## Авторски колектив

**Мартин Ивов Янков**

Контакти: e-mail: m.i.yankov@gmail.com

**Мартин Лилов Нинов**

Контакти: e-mail: martin.ninov@gmail.com

**Йоана Иванова**

Контакти: e-mail: joana\_ivanova@abv.bg

**Ради Радев**

Контакти:

# Глава 2. Примитивни типове и променливи

|  |  |
| --- | --- |
| **Задача 1. Най-подходящ целочислен тип данни** | |
| **Условие**  Напишете програма, която при зададено цяло число да показва на конзолата най-подходящия тип данни за него: sbyte, byte, short, ushort, int, uint, long и ulong. Най-подходящ тип данни в случаят означава да не съществува друг „по-малък“ тип данни, който да може да съхрани числото. | |
| **Описание на входа**  Входът се въвежда на един ред. На единстения ред ще бъде въведено цяло число от  -9,223,372,036,854,775,808 до + 18,446,744,073,709,551,615. Входните данни ще бъдат винаги в коректен формат и не трябва да бъдат проверявани. | |
| **Описание на изхода**  Изходът се извежда на един ред. На единственият ред се извеждат с малки букви един от следните типове данни: **sbyte**, **byte**, **short**, **ushort**, **int**, **uint**, **long** и **ulong.** | |
| **Анализ на задачата**  Дефинираме променлива от тип decimal и присвояваме прочетеното от конзолата на нея. Правим това, тъй като в началото не можем да сме сигурни дали подадената стойност е отрицателна или е по-голяма от максималната стойност на типа long (9,223,372,036,854,775,807), а decimal има много по-голям „обсег“ от стойности.  С 1 “if” проверяваме дали стойността на числото е по-голяма или равна на нула. Ако е – с 8 вътрешни “if else”-а проверяваме най-подходящият тип данни за числото. Започваме от горната граница. Използваме свойствата на класовете типове данни .MaxValue и .MinValue   * Ако числото е по-голямо от максималната стойност на long, значи е ulong * Ако числото е по-голямо от максималната стойност на uint, значи е long * Ако числото е по-голямо от максималната стойност на int, значи е uint * Ако числото е по-голямо от максималната стойност на ushort, значи е int * Ако числото е по-голямо от максималната стойност на short, значи е ushort * Ако числото е по-голямо от максималната стойност на byte, значи е short * Ако числото е по-голямо от максималната стойност на sbyte, значи е byte * Ако и последното не е вярно, то най-подходящата стойност е sbyte   Аналогично ако числото по-малко от нула, но проверките са за по-малко от минималната стойност на съответните типове данни. | |
| **Решение (сорс код)** | | |
| using System;  namespace WholeNumbersDataTypes  {  class Program  {  static void Main(string[] args)  {  string input = Console.ReadLine();  decimal number = decimal.Parse(input);  if (number >= 0)  {  if (number > Int64.MaxValue)  {  Console.WriteLine("ulong");  }  else if (number > UInt32.MaxValue)  {  Console.WriteLine("long");  }  else if (number > Int32.MaxValue)  {  Console.WriteLine("uint");  }  else if (number > UInt16.MaxValue)  {  Console.WriteLine("int");  }  else if (number > Int16.MaxValue)  {  Console.WriteLine("ushort");  }  else if (number > Byte.MaxValue)  {  Console.WriteLine("short");  }  else if (number > SByte.MaxValue)  {  Console.WriteLine("byte");  }  else  {  Console.WriteLine("sbyte");  }  }  else  {  if (number < Int32.MinValue)  {  Console.WriteLine("long");  }  else if (number < Int16.MinValue)  {  Console.WriteLine("int");  }  else if (number < SByte.MinValue)  {  Console.WriteLine("short");  }  else  {  Console.WriteLine("sbyte");  }  }  }  }  } | | |
| **Тестове**  Интересни случаи за тестване:   * Максималната стойност от заданието * Минималната стойност от заданието * Положителна стойност, която да връща long * Стойност, която да връща uint * Положителна стойност, която да връща int * Отрицателна стойност, която връща int * Стойност, която да връща ushort * Положителна стойност, която да връща short * Отрицателна стойност, която връща short * Стойност, която да връща byte * Положителна стойност, която да връща sbyte * Отрицателна стойност, която връща sbyte | | |
| **Вход** | **Изход** | |
| 18446744073709551615 | ulong | |
| **Вход** | **Изход** | |
| -9223372036854775808 | long | |
| **Вход** | **Изход** | |
| 9223372036854775807 | long | |
| **Вход** | **Изход** | |
| 4294967295 | uint | |
| **Вход** | **Изход** | |
| 2147483647 | int | |
| **Вход** | **Изход** | |
| -2147483648 | int | |
| **Вход** | **Изход** | |
| 65535 | ushort | |
| **Вход** | **Изход** | |
| 32767 | short | |
| **Вход** | **Изход** | |
| -32768 | short | |
| **Вход** | **Изход** | |
| 255 | byte | |
| **Вход** | **Изход** | |
| 127 | sbyte | |
| **Вход** | **Изход** | |
| -128 | sbyte | |

