Standartinės įvesties ir išvesties komandos konsolėje ir debuginimas

Šios pamokos tikslas yra susipažinti su bazinėmis C# kalbos struktūromis ir įvesties bei išvesties funkcijomis konsolės aplinkoje.

Užduotis

Parašyti Windows konsolės programą, kuri leistų įvesti skaičių sąrašą ir apskaičiuotų šių skaičių vidurkį.

Programavimo darbo esmė yra suvokti duotą problemą, ją išanalizuoti ir suskaidyti į mažesnes problemas tol kol gaunamos komandos, kurias jau galima užkoduoti kaip komandas kompiuteriui. Gera praktinė taisyklė yra pabandyti suskaidyti visą problemą į dvi ar tris smulkesnes, tada kiekvieną iš jų suskaldyti į dvi ar tris smulkesnes, ir taip toliau. Tol, kol jau gaunamos kompiuteriui suprantamos komandos.

C# kalbos struktūros, kurių jums prireiks šiai užduočiai

Tipai

```
int count = 0;
decimal sum = 0;
string a = "value";
```

int - sveikieji skaičiai. Reikšmių diapazonas nuo -2 147 483 648 iki 2 147 483 647
decimal - realieji skaičiai skirti dešimtainei skaičiavimo sistemai. Reikšmių diapazonas -7,9 x 10^{28} iki 7,9 x 10^{28}, tikslumas 28 skaičiai po kablelio.
string - simbolių eilutė, talpinanti Unicode simbolius. Unicode - tarptautinis kodavimo standartas apimantis 109 242 dažniausiai naudojamus simbolius. Ilgis yra neribotas, tiksliau - tiek kiek telpa į kompiuterio atmintį.

Įvesties ir išvesties funkcijos konsolei

string variable = **Console.ReadLine()**;
Nuskaito vieną eilutę iš konsolės į kintamąjį. Įvedant eilutę reikia įvesti norimus simbolius ir paspausti Enter. Enter simbolis nebus įvestos eilutės dalis.

Console.WriteLine(expression**)**;

Pateiktą išraišką paverčia į simbolių eilutę ir išveda ją į konsolę. Gale prideda Enter simbolį, t.y. nuveda kursorių į kitą eilutę.

Išraiškų konvertavimas

int variable = **int.Parse(**stringExpression**)**;
Konvertuoja simbolių eilutę į **int** tipo kintamąjį. Jei eilutė talpina simbolius, kurie negali būti konvertuoti į skaičių, tai programa išmes klaidą (angl. “exception”).

decimal variable = **decimal.Parse(**stringExpression**)**;
Veikimas toks pats kaip ir **int** atveju, bet rezultatas yra **decimal** tipo kintamasis.

Masyvai

decimal[] numbers = **new decimal[**intExpression**]**;
Masyvas tai sąrašas reikšmių. Šios reikšmės gali būti įrašomos ir nuskaitomos naudojant masyvo indeksą. Indeksai gali turėti reikšmes nuo 0 iki masyvo ilgio, jo neįskaitant. Pvz. masyvo kurio ilgis yra 3 validūs indeksai yra - 0, 1 ir 2. Sukuriant masyvą yra nurodomas reikšmių tipas ir jo ilgis. Masyvo ilgio negalima keisti. Jei visdėlto ilgį norima pakeisti, tai reikia kurti naują masyvą.

int length = numbers**.Length**;
Masyvo ilgis.

numbers[index] = decimalExpression;
Reišmės įrašymas į masyvą.

number **= numbers[index]**;
Reikšmės nuskaitymas iš masyvo.

Ciklai

```
for (int i = 0; i < count; i++)
{
    … // expressions
}
```

Ciklas tai kalbos struktūra, leidžianti kartoti norimas komandas tol kol ciklo išraiška tai leidžia.
Ciklas **for** turi tris išraiškas.
int i = 0 - pirma išraiška, nurodo pradines sąlygas.
i < count - antra išraiška, nurodo sąlygą kuri turi būti įvykdoma, kad ciklas toliau veiktų.
i++ - trečia išraiška, nurodo veiksmus kurie turi būti atliekami po kiekvieno ciklo. Šie veiksmai turi keisti antrąją išraišką taip, kad ji kada nors užsibaigtų. Kitu atveju gausis situacija - amžinas ciklas.

