# Решения на задачите от глава 04 на книгата "Въведение в програмирането със C#"

Предлагаме ви решения на задачите от [книгата "Въведение в програ­мирането със C#"](http://www.introprogramming.info), заедно с анализ на задачата, описание на използва­ните идеи, алгоритми, подходи за решаване и тестове.

## Авторски колектив

**Име: Венцислав Василев Щерев**

Контакти: v.v.shterev@gmail.com

**Име: Весела Коцева**

Контакти: vs\_kotseva@yahoo.com

**Име: Павел Димитров Колев**

Контакти: paveldk@almaalter.com

**Име: Стилиян Марков**

Контакти: stelko88@abv.bg

# Шаблони за описание на задачите

|  |  |
| --- | --- |
| Задача 01. Сумиране на три числа | |
| **Условие**  Напишете програма, която чете от конзолата три числа от тип int и отпечатва тяхната сума. | |
| **Описание на входа**  Входните данни се въвеждат като всяко едно от трите числа се въвежда на отделен ред. | |
| **Описание на изхода**  Изходът се извежда като на един ред от конзолата се изписва сумата на тези три числа. | |
| **Анализ на задачата**  Записваме входа от конзолата в string променлива.  Парсваме входните данни в три променливи от тип int, съответно firstNum, secondNum, thirdNum.  След това ги сумираме и запазваме резултата в променливата sum, която е от тип long, за да се предпазим от прехвърляне на обхвата на типа int. За да няма проблм при сумирането се налага кастване към тип long. | |
| **Решение (сорс код)** | |
| using System;  namespace \_01\_ReadThreeInt  {  class ReadThreeInt  {  static void Main()  {  string inputStr = Console.ReadLine();  int firstNum = int.Parse(inputStr);  inputStr = Console.ReadLine();  int secondNum = int.Parse(inputStr);  inputStr = Console.ReadLine();  int thirdNum = int.Parse(inputStr);  long sum = (long)firstNum + secondNum + thirdNum;  Console.WriteLine(sum);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • Когато и трите числа са с максималната стоиност на типа int – тогава се очаква резултата да надхвърли типа int.  • Когато и трите числа са с минималната стоиност на типа int – тогава се очаква резултата да прехвърли типа int.  • Тест за памет – мин. необходимата памет трябва да е достатъчна за string, int и long променливите. | |
| **Вход** | **Изход** |
| 0  0  0 | 0 |
| **Вход** | **Изход** |
| 1  2  3 | 6 |
| **Вход** | **Изход** |
| -1  -2  -3 | -6 |
| **Вход** | **Изход** |
| 5  -5  10 | 10 |
| **Вход** | **Изход** |
| 128  128  128 | 384 |
| **Вход** | **Изход** |
| -129  -129  -129 | -387 |
| **Вход** | **Изход** |
| 32768  32768  32768 | 98304 |
| **Вход** | **Изход** |
| -32769  -32769  -32769 | -98307 |
| **Вход** | **Изход** |
| 2147483647  2147483647  2147483647 | 6442450941 |
| **Вход** | **Изход** |
| -2147483648  -2147483648  -2147483648 | -6442450944 |