|  |  |
| --- | --- |
| **Задача 2. Най-подходящ тип данни за числа с плаваща запетая** | |
| **Условие**  Напишете програма, която при зададено нецяло число(десетична дроб) да показва на конзолата най-подходящия от типовете float, double или decimal. Най-подходящ тип данни в случаят означава да не съществува друг „по-малък“ тип данни, който да може да съхрани числото. | |
| **Описание на входа**  Входът се въвежда на един ред. На единстения ред ще бъде въведено нецяло число(десетична дроб) от 0 до 10, с най-много 27 знака след десетичната запетая. Плаващата запетая ще бъде точка. Входните данни ще бъдат винаги в коректен формат и не трябва да бъдат проверявани. | |
| **Описание на изхода**  Изходът се извежда на един ред. На единственият ред се извеждат с малка буква един от следните типове данни: float, double и decimal. | |
| **Анализ на задачата**  Четем входа в променлива от тип стринг. Декларираме нова променлива от тип byte, в която ще държим броя на цифрите след плаващата запетая.  Използваме свойството на стринговете .Length, което връща общият брой символи и метода .IndexOf(char symbol), за да намерим индекса на точката в стринга. Вадим двете и добавяме 1, за да започнем да броим от следващият символ след точката и присвояваме резултата на променливата, която държи броя на цифрите след точката.  С if-else конструкция проверяваме последователно:   * Ако цифрите са по-малко от 7 -> float * Ако цифрите са по-малко от 16 -> double * Ако не -> decimal | |
| **Решение (сорс код)** | | |
| using System;  namespace FloatingPointNumbersBestDataType  {  class Program  {  static void Main(string[] args)  {  string input = Console.ReadLine();  byte digitsAfterFloatingPoint =  (byte)(input.Length - (input.IndexOf('.') + 1));  if (digitsAfterFloatingPoint < 8)  {  Console.WriteLine("float");  }  else if (digitsAfterFloatingPoint < 16)  {  Console.WriteLine("double");  }  else  {  Console.WriteLine("decimal");  }  }  }  } | | |
| **Тестове**  Интересни случаи за тестване:   * Число със 7 цифри след запетаята * Число с 15 цифри след запетаята * Число с 27 цифри след запетаята * Число без цифри след запетаята * Число с 8 цифри след запетаята * Число с 18 цифри след запетаята | | |
| **Вход** | **Изход** | |
| 5.1234567 | float | |
| **Вход** | **Изход** | |
| 3.1234 | float | |
| **Вход** | **Изход** | |
| 7.100 | float | |
| **Вход** | **Изход** | |
| 6.123456789012345 | double | |
| **Вход** | **Изход** | |
| 9.123456789012 | double | |
| **Вход** | **Изход** | |
| 1.123456789012345678901234567 | decimal | |
| **Вход** | **Изход** | |
| 1.12345678901234567890 | decimal | |
| **Вход** | **Изход** | |
| 9 | float | |
| **Вход** | **Изход** | |
| 5.12345678 | double | |
| **Вход** | **Изход** | |
| 6.123456789012345678 | decimal | |

