# 5. Strukturované programování

## Cíl

* + Snaha větší srozumitelnosti, vyšší kvality a kratší doby programování programu

## Algoritmus

* + Jeden vstup a jeden výstup

## Složení na nejnižší úrovni programu

* + 3. struktury, které se skládají z programovacích konstrukcí

### Sekvence

* + Posloupnost příkazů

### Výběr

* + Podmínky – IF, else
  + Výběr dle hodnoty – switch, case

### Opakování

* + Cykly
  + S podmínkou za tělem cyklu – repeat until
  + S podmínkou před tělem cyklu – while
  + S parametrem

## Pojmy strukturovaného programování – vše bráno jako C#

### Komentáře

* + Jednořádkové – // text
  + Víceřádkové – /\* text \*/

### Klíčová slova

* + Rezervované Identifikátory – slova rezervovaná jazykem (double, null, while)

### Identifikátory

* + Slova, která zastupují jméno proměnné, dat. typu apod. (int cislo)

### Datový typ

* + Určuje, jaký typ hodnot může obsahovat proměnná – int – celé číslo, string – slovo atd.

### hodnota

* + informace v paměti, která obsahuje hodnotu proměnné – int cislo = 2;

### konstanta

* + identifikátor, který není možné měnit za běhu programu – const int cislo = 2;

### proměnná

* + identifikátor, který je možné měnit za běhu programu – int cislo = 2;

### příkaz

* + instrukce, které se za sebou vykonávají během programu – C# ukončení ;

### výrazy

* + informace, která se v příkazu vyhodnocuje, třeba a + b

### operátory

* + symboly, které reprezentují nějakou operaci
    - aritmetické – + - / \*
      * logické – && ! ||
      * relační – <> == a jejich kombinace

### operandy

* + identifikátory se kterými pracují operátory

### funkce (metoda)

* + podprogram, který je možné opakovaně použít
  + vrací hodnotu

### procedura

* + podprogram, který je možný opakovaně použít
  + reprezentuje se jen jako posloupnost příkazů
  + nevrací hodnotu

### Parametry funkce/procedury

* + vstupní data funkce / procedury
  + zapisuje se do závorek za jméno funkce – public void funkce (string slovo)

# 6. OOP

## Cíl

* + vytvářet malé znovuvyužitelné jednotky – objekty
  + snaha přiblížit programování fungování normálního světa
    - třída – kategorie
    - objekt – věc v kategorii
    - vlastnost – vlastnost věci
    - metoda – udělat nějakou činnost, třeba něco vykreslit

## vytvoření

* + konstruktor – metoda, která je automaticky vytvořena po vzniku třídy
    - obsahuje instrukce které má mít nový objektu
    - v podstatě se tím tvoří nové objekty
    - následující příklad
      * vytvořil jsem třídu „jedna“
      * uvnitř je construktor „jedna“ – construktor musí mít stejné jméno jako třída
      * následně jsem jí pomocí funkce new „vzbudil“ uprostřed Main metody

|  |
| --- |
| *class jedna*  *{*  *Variables*  *…*  *jedna()*  *{*  *//jakakoliv funkce*  *Console.WriteLine(“ahoj”);*  *}*  *public static void Main()*  *{*  *jedna jednaDva = new jedna();*  *//dalsi kod*  *}*  *}* |

## Objekt

* + vlastní entita s vlastnostmi a metodami
  + objekt by měl být natolik soběstačný, že by mělo být možné ho implementovat kdekoliv

### vnitřní stav

* + - popisuje objekt
    - jeho stav se ukládá do datového pole, záznamu či proměnné

## třída

množina objektů s určitými vlastnostmi – např třída auto bude mít objekty kolo, motor, dveře apod.

udává nějaké obecné vlastnosti dané třídy

* + to bude ve formátu třeba jenom int cenaProduktu;
  + nenabývá specifických hodnot

## instance

objekt již v nějaké třídě – to by bylo třeba to kolo

vytvoří se pomocí construktoru

zruší se pomocí destruktoru

## skládání

objekty mohou být součástí dalších objektů

např.

|  |
| --- |
| kruh |
| S: bod |
| r: double |

|  |
| --- |
| bod |
| X: int; |
| Y: int; |

|  |
| --- |
| Funkce |
| souradnice1: bod; |
| souradnice2: bod2;  .  .  . |

## Dědičnost tříd

třídy mohou mezi sebou dědit

dědí se tak že třídy, které jsou „níže“ v generaci budou dědit po třídách „výše“

dědit se mohou atributy nebo metody

může být nahrazeno skládáním

## modifikátory přístupnosti

určují, jak moc viditelné vlastnosti objektu jsou

public (je možné vidět vše odkudkoliv)

private (možné vidět a přistupovat jen metody daného objektu )

protected (je možné je vidět zvenku, ale není možné je modifikovat)

## zapouzdření

zobrazení pouze veřejných vlastností objektu, vnitřek samotný již není třeba znát

vnitřní stav objektu

* + skrytý před uživatelem a jinými částmi programu
  + měnit ho můžou pouze jeho vlastní metody

práce s objekty je možná jen pomocí „Interface“ – metody, které uživatel může používat jsou jenom ty viditelné

## polymorfismus

umožňuje používat jeden „Interface“ pro práci s více objekty

v podstatě jde o to, že vytvoří hlavní třída, kde jsou obecné informace a metody společné pro všechny objekty

polymorfismus umožňuje přepsat určité atributy, tak aby vyhovovaly přesným potřebám

např.

|  |
| --- |
| class Objekty |
| Atribut barva |
| Metoda vykresli |
| SubClass kruh |
| Atribut barva |
| Metoda vykresli – již specifikovaná pro kruh |

|  |
| --- |
| SubClass ctverec |
| Atribut barva |
| Metoda vykresli – již specifikovaná pro ctverec |

## Abstrakce

### abstraktní třída

* + zjednodušení objektu
  + třída ze které není možné vytvořit objekt => objekt je vytvořen dědičností
  + je to předpis pro nějakou konkrétní třídu, která je odvozená pomocí dědičnosti => implementace je až v děděné třídě

### abstraktní metoda

* + metoda o které víme že jí nebudeme využívat zrovna v této třídě, ale spíše jí budeme modifikovat v subclásách, např. metoda vykresli v class Objekty
  + označují se klíčovým slovem abstract nebo virtual

## delegování

pověření objektu na provedení činnosti jiným objektem

## výjimky

chybový stav programu

je nutné je nejdříve ošetřit, než bude program dále pokračovat

příkladem může být třeba dělení nulou, nebo přístup k chráněnému objektu

### aktivní

pomocí if else

### pasivní

pomocí try – catch

## delegáty

předpis pro typ funkce

určuje

* + návratový datový typ
  + počet parametrů

používá se, pokud je nutné pracovat s funkcí jako s daty

* + např. uložení do proměnné

## generiky

obecné datové typy

používají se, pokud nevíme, co má být datovým typem algoritmu

konkrétní datový typ se dosadí až při implementaci algoritmu

## MVC

oddělení logiky od výstupu

model (data) – view (pohledy) – controller (kontrolery)

### model

* + obsahuje veškerou logiku
    - např výpočty
    - nestará se o to, jak bude výstup dat dále zpracován

### view

* + stará se o zobrazení výstupu uživateli

### controller

* + prostředník pro komunikaci mezi modelem, view a uživatelem