|  |  |
| --- | --- |
| **Задача 02. Намиране на обиколка и лице на кръг** | |
| **Условие**  Напишете програма, която чете от конзолата радиуса "r" на кръг и отпечатва неговия периметър и обиколка. "r" да бъде положително цяло число от 0 до 255. | |
| **Описание на входа**  Входните данни се въвеждат като на един единствен ред се въвежда стойноста на "r". | |
| **Описание на изхода**  Изходът се извежда като на първия ред се иписва периметъра на окръжността, на втория - лицето на кръга. И лицето и периметъра са с точност до третия знак след десетичната запетая. | |
| **Анализ на задачата**  Записваме входа от конзолата в string променлива.  Преобразуваме входните данни към число от тип byte с наименование radius.  За пресмятане на периметъра използваме формулата: .  За пресмятане на лицето използваме формулата: .  Kато за числото π ползваме константата Math.PI.  За да форматираме изходните данни ползваме StringFormat компонентата {0:F3}, която закръгля стойноста до третия знак. | |
| **Решение (сорс код)** | |
| using System;  namespace FindCircleAreaAndPer  {  class FindPerimeterAndArea  {  static void Main()  {  string inputStr = Console.ReadLine();  byte radius = byte.Parse(inputStr);  double perimeter = 2 \* radius \* Math.PI;  double area = radius \* radius \* Math.PI;  Console.WriteLine("{0:F3}", perimeter);  Console.WriteLine("{0:F3}", area);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • Когато въведената стойност на "r" е отрицателно число – тогава се очаква да възникне грешка.  • Когато въведената стойност на "r" е твърде голяма (по - голяма от 255) число – тогава се очаква да възникне грешка.  • Когато въведената стойност на "r" не е цяло число – тогава се очаква да възникне грешка. | |
| **Вход** | **Изход** |
| 0 | 0.00  0.00 |
| **Вход** | **Изход** |
| 1 | 6.283  3.141 |
| **Вход** | **Изход** |
| 5 | 31.416  78.540 |
| **Вход** | **Изход** |
| 10 | 62.832  314.159 |
| **Вход** | **Изход** |
| 15 | 94.248  706.858 |
| **Вход** | **Изход** |
| 100 | 628.319  31415.927 |
| **Вход** | **Изход** |
| 255 | 1602.212  204282.062 |
| **Вход** | **Изход** |
| -1 | System.OverflowExeption |
| **Вход** | **Изход** |
| 256 | System.OverflowExeption |
| **Вход** | **Изход** |
| 5.5 | System.FormatExeption |