|  |  |
| --- | --- |
| **Задача 3. Точност на променливи с плаваща запетая** | |
| **Условие**  Напишете програма, която събира вярно променливи с плаваща запетая с точност до 0.000001. | |
| **Описание на входа**  Входът се въвежда на два ред. На всеки ред ще бъде въведено нецяло число(десетична дроб) от 0 до 10, с най-много 15 знака след десетичната запетая. Входните данни ще бъдат винаги в коректен формат и не трябва да бъдат проверявани. | |
| **Описание на изхода**  Изходът се извежда на един ред. На единственият ред се извежда сборът на двете с точност точно шест цифри след запетаята. | |
| **Анализ на задачата**  Запазваме входните данни в тип decimal, тъй като този тип се ползва, когато искаме да сме сигурни в точността. Извеждаме резултата от сбора на двете числа като използваме конструкцията {0:0.000000} в плейсхолдера, за да го изведем с шест цифри след запетаята. | |
| **Решение (сорс код)** | | |
| using System;  namespace FloatingNumbersPrecision  {  class Program  {  static void Main(string[] args)  {  string input = Console.ReadLine();  decimal firstNumber = decimal.Parse(input);  input = Console.ReadLine();  decimal secondNumber = decimal.Parse(input);  decimal result = firstNumber + secondNumber;  Console.WriteLine("{0:0.000000}", result);  }  }  } | | |
| **Тестове**  Интересни случаи за тестване   * Сбор на две цели числа * Сбор на две числа със 6 цифри след запетаята * Сбор на две числа с 8 цифри след запетаята * Сбор на две числа с 15 цифри след запетаята * Сбор на две числа с различен брой цифри след запетаята | | |
| **Вход** | **Изход** | |
| 9  7 | 16.000000 | |
| **Вход** | **Изход** | |
| 4.34  7.85 | 12.190000 | |
| **Вход** | **Изход** | |
| 2.5678  8.9234 | 11.491200 | |
| **Вход** | **Изход** | |
| 6.482391  8.402381 | 14.884772 | |
| **Вход** | **Изход** | |
| 2.94718562  7.59437684 | 10.541562 | |
| **Вход** | **Изход** | |
| 5.123144532343  7.342353445256 | 12.465498 | |
| **Вход** | **Изход** | |
| 4.234456854345819  7.593480234783402 | 11.827937 | |
| **Вход** | **Изход** | |
| 6.192074934723942  5.231 | 11.423075 | |
| **Вход** | **Изход** | |
| 8.32425  1.345259890427394 | 9.669510 | |
| **Вход** | **Изход** | |
| 7.3423425325  9.3453453454234 | 16.687688 | |

|  |  |
| --- | --- |
| **Задача 4. Инициализиране на променлива от тип** int **в шестнадесетичен формат** | |
| **Условие**  Инициализирайте променлива hexNumber от тип int и й присвоете стойност 00FE(шестнайсетичен формат). Напишете програма, която при въведена целочислена стойност в шестнайсетичен формат, да сумира стойността й с тази на вече инициализираната променлива и да извежда на конзолата резултата в десетичен вид. | |
| **Описание на входа**  Входът се въвежда на един ред. На първия ред ще бъде въведено цяло число от 0 до 65535 в шестнайсетичен формат. Входните данни ще са винаги представени с четири знака, като ако няма нужда от четирите, то се добавят нули отпред. Входните данни ще бъдат винаги в коректен формат и не трябва да бъдат проверявани. | |
| **Описание на изхода**  Изходът се извежда на един ред. На единственият ред се извежда стойността в десетичен формат. Пример:  **Вход:**  00FE  **Изход:**  508 | |
| **Анализ на задачата**  Инициализираме променливата и присвояваме стойността с конструктора използващ „0x” пред шестнайсетичното число. Четем входа и използваме класът и методът в .NET Convert.ToInt32(string number, int fromBase), който върши точно исканото от задачата. Накрая извеждаме сумата. | |
| **Решение (сорс код)** | | |
| using System;  namespace IntegerInHexadecimalFormat  {  class Program  {  static void Main(string[] args)  {  int hexNumber = 0x00FE;  string input = Console.ReadLine();  int number = Convert.ToInt32(input, 16);  Console.WriteLine(number + hexNumber);  }  }  } | | |
| **Тестове**  Интересни случаи за тестване   * Примера от заданието * Вход 0000 * Вход 000А * Вход 00BB * Вход 0CCC * Вход FFFF * Вход D000 * Вход 9900 * Вход 8880 * Вход с различни символи | | |
| **Вход** | **Изход** | |
| 00FE | 508 | |
| **Вход** | **Изход** | |
| 0000 | 254 | |
| **Вход** | **Изход** | |
| 000A | 264 | |
| **Вход** | **Изход** | |
| 00BB | 441 | |
| **Вход** | **Изход** | |
| 0CCC | 3530 | |
| **Вход** | **Изход** | |
| FFFF | 65789 | |
| **Вход** | **Изход** | |
| D000 | 53502 | |
| **Вход** | **Изход** | |
| 9900 | 39422 | |
| **Вход** | **Изход** | |
| 8880 | 35198 | |
| **Вход** | **Изход** | |
| 9F7C | 41082 | |

