Programování

Chyby v programu, ovládání VS, algoritmy a diagramy, datové strukruty

Typová konverze

- Dle datového typu proměnné víme, jaký typ hodnoty můžeme očekávat
- Datový typ nám i omezuje, jakou hodnotu můžeme do proměnné vložit
  - Např. nemůžeme do proměnné, která drží číselnou hodnotu vkládat řetězec. Stejně tak nemůžeme do proměnné, která nám drží celé číslo, vložit číslo desetinné
- Pokud to opravdu silou chceme, používáme **KONVERZE**
- Konverze nám umožňuje změnit datový typ příslušné hodnoty
- V obecnosti má tvar: **(požadovaná hodnota) výraz**
  - Výrazem může být samotná hodnota případně libovolný výpočet nebo volání funkce
- Během konverze může docházet ke ztrátě informace
- Speciálně lze nad objekty a proměnnými volat funkci **ToString()** k převedení na řetězec

Pole [ ]

- Statická homogenní datová struktura
  - umožňuje uchovávat více hodnot stejného datového typu
  - Po stanovení velikosti pole již nelze v průběhu měnit počet prvků
- Dle datového typu, jednotlivých položek pole lze využívat speciální funkce – řazení, suma, průměr, vyhledávání, …
- Pro práci s konkrétní položkou musíme uvést její index (začínáme od 0)

```
double[] novePole = new double[4];

bool[] pravdivostniPole = { true, false, true, true, false };

string[] poleSlov = { "ahoj", "svete", "jsem", "tady" };

int[] poleCisel = new int[3];
poleCisel[0] = 1;
poleCisel[2] = 4;
```

List < >

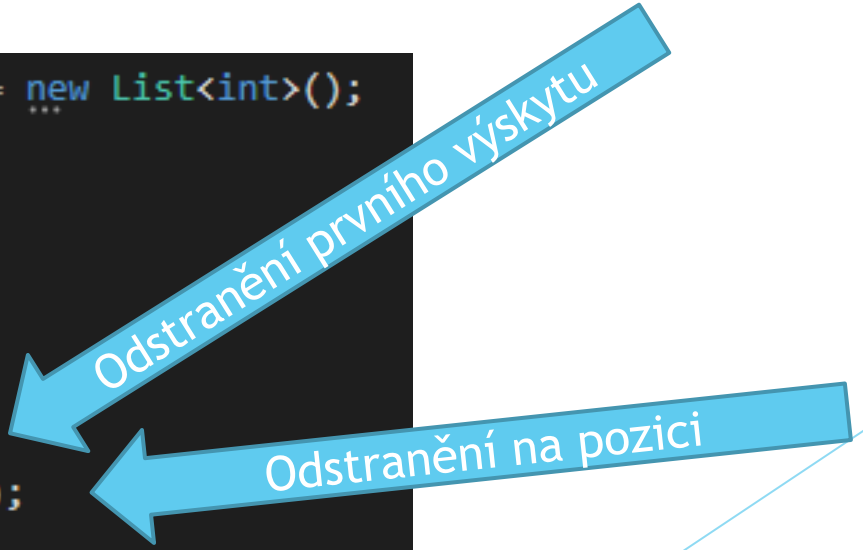
- Dynamická homogenní datová struktura
- Oproti poli nedefinujeme jeho velikost – ta se mění podle počtu prvků
- Pro práci s listem využíváme funkce **Add()** pro přidání a **Remove()** případně **RemoveAt()** pro odebrání – nová položka se přidává vždy na konec
- Pro editaci prvku na příslušném poli využíváme indexace stejně jako u pole
- Pokud nám stačí zjistit hodnotu na příslušném indexu, lze využít funkce **ElementAt()**

```
List<int> listCisel = new List<int>();

listCisel.Add(1);
listCisel.Add(17);
listCisel.Add(1);
listCisel.Add(3);

listCisel.Remove(1);
listCisel.RemoveAt(2);

listCisel[1] = 500;
```