|  |  |
| --- | --- |
| **Задача 03. Информация за фирма** | |
| **Условие**  Дадена фирма има име, адрес, телефонен номер, факс номер, уеб сайт и мениджър. Мениджърът има име, фамилия и телефонен номер. Напишете програма, която чете информацията за фирмата и нейния мениджър и я отпечатва след това на конзолата. | |
| **Описание на входа**  На първите пет реда се въвежда информацията за фирмата, като на първия ред въвеждаме името на фирмата, на втория – адреса, на третия – телефона, на четвъртия факса и на последния от първите 5 – интернет адреса.  На следващите четири реда въвеждаме първото и фамилното име на менижира, неговите години и телефония му номер. | |
| **Описание на изхода**  Изходът се състои от два реда. На първия извеждаме информацията за фирмата форматирана по следния начин:  [Company: Name:”CName”, Address:”CAddress”, Phone:”CName”, Fax:”CName”, WebSite:”CName”]  Втория ред съдържа информацията на мениджъра форматирана по следния начин:  [Manager: Name:”MName”, Age:”MAge”, Phone:”MPhone”] | |
| **Анализ на задачата**  Входа от първите пет реда от конзолата го записваме в пет променливи от тип string, съответно: companyName; companyAddress; companyPhone; companyFax; companySite.  След това в нови три променливи от тип string и една от тип byte, запазваме съответно първото име, фамилното име и телефона, а в byte променливата запазваме, след преобразуване към byte, и неговите години.  За да форматираме изходните данни по подходящ начин и печатаме. | |
| **Решение (сорс код)** | |
| using System;  namespace \_03\_CompanyAndManager  {  class Company  {  static void Main(string[] args)  {  string companyName = Console.ReadLine();  string companyAddress = Console.ReadLine();  string companyPhone = Console.ReadLine();  string companyFax = Console.ReadLine();  string companySite = Console.ReadLine();  string managerFName = Console.ReadLine();  string managerLName = Console.ReadLine();  byte managerAge = byte.Parse(Console.ReadLine());  string managerPhone = Console.ReadLine();  string managerFullName = managerFName + " " + managerLName;  Console.WriteLine("[Company: Name:{0}, Address:{1}, Phone:{2}, Fax:{3}, WebSite:{4}]", companyName, companyAddress, companyPhone, companyFax, companySite);  Console.WriteLine("[Manager: Name:{0}, Age:{1}, Phone:{2}]", managerFullName, managerAge, managerPhone);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • При въвеждане на телефони номера със специялни знаци, като ”+”, ”/” и т.н – грешка не бива да възниква.  • Когато въведената стойност за годините на мениджъра е отрицателно число– тогава се очаква да възникне грешка.  • Когато въведената стойност за годините на мениджъра не е цяло число – тогава се очаква да възникне грешка.  • Когато въведената стойност за годините на мениджъра е по-голяма от 255 – тогава се очаква да възникне грешка. | |
| **Вход** | **Изход** |
| Telerik Academy  Sofia 1729, Mladost 1A, Alexander Malinov st. 31  08887777777  0980566577  http://academy.telerik.com  Svetlin  Nakov  30  0888656778 | [Company: Name:Telerik Academy, Address:Sofia 1729, Mladost 1A, Alexander Malinov st. 31, Phone:08887777777, Fax:0980566577, WebSite: http://academy.telerik.com]  [Manager: Name:Svetlin Nakov, Age:30, Phone: 0888656778] |
| **Вход** | **Изход** |
| Super Academy  Sofia 1729  08887777777  (098) 0566577  academy.com  Svetlin  Svetlinov  30  +888656778 | [Company: Name:Super Academy, Address:Sofia 1729, Phone:08887777777, Fax:(098) 0566577, WebSite: academy.com]  [Manager: Name:Svetlin Svetlinov, Age:30, Phone: +888656778] |
| **Вход** | **Изход** |
| SuperStar  Sofia 456  (02) 7777777  098/0566577  star.com  Svetlin  Svetlinov  55  +359888656778 | [Company: Name:SuperStar, Address:Sofia 456, Phone(02) 7777777, Fax:098/0566577, WebSite: star.com]  [Manager: Name:Svetlin Svetlinov, Age:55, Phone: +359888656778] |
| **Вход** | **Изход** |
| SuperStar  Sofia 456  (02) 7777777  098/0566577  star.com  Svetlin  Svetlinov  -55  +359888656778 | System.OverflowExeption |
| **Вход** | **Изход** |
| SuperStar  Sofia 456  (02) 7777777  098/0566577  star.com  Svetlin  Svetlinov  555  +359888656778 | System.OverflowExeption |
| **Вход** | **Изход** |
| SuperStar  Sofia 456  (02) 7777777  098/0566577  star.com  Svetlin  Svetlinov  55.5  +359888656778 | System.FormatExeption |

|  |  |
| --- | --- |
| **Задача 04. Виртуални Колони** | |
| **Условие**  Напишете програма, която отпечатва три числа в три виртуални колони на конзолата. Всяка колона трябва да е с широчина 10 символа, а числата трябва да са ляво подравнени. Първото число трябва да е цяло число в шестнадесетична бройна система, второто да е дробно положи-телно, а третото – да е дробно отрицателно. Последните две числа да се закръглят до втория знак след десетичната запетая. | |
| **Описание на входа**  Входът се въвежда в три реда, като от първия ред се взима целочислената стойност, а от останалите два реда се взимат съответно положителното и отрицателното дробни числа. | |
| **Описание на изхода**  На единствения ред от изхода се извеждат и трите числа в три виртуални колони от по 10 символа, като първо е целочислената стойност преобразувана в шестнадесетична бройна система, след това са положителното и отрицателното дробно число. | |
| **Анализ на задачата**  Записваме входа от конзолата в string променлива.  Парсваме входните данни в три променливи, от тип int и две от тип double, съответно numForFirstCol, numForSecondCol, numForThirdCol.  След това ги печатаме на конзолата, като за да подравним в ляво изходните данни ползваме Alignment компонента {0,-10}. За да бъде изпечатано първото число в шестнадесетична бройна система е необходимо след Alignment компонента да се въведе и StringFormat компонентата {0,-10:X}. За да бъдат закръглени дробните числа до 2 знак е необходимо след Alignment компонента да се въведе и StringFormat компонентата {0,-10:F2}. | |
| **Решение (сорс код)** | |
| using System;  namespace Virtual\_Colums  {  class VirtualColums  {  static void Main()  {  string inputLine = Console.ReadLine();  int numForFirstCol = int.Parse(inputLine);  inputLine = Console.ReadLine();  double numForSecondCol = double.Parse(inputLine);  inputLine = Console.ReadLine();  double numForThirdCol = double.Parse(inputLine);  Console.WriteLine("{0,-10:X}{1,-10:F2}{2,-10:F2}",  numForFirstCol,numForSecondCol,numForThirdCol);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • При въвеждане на дробно(дробно защото, целочисленото е от тип int, а пък неговите минимална и максимална стойност преобразувани в hex няма опасност да се надхвърлят 10 символа) число по-дълго от 10 символа се очаква съответната колона да стане по-дълга от десет символа и да се получи слепване със съседното число ако има такова. | |
| **Вход** | **Изход** |
| 0  0  -0 | 0 0.00 0.00 |
| **Вход** | **Изход** |
| 123  12345  -12345 | 123 12345.00 -12345.00 |
| **Вход** | **Изход** |
| 2147483647  0  -0 | 7FFFFFFF 0.00 0.00 |
| **Вход** | **Изход** |
| 0  1234567890.12  -0 | 0 1234567890.120.00 |
| **Вход** | **Изход** |
| 0  0  -1234567890.12 | 0 0.00 -1234567890.12 |