|  |  |
| --- | --- |
| Задача 6. PrintNameAndSex | |
| **Условие**  Напишете програма, която чете от конзолата името на потребителя и неговия пол. Декларирайте променлива isMale от тип bool и присвоете стойността на последната в зависимост от въведения пол. Отпечатайте на конзолата името и “true”/ “false” в зависимост от въведения пол. | |
| **Описание на входа**  На първия ред се въвежда името на потребителя, на втория неговия пол(съответно female,ако е жена и male, ако е мъж). | |
| **Описание на изхода**  Отпечатайте: името ismale? – true или false | |
| **Анализ на задачата**  Четат се двата реда от конзолата, ако въведения пол е „male” на променливата isMale се присвоява „true”.Резултата се отпечатва. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace IsMale  {  class Program  {  static void Main(string[] args)  {  string name = Console.ReadLine();  string sex = Console.ReadLine();  bool isMale = false;  if (sex == "male")  {  isMale = true;  }  Console.WriteLine(name + " isMale? - " + isMale);  }  }  } | |
| **Тестове**  Направени са два базови теста,в зависимост от пола. | |
| **Вход** | **Изход** |
| Joana Ivanova  female | Joana Ivanova ismale? - False |
| **Вход** | **Изход** |
| Stefan Uzunov  male | Stefan Uzunov ismale? - True |

|  |  |
| --- | --- |
| **Задача 7: Конкатенация в Object** | |
| **Условие**  Напишете програма, която да чете две стойности от конзолата от тип string. Декларирайте променлива от тип object. Присвоете на тази променлива стойността, която се получава от конкатенацията на двете стрингови променливи (добавете интервал, ако е необходимо). Отпе­чатайте променливата от тип object. | |
| **Описание на входа**  Входните данни се въвеждат на два реда. На първият ред се въвежда първата стойност от тип стринг. На вторият ред – втората стойност от тип стринг. Двете стойности са думи на английски език, които образуват словосъчетание. Данните са винаги в коректен формат като е възможно да има допълнителни интервали към думите (не помежду буквите). Възможно е някоя от думите да е празен string.  **Пример:**  Beautiful  sun | |
| **Описание на изхода**  Изходните данни се извеждат на един ред. На единственият ред се принтира стойността на пронеливата от тип object като между двете думи трябва да има само един интервал (ако е нужно).  **Пример:**  Beautiful  sun  Beautiful sun | |
| **Анализ на задачата**  Декларираме три променливи, две от тип string и една от тип object. С двете от тип стринг четем двата последователни реда от конзолата. Присвояваме на променливата от тип object стойност чрез метода String.Format(); като използваме и метода Trim(); за да премахнем празните места пред и зад текста. | |
| Решение (Сорс код) | |
| using System;  class Concatenation  {  static void Main(string[] args)  {  string firstWord = "";  string secondWord = "";  object concatenation = null;  firstWord = Console.ReadLine();  secondWord = Console.ReadLine();  if (firstWord.Trim().Length == 0)  {  concatenation = secondWord;  }  else  {  concatenation = String.Format("{0} {1}", firstWord.Trim(),secondWord.Trim());  }  Console.WriteLine(concatenation);  }  } | |
| **Тестове**  Интересни случаи за тестване   * Даденият пример в условието * Поредица от цифри без допълнителни интервали * Първата дума е с интервали отпред * Втората дума е с интервали отпред * Първата дума е с интервали отзад * Втората дума е с интервали отзад * И двете думи са с интервали отпред и отзад * Първата дума да е празен string * Две дълги думи без интервали * Две дълги думи с интервали | |
| **Вход** | **Изход** |
| “Beautiful”  “sun” | Beautiful sun |
| **Вход** | **Изход** |
| “1234567890”  “0987654321” | 1234567890 0987654321 |
| **Вход** | **Изход** |
| “ Wonderful”  “day” | Wonderful day |
| **Вход** | **Изход** |
| “Fantastic”  “ light” | Fantastic light |
| **Вход** | **Изход** |
| “Mighty “  “magic” | Mighty magic |
| **Вход** | **Изход** |
| “Night”  “watch “ | Night watch |
| **Вход** | **Изход** |
| “ Count “  “ Dracula “ | Count Dracula |
| **Вход** | **Изход** |
| “ ”  “empty” | empty |
| **Вход** | **Изход** |
| e7test9in.txt | E7test9out.txt |
| **Вход** | **Изход** |
| e7test10in.txt | e7test10out.txt |