Ošetření výjimek v běhu programu

- Pro kritickou sekci kódu je vhodné mít vymyšleno, co se má stát pokud se vyskytne problém
- **try-catch** funguje jako pískoviště, kde si v sekci **try** necháme proběhnout kód, kde se může objevit výjimka (Exception) v části **catch** pak říkáme, co se má stát
- Mimo výjimky, které již existují lze vyhazovat i výjimky vlastní pomocí **throw**

```
int[] array = { 1, 2, 3, 4 };

try
{
    Console.WriteLine(array[5]);
}
catch (IndexOutOfRangeException e)
{
    Console.WriteLine("Nedostupný index pole");
}

throw new Exception("Stejně vytvoříme chybu");
```

Pole nemá takový index !!!

Jaký je čas?
Čas na Kahoot!

Postup vytváření algoritmu

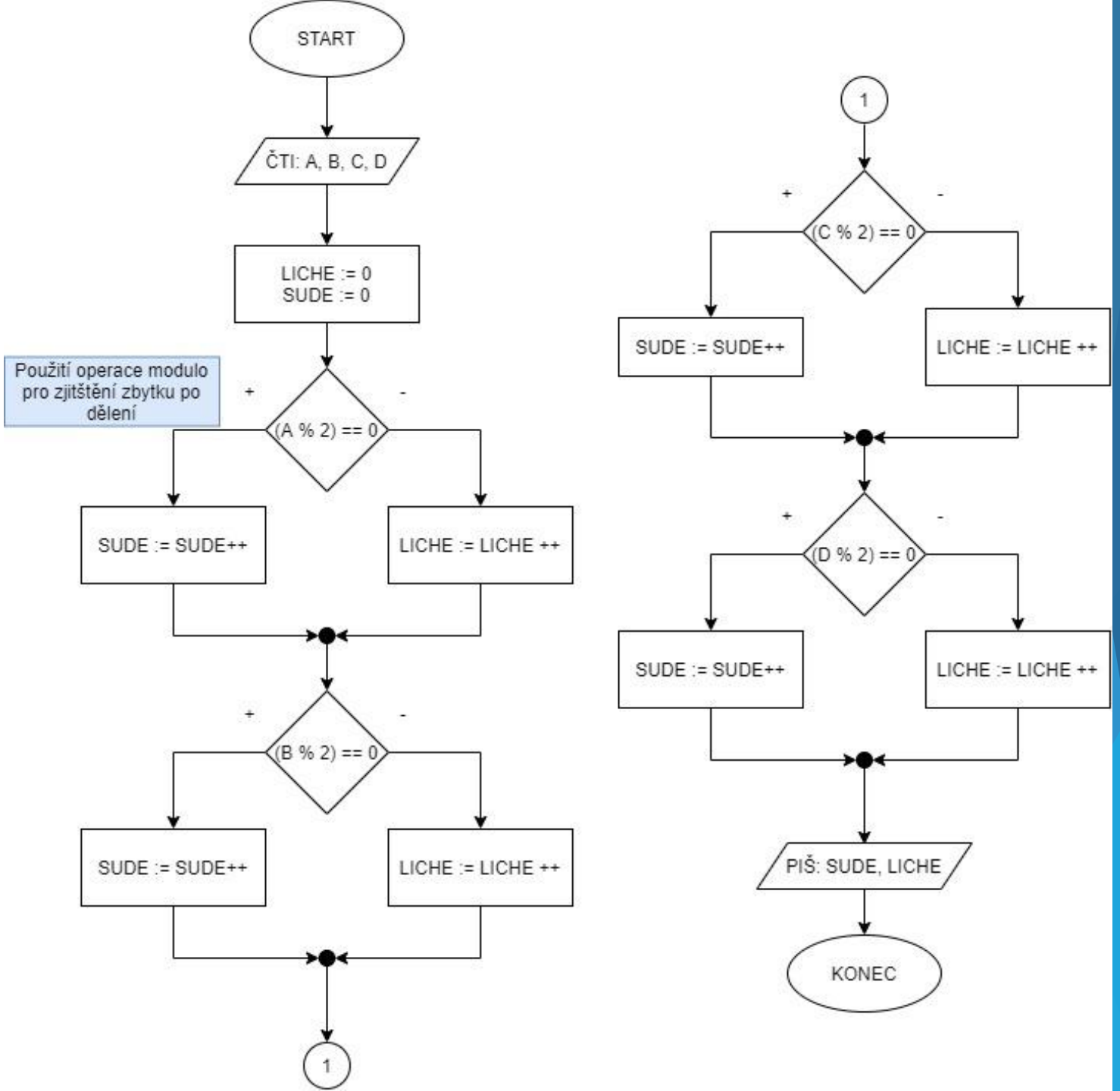
- Formulace problému
  - Formulace požadavků, určení vstupů/výstupu a požadavků na přesnost
- Analýza úlohy
  - Ověření řešitelnosti a počtu řešení dle zadaných vstupů
- Vytvoření algoritmu
  - Sestavení sledu operací, které vedou k požadovanému výsledku
- Sestavení programu
  - Vytvoření zdrojového kódu v příslušném programovacím jazyce
- Odladění programu
  - Odstraňování logických a syntaktických chyb v programu