|  |  |
| --- | --- |
| **Задача 05. Всички числа делими на 5 в даден интервал** | |
| **Условие**  Напишете програма, която чете от конзолата две цели числа (int) и отпечатва, колко числа има между тях, такива, че остатъкът им от деленето на 5 да е 0. Пример: в интервала (17, 25) има 2 такива числа. Самите граници също трябва да бъдат взети в предвид. | |
| **Описание на входа**  Входът се състои от два реда. На първия ред се въвежда долната граница, а на втория горната граница. | |
| **Описание на изхода**  Изхода се състои от един ред, на който се изписва броят на всички числа в дадения интервал,които са делими на 5. Към това число спадат и самите граници на интервла. | |
| **Анализ на задачата**  Записваме всеки ред от входа в string променлива.  Парсваме входните данни в две променливи от тип int, съответно firstNum и secondNum**.**  За да разберем колко числа делими на пет се съдържат в интервал от нула до дадено число е нужно само да разделим това число на 5. По пътя на тази логика за да намерим всички делими на 5 в дадена граница е необходима да разделим горната граница на 5 и от полученото да извадим частното на долната граница и 5.  Ето и формулата . Но тъй като и голната граница може да бъде число, което отговаря на условието , тогава проверяваме дали то се дели без остатък и ако да добавяме едно към резултата. | |
| **Решение (сорс код)** | |
| using System;  namespace DivisibleBy5  {  class AllDivisibelBy5  {  static void Main()  {  string inputStr = Console.ReadLine();  int firstNum = int.Parse(inputStr);  inputStr = Console.ReadLine();  int secondNum = int.Parse(inputStr);  int numberOfP = (secondNum / 5) - (firstNum / 5);  if (firstNum % 5 == 0)  {  numberOfP++;  }  Console.WriteLine(numberOfP);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • Тъй като за граници се ползват числа от тип int, е възможно за долна граница може да бъде въведено число равно или по-малко на нула. Тъй като и нулата е число което се дели на 5 без остатък, тогава се очаква и нулата да влиза в сметките.  • Пресмятането на числата в интервала по гореспоменатата формула става сравнително по-бързо отколкото, пресмятането с цикъл. Ето защо тест с долна граница мин стойност на int и горна – макс стойност на int е удачен (тест за скорост). | |
| **Вход** | **Изход** |
| 17  25 | 2 |
| **Вход** | **Изход** |
| 15  25 | 3 |
| **Вход** | **Изход** |
| 0  25 | 6 |
| **Вход** | **Изход** |
| -25  25 | 11 |
| **Вход** | **Изход** |
| -1000  1000 | 401 |
| **Вход** | **Изход** |
| -10000  10000 | 4001 |
| **Вход** | **Изход** |
| -100000  100000 | 40001 |
| **Вход** | **Изход** |
| -1000000  1000000 | 400001 |
| **Вход** | **Изход** |
| -10000000  10000000 | 4000001 |
| **Вход** | **Изход** |
| -2147483648  2147483647 | 858993458 |