|  |  |
| --- | --- |
| Задача 9. StringUse | |
| **Условие**  Напишете програма, която чете от конзолата текст, съдържащ кавички. Декларирайте две променливи от тип string и им присвоете стойността на въведения вход като в едната променлива използвате quoted string,а в другата недейте. Отпечатайте на конзолата стойностите на променливите, така че да се вижда начина на декларацията на кавичките,табулациите и нови редове. | |
| **Описание на входа**  Входът може да съдържа табулации, единични и двойни кавички, нови редове.  Краят му се обозначава с въвеждане на точка и натискане на бутона “Enter”. | |
| **Описание на изхода**  На първият ред трябва да е показан първият начин за избягване на дефиниране на кавички, табулации, нови редове.  На вторият ред съответно вторият начин. | |
| **Анализ на задачата**  Създаваме два StringBuilder-a, в които да запазваме двата ресултата и докато не се въведе „.” и „Enter”. Прочитаме входа символ по символ и когато срещнем нов ред,табулация или кавичка за първия начин използваме символа за escaping ‘\’, а другият начин за деклариране е като преди стринга се впише @. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_2\_QuotedString  {  class Program  {  static void Main(string[] args)  {  StringBuilder escapedQuotes = new StringBuilder();  StringBuilder escapedQuotesAsQuotedString = new StringBuilder();  escapedQuotesAsQuotedString.Append("@\"");  char key = ' ';  while (key != '.')  {  key = (char)Console.Read();  if (key == '\\' || key == '\"' || key == '\'')  {  escapedQuotes.Append("\\");  escapedQuotes.Append(key);  if (key == '\"')  {  escapedQuotesAsQuotedString.Append("\"");  // escapedQuotesAsQuotedString.Append(key);  }  if (key == '\'')  {  escapedQuotesAsQuotedString.Append("\'");  // escapedQuotesAsQuotedString.Append(key);  }  }  else if (key == (char)ConsoleKey.Tab)  {  escapedQuotes.Append("\\t");  escapedQuotesAsQuotedString.Append("\\t");  }  else if (key == (char)ConsoleKey.Enter)  {  escapedQuotes.Append(@"\n");  escapedQuotesAsQuotedString.Append(@"\n");  }  else if (key == '\n')  {  continue;  }  else  {  escapedQuotes.Append(key);  escapedQuotesAsQuotedString.Append(key);  }  }  escapedQuotesAsQuotedString.Append("\"");  Console.WriteLine(escapedQuotes);  Console.WriteLine(escapedQuotesAsQuotedString);  }  }  } | |
| **Тестове**   1. Само нови редове. 2. Текст с кавички,табулации,нови редове. 3. Само табулации. 4. Само кавички. | |
| **Вход** | **Изход** |
| The "use" of quotations  causes difficulties. | The\t\”use” of quotations\ncauses difficulties.  @”The\t\”use” of quotations\ncauses difficulties.” |
| **Вход** | **Изход** |
| “Quotation”  . | \n\n\n\”Quotation\”\n.  @”\n\n\n”Quotation”\n.” |

