Základní operátory

- + (binární) sčítání,
- - odečítání, podobně * a /
- = přiřazení,
- == porovnání na rovnost, !=, >, <, >=, <= (nerovnosti),
- logické && (and), || (or), ! (not), & (and), | (or), ^ (xor),
- první dvě vyhodnocují líně (je-li výsledek jasný, přestanou), 4. a 5. vyhodnocují úplně (takto v C#, v C to bylo trochu jinak).
- unární ++, -- (prefixové a postfixové, tedy a++ vs ++a).
- Pozor na priority!
- Řešení jako v Pascalu, tedy závorkovat.
- Pozor na porovnání a přiřazení!
- Přiřazovací příkazy: +=, -=, *=, /=, &= apod.

Komentáře, bloky

- Jednořádkové `// do konce řádku`
- Víceřádkové `/* komentář hodně dlouhý dokud ho neukončíme explicitně */`
- Složené závorky označují blok (jako v Pascalu `begin` a `end`), tedy { `blok; příkazů;`}

Základní řídicí struktury

- `if (podm) příkaz;`
- `if (podm) příkaz; else příkaz;`
- `if (podm) { blok } [else {blok}]`
- `while(podm) příkaz_nebo_blok`
- `for(init;podm;inkr) tělo`
- *init je inicializační kód, podm je podmínka, inkr je inkrementační výraz, tedy příklad:*
- `for(a=1;a<10;a++)b+=a;`
- *Co je tohle?* `if(a=5) a_je_pet();`
- `do příkazy(); while(podm);`
- *Poznámka:* `a=b=c=1;`

Konstrukce `switch`

ekvivalent `case ... of`

```
switch(i)
{       case 1: i_je_jedna(); break;
        case 2: case 3: i_je_2_nebo_3(); break;
        default: vsechno_je_jinak(); break;
}
```

Break ukončuje jednotlivé větve, v C se vykonával kód, dokud se nenarazí na `break`, C# toto neumožňuje (jednu větev je třeba ukončit před začátkem druhé).

Definice funkcí

```
typ jmeno(parametry)
{ telo }
```

Chceme-li vrátit návratovou hodnotu, použijeme klíčové slovo `return` a za ně umístíme vracený výraz: `return 0;`

Definice funkcí

podrobnosti

- procedura je funkce, která vrací `void`,
- `void` `vracíme` pomocí `return;`,
- kulaté závorky jsou operátor zavolání (nebo operátor definice funkce), nelze je tudíž oproti Pascalu vynechat,
- Parametry jsou implicitně předávány hodnotou. Jazyk C předání referencí neměl (místo toho byly pointery).
- C# umí předávat parametry též referencí a výsledkem:
- `void divna(ref int a, out int vysledek);`
- proměnná `a` je předána referencí, proměnná `vysledek` výsledkem, což je téměř totéž (tedy je řešena též odkazem, ale hodnota proměnné předávané výsledkem není funkci vůbec předána).

Objektové programování I

- Objekty, třídy, metody a atributy.
- Svět sestává z objektů, většina věcí je objekt.
- Objekt student se na přednášce vyskytuje několikrát,
- jednotlivci přísluší do třídy student.
- Jsou vybaveni atributy a metodami (aby mohli studovat).

Objektové programování II

- Objekty jsou instance třídy.
- Statické metody (resp. atributy) příslušejí třídě (ne jednotlivým objektům).
- Modifikátory `public`, `private`, `protected` řídí přístup k jednotlivým položkám (může každý / může jen metoda dané třídy / smí současná třída a potomek).
- Třídu (v C++, Javě, C# a dalších) vytváříme pomocí klíčového slova `class`.

Rituál programování v C#

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{    class Program
     {    static void Main(strings[] args)
          {    Console.WriteLine("Sem se bude psát
ten program!");
          }
     }
}
```

Operátor tečky má podobný význam jako v Pascalu.

Eukleidův algoritmus

```
static void Main(string[] args)
{    Console.WriteLine("Dvě čísla sem:");
     int a=int.Parse(Console.ReadLine());
     int b=int.Parse(Console.ReadLine());
     while(a!=b)
          if(a>b) a-=b;
          else b-=a;
     Console.Write("NSD je: ");
     Console.WriteLine(a);
}
```

Objekty

v jazyku C# a jejich využití

- Svět stále sestává z objektů,
- objekty konkrétní VS abstraktní,
- ponožka VS obraz (na slidu).
- Objekty patří do jednotlivých tříd, tedy
- uvedená jména objektů jsou názvy tříd.

Objekty, atributy, metody

- Objekty jsou nástupci struktur (recordů),
- struktury měly proměnné (prvky), u objektů tomu říkáme atributy,
- objekty mají i funkce zvané metody.
- V programu vytváříme různé objekty, nejen Program.