|  |  |
| --- | --- |
| **Задача 06. Сравнение без условни конструкции** | |
| Условие  Напишете програма, която чете две числа от конзолата и отпечатва по-голямото от тях. Решете задачата без да използвате условни конструкции. | |
| **Описание на входа**  Входът се състои от два реда. На всеки ред въвеждаме по едно число. | |
| **Описание на изхода**  Изхода се състои от един ред, на който се изписва по-голямото от двете числа. | |
| **Анализ на задачата**  Записваме всеки ред от входа в string променлива.  За да запазим двете входни числа ползваме променливи от тип decimal, съответно firstNum и secondNum. Ползваме decimal, защото в условието не е определено какви числа трябва да бъдат сравнявани.  За самото сравнение на входните две числа ползваме статичния метод на класа Math.Max, който при подаване на две числа, връща като резултат по-голямото от тях.  Изпечатваме резултата на конзолата. | |
| **Решение (сорс код)** | |
| using System;  namespace WithoutIF  {  class GreaterWithoutIF  {  static void Main(string[] args)  {  string inputStr = Console.ReadLine();  decimal firstNum = decimal.Parse(inputStr);  inputStr = Console.ReadLine();  int secondNum = int.Parse(inputStr);  decimal greater = Math.Max(firstNum, secondNum);  Console.WriteLine(greater);  }  }  } | |
| **Тестове**  Тъй като за тази задача ползваме единствено Math.Max, то интересни случаи за тестване няма. | |
| **Вход** | **Изход** |
| 1  10 | 10 |
| **Вход** | **Изход** |
| 10  1 | 10 |
| **Вход** | **Изход** |
| 1  1 | 1 |
| **Вход** | **Изход** |
| 10.5  10.6 | 10.6 |
| Вход | Изход |
| 10.6  10.5 | 10.6 |
| **Вход** | **Изход** |
| 123456789.123456789  123456789.123456788 | 123456789.123456789 |