|  |  |
| --- | --- |
| **Задача 10: Сърце** | |
| **Условие:**  Напишете програма, която принтира фигура във формата на сърце със знак, подаден от конзолата. | |
| **Описание на входа:**  На еднственият ред в конзолата се въвежда един знак(символ). | |
| **Описание на изхода:**  Изходът се извежда на следващите пет реда във формата на сърце от дадения символ. Ако  Пример:  О  ОО ОО  О О О  О О  О О  О | |
| **Анализ на задачата:**  Инициализираме една променлива от тип char и четем от конзолата входа.  С Console.WriteLine(); и интервали изписваме желаната форма на сърце на конзолата. | |
| **Решение (сорс код)** | |
| using System;  namespace Hearts  {  class Program  {  static void Main(string[] args)  {  char symbol = ' ';  symbol = char.Parse(Console.ReadLine());  Console.WriteLine(" {0}{0} {0}{0} ", symbol);  Console.WriteLine("{0} {0} {0}", symbol);  Console.WriteLine(" {0} {0} ", symbol);  Console.WriteLine(" {0} {0} ", symbol);  Console.WriteLine(" {0} ", symbol);  }  }  } | |
| **Тестове**  Интересни случаи за тестване:   * Примерният вход от условието на задачата * Цифра * Малка буква * Голяма буква * Стандартен специален символ примерно % * Специфичен символ примерно € | |
| **Вход** | **Изход** |
| О | ОО ОО  О О О  О О  О О  О |
| **Вход** | **Изход** |
| 1 | 11 11  1 1 1  1 1  1 1  1 |
| **Вход** | **Изход** |
| а | аа аа  а а а  а а  а а  а |
| **Вход** | **Изход** |
| А | АА АА  А А А  А А  А А  А |
| **Вход** | **Изход** |
| % | %% %%  % % %  % %  % %  % |
| **Вход** | **Изход** |
| β | ββ ββ  β β β  β β  β β  β |
| **Вход** | **Изход** |
| Ω | ΩΩ ΩΩ  Ω Ω Ω  Ω Ω  Ω Ω  Ω |
| **Вход** | **Изход** |
| ∑ | ∑∑ ∑∑  ∑ ∑ ∑  ∑ ∑  ∑ ∑  ∑ |
| **Вход** | **Изход** |
| ∞ | ∞∞ ∞∞  ∞ ∞ ∞  ∞ ∞  ∞ ∞  ∞ |
| **Вход** | **Изход** |
| Θ | ΘΘ ΘΘ  Θ Θ Θ  Θ Θ  Θ Θ  Θ |

