Употреба основних типова података

Пример:

```
static void Main(string[] args)
{
    int mojBroj;
    string mojString;
    mojBroj = 17;
    mojString = "\"mojBroj\" je";
    Console.WriteLine($"{mojString} {mojBroj}");
    Console.ReadKey();
}
```

Даје:

"mojBroj" je 17

Press any key to continue . . .

Линије кодова: `int mojBroj;string mojString;` су линије декларације промењивих које се користе у коду.

Декларисана је промењива mojBroj типа int и промењива mojString типа string.

Следеће две линије кода додељују вредности овим промењивима: `mojBroj = 17;` `mojString = "\"mojBroj\" je";` .

Овде се додељују две литералне (фиксне) вредности промењивима коришћењем оператора доделе =.

Промењива mojString добија вредност стринга (“mojBroj” je) и пошто су и наводници део тог стринга користе се косе црте за поставку ескејп карактера.

У линији кода: `Console.WriteLine($"{mojString} {mojBroj}");` знак $ испред наводника се користи за Стринг Интерполацију.

На овај начин садржај у стрингу смештен у витичасте заграде није стринг литерал већ се понаша као промењива која има неку вредност па ће се та вредност појавити на конзоли као да је стринг литерал.

Линија `Console.ReadKey();` паузира даље извршавање кода све док корисник не притисне дугме ENTER на тастатури.

Манипулација промењивима помоћу аритметичких оператора

Пример:

```
static void Main(string[] args)
{
    double x, y;
    string ime;
    Console.WriteLine("Unesi tvoje ime:");
    ime = Console.ReadLine();
    Console.WriteLine($"Zdravo {ime}!");
    Console.WriteLine("Uneti prvi broj:");
    x = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine("Uneti drugi broj:");
    y = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine($"Zbir {x} i {y} je " + $"{x + y}.");
    Console.WriteLine($"Razlika {x} i " + $"{y} je {x - y}.");
    Console.WriteLine($"Proizvod {x} i {y} " + $"je {x * y}.");
    Console.WriteLine($"Kolicnik {x} i " + $"{y} je {x / y}.");
    Console.WriteLine($"Ostatak deljenja {x} i " + $"{y} je {x % y}.");
    Console.ReadKey();
}
```

Даје:

```
Unesi tvoje ime:
JA
Zdravo JA!
Uneti prvi broj:
12
Uneti drugi broj:
5
Zbir 12 i 5 je 17.
Razlika 12 i 5 je 7.
Proizvod 12 i 5 je 60.
Kolicnik 12 i 5 je 2.4.
Ostatak deljenja 12 i 5 je 2.
Press any key to continue . . .
```

Са линијом кода `ime = Console.ReadLine();` се од корисника тражи унос преко тастатуре па се унешен податак смешта у стринг промењиву ime.

Линија `ime = Console.ReadLine();` генерише стринг па у овом делу кода нема проблема пошто се као име корисника очекује стринг.

Међутим, у линији кода која тражи од корисника унос бројчане вредности, стринг као унета вредност нема смисла, па се мора извршити одмах и конверзија типа података.

Прво су промењиве x и y декларисане као типа double: `double x, y;`.

Са линијом кода `x = Convert.ToDouble(Console.ReadLine());` добијени податак од стране корисника је типа стринг па се конвертује у тип double и додељује промењивој x.

Остатак кода користи Стринг Интерполацију за приказ резултата различитих аритметичких операција.

Декларисање локалних промењивих са имплицитно одређеним типом података

Од компајлера програмског језика C# се може тражити да сам одреди тип промењиве приликом декларације и да га тако прилагоди типу податка који се смешта у ту промењиву.

```
var broj = 99;
var tekst = "String";
```

За промењиве broj и tekst се каже да су промењиве са имплицитно одређеним типом.
Кључна реч var наводи компајлер да тип промењиве утврди из типа израза у иницијализацији.
Али, касније у програму се овим промењивима не може доделити вредност неког другог типа.
Такође, службена реч var се не може користити над промењивом која је већ декларисана у коду.

Остатак дељења код double

За разлику од других програмских језика, где се не могу користити операнди типа double или float над операцијом остатка дељења, C# допушта да резултат не буде цео број.
Нпр 7.0 % 2.4 даје 2.2.

Бесконачне вредности

Обично дељење нулом доводи до грешке, али типови double и float имају посебну вредност која може да представи бесконачност, па је вредност 5.0 / 0.0 једнака infinity.
Изузетак овог правила је 0.0 / 0.0 које истовремено је једнако и нула и бесконачно па је та вредност NaN (скраћеница од “not a number“).

Конверзија типова података

Постоје две форме конверзије типова података: имплицитна и експлицитна конверзија.

Имплицитна конверзија

Синтакса: promenjiva1 = promenjiva2;
На овај начин се имплицитно мења тренутни тип података promenjiva2 у тип података promenjiva1, под условом да су то типови података који су дозвољени за имплицитну конверзију.
Пример:

```
ushort prom1;
char prom2 = 'a';
prom1 = prom2;
Console.WriteLine($"promenjiva2: {prom2}");
Console.WriteLine($"promenjiva1: {prom1}");
```

Даје:
promenjiva2: a
promenjiva1: 97

Иако обе промењиве имају запамћене исте информације, ти подаци се интерпретирају на другачије начине због начина на који се представљају различити типови података.
Једноставно правило за имплицитну конверзију: било који тип података чији опсег могућих вредности се потпуно уклапа у опсег могућих вредности другог типа података се може имплицитно конвертовати у тај други тип података.

Експлицитна конверзија

Експлицитна конверзија података настаје када се експлицитно тражи од компајлера да конвертује вредност из једног типа података у други тип података.
Један начин јесте спровођење кастовања (cast) што значи форсирање конверзије:
(ciljniTipPodataka) promenjivaPocetnogTipa
Овима се конвертује вредност из promenjivaPocetnogTipa у тип података ciljniTipPodataka.
Многи типови података нису компатабилни па кастовање изазива или поруку о грешци или се кастовање изврши али добијени подаци нису тачни иако није пријављена грешка током конверзије.
Најчешћи начин експлицитне конверзије је употреба команди попут Convert.ToDouble() али и овај начин конверзије може довести до појаве грешке током провере рада компајлера.

Задаци за самосталан рад

1. У следећем коду, како ће се прићи имену slavan унутар именског простора fenomenalan?

```
namespace izvanredan
{
    //kod u imenskom prostoru izvanredan
}
namespace super
{
    namespace fenomenalan
    {
        //definisano slavan
    }
}
```

2. Написати код којим се омогућава унос четири цела позитивна броја и исписивање њиховог производа на конзоли.

3. Изменити код из задатка 2 тако да се користе следеће претпроцесорске команде:

```
using static System.Console;
using static System.Convert;
```