Přehled používaných symbolů

Používáme i pro vstupní instrukce

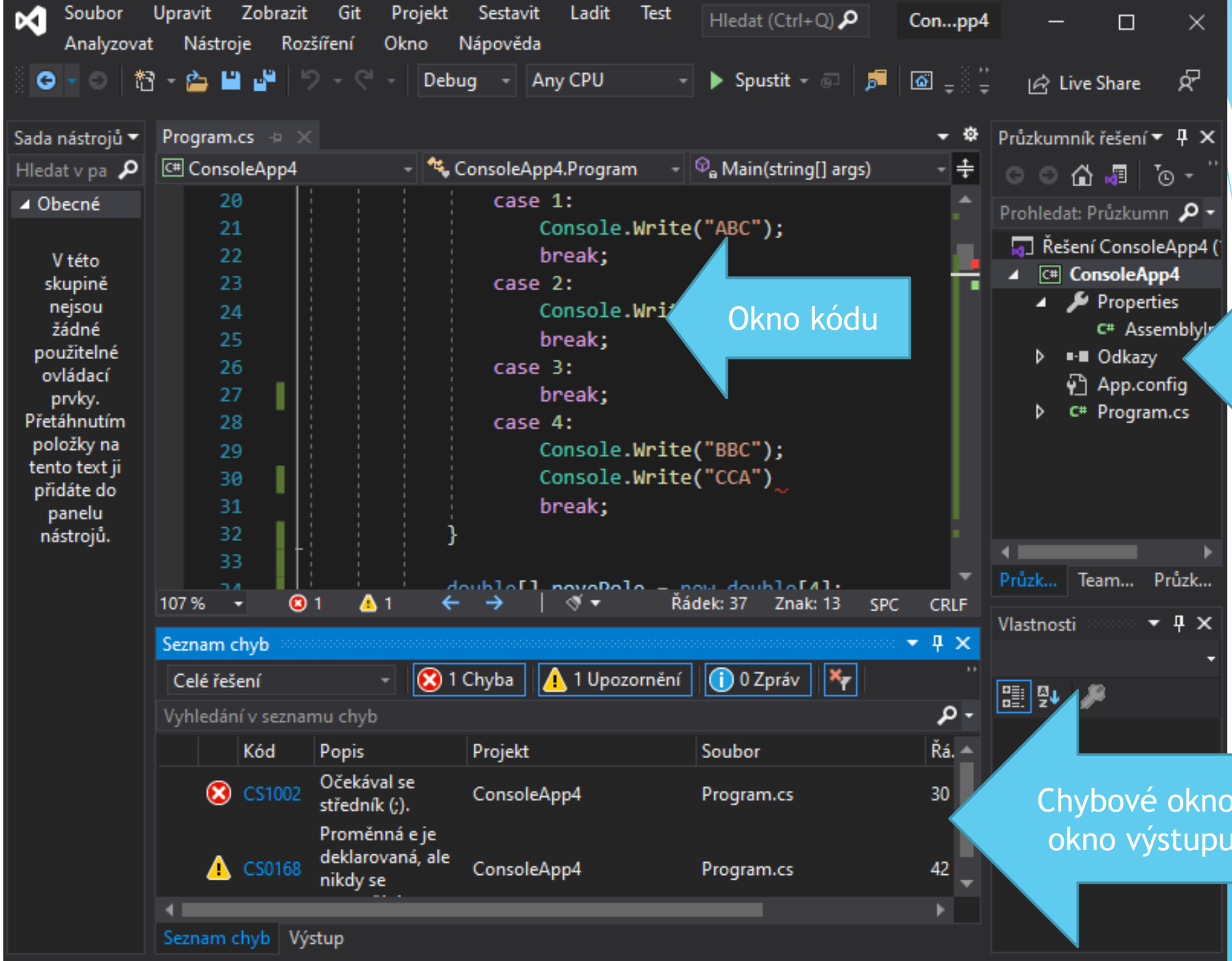
Vývojový diagram

- Vždy obsahuje právě jeden začátek a konec
- Šipky nám pomáhají ve čtení toku programu
- Obsahem rozhodovacích podmínek jsou výrazy, o kterých lze říci zda jsou pravdivé
- Nejlépe užít výrazu stylu: **X < Y** nebo **A == B** apod.
- Proměnné, které v průběhu použijeme uvádíme až v momentě použití



Práce s Visual Studiem

- Dle vytvářeného typu projektu se nám liší rozložení jednotlivých oken
- Rozdíl Konzolová vs. Formulářová aplikace
- Pokud nechceme, aby se nám konzole po doběhnutí programu sama zavřela – spouštíme pomocí CTRL + F5 místo zelené šipky spuštění
- V chybovém okně nás IDE upozorňuje na jakém řádku se chyba nachází včetně informace o jaký typ chyby se jedná
- Pokud si s chybou nevím rady hledám -> řešení je v 99% na internetu
  - Stačí řádek z okna chyb zkopírovat a vložit do vyhledávače
- Pokud se v programu nachází chyby -> kompilátor neumožní spuštění (maximálně nabídne spuštění poslední funkční sestavení), upozornění nevadí
