - https://www.gutenberg.org/
- Datum a čas
  - http://www.timeanddate.fasterreader.eu/pages/cs/date-after-days-calc-cs.html

Vyhledávání dle části posloupnosti



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