Kodo sąlyginio vykdymo komanda - If

```
if (numbers.Length > 0)
{
   … // expressions
}
```

If struktūra vykdo norimas komandas tik tada jei išraiška tai leidžia. Pavyzdyje komandos, kurios bus parašytos tarp figūrinių skliaustelių, bus vykdomos tik tada jei masyvo ilgis bus didesnis nei 0.

Funkcijos

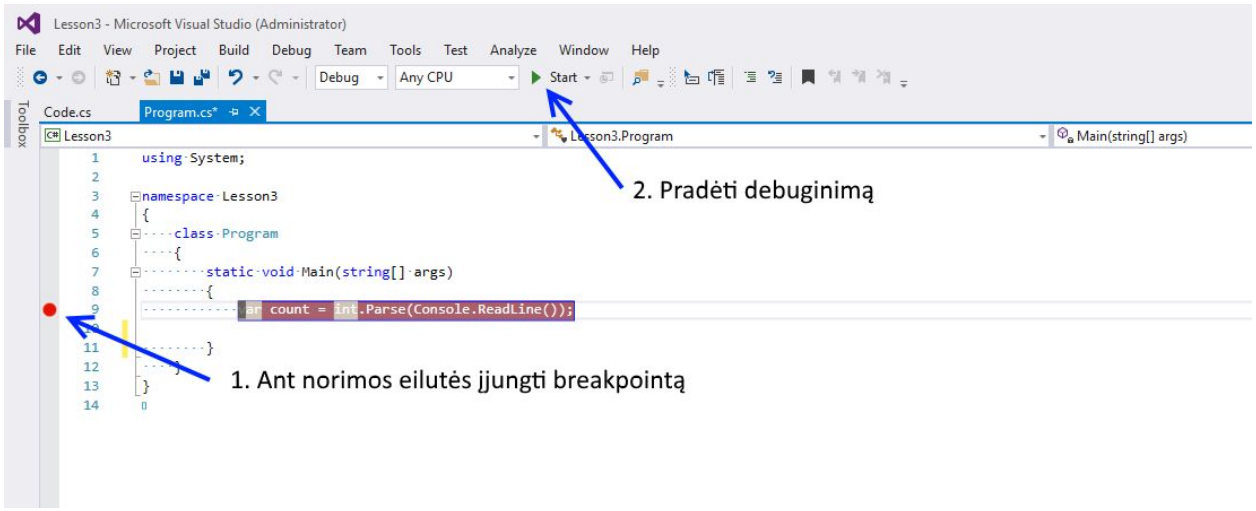
```
private static decimal CalculateAverage(decimal[] numbers)
{
      ...
      return decimalExpression;
}
```

Funkcijos apima grupę sakinių, kurie bus įvykdyti, kai bus iškviesta funkcija. Jos leidžia patogiau struktūrizuoti programą taip, kad kodas būtų suprantamesnis. Funkcijos gali turėti vieną ar daugiau parametrų (pavyzdyje - **decimal[] numbers**), ir turi turėti rezultato tipą (pavyzdyje - **decimal**), bei rezultato grąžinimo komandą **return** (pavyzdyje - **return decimalExpression;**). Rezultato tipas gali būti - **void**, kas reiškia kad rezultato nėra. Tokiu atveju **return** komanda gali būti, bet nėra privaloma.

Debuginimas

Debuginimas tai programos kodo vykdymo eigos sekimas. Tai yra dažnas būdas ieškoti programos kodo klaidų ar suprasti kaip iš tiesų veikia programa.

Debuginimas pradedamas ant norimos eilutės įjungus breakpointą ir startuojant programą debuginimo režimu.



Programa startuoja ir yra vykdoma tol kol pasiekiamas pirmas breakpointas.

Bazinės debuginimo galimybės pavaizduotos žemiau.