|  |  |
| --- | --- |
| Задача 12. EmployeeRecords | |
| **Условие**  Фирма, занимаваща се с маркетинг, иска да пази запис с данни на нейните служители. Всеки запис трябва да има следната характеристика – първо име,фамилия,възраст,пол(‘m’ и ‘f’) и уникален номер на служителя(27560000 до 27569999). Характаристиките се четат от конзолата.Декларирайте необходимите променливи, за да се запази информацията за един служител,като използвате подходящите типове данни и описателни имена. Отпечатайте използвания тип данни с описателното му име и характеристиката на записа. | |
| **Описание на входа**  1 ред - първото име на служителя;  2 ред – фамилия;  3 ред – възраст;  4 ред – пол;  5 ред уникален номер на служителя; | |
| **Описание на изхода**  1 ред – тип данни описателно име = първото име на служителя  2 ред – тип данни описателно име = фамилия на служителя  3 ред – тип данни описателно име = възраст на служителя  4 ред – тип данни описателно име = пол на служителя  5 ред – тип данни описателно име = уникален номер на служителя | |
| **Анализ на задачата**  Първо решаваме кой тип данни да използваме за всяка от променливите, измисляме подходящо описателно име, отговарящо на camelCase, добавяме равно и въведеното от конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace EmployeeRecord  {  class Program  {  static void Main(string[] args)  {  string firstName = Console.ReadLine();  string lastName = Console.ReadLine();  sbyte age = sbyte.Parse(Console.ReadLine());  char sex = char.Parse(Console.ReadLine());  int uniqueNumber = int.Parse(Console.ReadLine());  Console.WriteLine("string firstName = " + firstName);  Console.WriteLine("string lastName = " + lastName);  Console.WriteLine("sbyte age = " + age);  Console.WriteLine("char sex = " + sex);  Console.WriteLine("int uniqueNumber = " + uniqueNumber);  }  }  } | |
| **Тестове**   1. **Описателните имена дали са коректно зададени.** 2. **Проверка дали използваните типове данни са най-подходящите.** | |
| **Вход** | **Изход** |
| Joana  Ivanova  19  f  27567654 | string firstName = Joana  string lastName = Ivanova  sbyte age = 19  char sex = f  int uniqueNumber = 27567654 |

|  |  |
| --- | --- |
| **Задача 13: Разменени стойности** | |
| **Условие**  Напишете програма, която чете две цели числа N и M, след което да разменя стойностите им и ги принтира на конзолата. За целта наименувайте първата променлива numberN, а втората – numberM. | |
| **Описание на входа**  Входните данни се въвеждат на два реда. На първият ред се въвежда N. На вторият ред се въвежда M. N и M са цели числа между -2147483648 до 2147483647. Данните винаги ще са в коректен формат. | |
| **Описание на изхода**  Изходните данни трябва да са на два реда. На първият ред трябва да е стойността на първата променлива. На вторият ред трябва да е стойността на втората променлива.  **Пример:**  Вход:  5  10  Изход:  10  5 | |
| **Анализ на задачата**  Декларираме две променливи от тип int. Четем техните стойности от конзолата и им присвояваме стойности с метода Parse.  1.Присвояваме на първата променлива стойност сумата на двете променливи.  2.Присвояваме на втората променлива стойност сумата минус втората променлива.  3.Присвояваме на първата променлива стойност сумата минус втората променлива (която вече е различна)  Така разменяме стойностите без да използваме трета променлива. | |
| **Решение (сорс код)** | |
| using System;  class ExchangeValues  {  static void Main(string[] args)  {  int a = 0;  int b = 0;  a = int.Parse(Console.ReadLine());  b = int.Parse(Console.ReadLine());  a = a + b;  b = a - b;  a = a - b;  Console.WriteLine(a);  Console.WriteLine(b);  }  } | |
| **Тестове**  Интересни случаи за тестване   * Даденият пример от условието * Две еднакви стойности * Максималната стойност и нула * Минималната стойност и нула * Максималната стойност и минималната стойност * При тестовете би следвало(чрез Reflection примерно) да се провери дали първата прочетена стойност е запазена в променлива numberN и дали втората е прочетена в numberM. След това да се провери дали и се извеждат в същият ред. * Може да се провери и дали използва трета променлива като евентуален последен тест. | |
| **Вход** | **Изход** |
| 5  10 | 10  5 |
| **Вход** | **Изход** |
| 2000  2000 | 2000  2000 |
| **Вход** | **Изход** |
| -2000  -2000 | -2000  -2000 |
| **Вход** | **Изход** |
| 2000  -2000 | -2000  2000 |
| **Вход** | **Изход** |
| 2147483647  0 | 0  2147483647 |
| **Вход** | **Изход** |
| -2147483648  0 | 0  -2147483648 |
| **Вход** | **Изход** |
| -2147483648  2147483647 | 2147483647  -2147483648 |
| **Вход** | **Изход** |
| 231293812  -32452432 | -32452432  231293812 |
| **Вход** | **Изход** |
| -2352423  5463452 | -2352423  5463452 |
| **Вход** | **Изход** |
| 56345435  76867867 | 76867867  56345435 |