- IDE nám hodně pomáhá (napovídá) co za kód má následovat nebo doporučí možnost úpravy
- Pokud nevíme, jak konkrétní funkce funguje, proklikem si zobrazím dokumentaci


Soubor Upravit Zobrazit Git Projekt Sestavit Ladit Test
Analyzovat Nástroje Rozšíření Okno Nápověda
Hledat (Ctrl+Q)
Con...pp4
Debug
Any CPU
Spustit
Live Share
Sada nástrojů
Obecné
V této skupině nejsou žádné použitelné ovládací prvky. Přetáhnutím položky na tento text ji přidáte do panelu nástrojů.
Program.cs
ConsoleApp4
ConsoleApp4.Program
Main(string[] args)
case 1:
Console.Write("ABC");
break;
case 2:
break;
case 3:
break;
case 4:
Console.Write("BBC");
Console.Write("CCA")
break;
}
Okno kódu
107 %
Řádek: 37
Znak: 13
SPC
CRLF
Průzkumník řešení
Prohledat: Průzkumn
Řešení ConsoleApp4
ConsoleApp4
Properties
Odkazy
App.config
Program.cs
Struktura projektu
Průzk...
Team...
Průzk...
Vlastnosti
Seznam chyb
Celé řešení
1 Chyba
1 Upozornění
0 Zpráv
Vyhledání v seznamu chyb
Kód
Popis
Projekt
Soubor
Řá.
CS1002
Očekával se středník (;).
ConsoleApp4
Program.cs
30
CS0168
Proměnná e je deklarovaná, ale nikdy se
ConsoleApp4
Program.cs
42
Chybové okno, okno výstupu
Seznam chyb
Výstup