|  |  |
| --- | --- |
| **Задача 07. Сумиране на 5 числа** | |
| Напишете програма, която чете пет числа (от тип int) и отпечатва тяхната сума. При невалидно въведено число да се подкани потребителя да въведе друго число. | |
| **Описание на входа**  Входът се състои от достатъчен на брой редове за да можевсяко едно от 5те числа да бъде въведено успешно. Например: Ако сгрешим на два пъти въвеждането на първото число, то тогава първите три реда от входа съответстват на въвеждането на първото число, същата процедура следва за всяко едно от петте числа. | |
| **Описание на изхода**  Изхода се състои от един ред, на който е сумата от въведените пет числа. | |
| **Анализ на задачата**  Първоначално инициализираме променлива от тип string с празен стринг(string.Empty), заделяме памет и за първото число от тип int(firstNum), създавеме си и една булева променлива(isNumberCorrect).  След това влизаме в while цикъл, който се изпълнява докато булевата променлива не стане истина.  За да стане булевата променлива истина е нужно статичния метод int.TryParse(), да върне като резултат истина, което пък от своя страна става когато се получи успешно парсване на стринга inputLine. При успешно парсване на inputLine резултатът от него се записва в out променливата, която в случая е firstNum и се излиза от while цикълът.  Същата процедура се изпълнява и за въвеждане на останалите числа.  След успешното въвеждане на всички числа те се сумират и сумата се запазва в long променливата sum. Тъй като и петте числа са от тип int за се преобразуват към long налага кастване((long)) на поне едно от тези числа.  Накрая печатаме сумата на конзолата. | |
| **Решение (сорс код)** | |
| using System;  namespace Sum5Nums  {  class SumFiveNums  {  static void Main()  {  string inputLine = string.Empty;  int firstNum = 0;  bool isNumberCorrect = false;  while (!isNumberCorrect)  {  inputLine = Console.ReadLine();  isNumberCorrect = int.TryParse(inputLine,out firstNum);  }  int secondNum = 0;  isNumberCorrect = false;  while (!isNumberCorrect)  {  inputLine = Console.ReadLine();  isNumberCorrect = int.TryParse(inputLine, out secondNum);  }  int thirdNum = 0;  isNumberCorrect = false;  while (!isNumberCorrect)  {  inputLine = Console.ReadLine();  isNumberCorrect = int.TryParse(inputLine, out thirdNum);  }  int fourthNum = 0;  isNumberCorrect = false;  while (!isNumberCorrect)  {  inputLine = Console.ReadLine();  isNumberCorrect = int.TryParse(inputLine, out fourthNum);  }  int fifthNum = 0;  isNumberCorrect = false;  while (!isNumberCorrect)  {  inputLine = Console.ReadLine();  isNumberCorrect = int.TryParse(inputLine, out fifthNum);  }  long sum = (long)firstNum + secondNum + thirdNum + fourthNum + fifthNum;  Console.WriteLine(sum);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • При подаване на вход, който съдържа буква или друг спец. символ се очаква потребителят да въведе друго число  • При подаване на вход, който е дробно число се очаква потребителят да въведе друго число.  • Когато някои от петте числа са с максималната стоиност на типа int – тогава се очаква резултата да надхвърли типа int.  • Когато някои от петте числа са с минималната стоиност на типа int – тогава се очаква резултата да прехвърли типа int. | |
| **Вход** | **Изход** | |
| 1  2  3  4  5 | 15 | |
| **Вход** | **Изход** | |
| 1  2  3  4  5 | 15 | |
| **Вход** | **Изход** | |
| -1  -2  -3  -4  -5 | -15 | |
| **Вход** | **Изход** | |
| -1  -2  -3  -4  - 5  -5 | 15 | |
| **Вход** | **Изход** | |
| 5  10.5  10  15  20  25 | 75 | |
| **Вход** | **Изход** | |
| A  B  C  D  Faamnkd, am  5  10  15  20  25 | 75 | |
| **Вход** | **Изход** | |
| 2147483647  2147483647  2147483647  2147483647  2147483647 | 10737418235 | |
| **Вход** | **Изход** | |
| -2147483648  -2147483648  -2147483648  -2147483648  -2147483648 | -10737418240 | |

|  |  |
| --- | --- |
| **Задача 08. Най-голямо от 5 числа** | |
| **Условие**  Напишете програма, която чете пет числа от конзолата и отпечатва най-голямото от тях. | |
| **Описание на входа**  Входът се състои от пет реда. На всеки ред въвеждаме по едно число. | |
| **Описание на изхода**  Изхода се състои от един ред, на който се изписва най-голямото от петте числа. | |
| **Анализ на задачата**  Записваме всеки ред от входа в string променлива.  За да запазим петте входни числа ползваме променливи от тип decimal, съответно firstNum, secondNum, thirdNum, fourtNum и fifthnum. Ползваме decimal, защото в условието не е определено какви числа трябва да бъдат сравнявани.  В нова променлива от тип decimal greater, записваме ресултата от метода Math.Max(firstNum, secondNum), който връща по голямото от първите две числа. След това правим същото за третото число с тази разлика, че като параметри на метода Мath.Max(firstNum, greater), подаваме третото число и greater(което всъщност е досегашното най-голямо). Повтаряме тази стъпка за всяко едно въведено число.  Изпечатваме резултата на конзолата. | |
| **Решение (сорс код)** | |
| using System;  namespace GreaterFrom5Nums  {  class GreaterFromFive  {  static void Main()  {  string inputStr = Console.ReadLine();  decimal firstNum = decimal.Parse(inputStr);  inputStr = Console.ReadLine();  decimal secondNum = decimal.Parse(inputStr);  inputStr = Console.ReadLine();  decimal greater = Math.Max(firstNum, secondNum);  decimal thirdNum = decimal.Parse(inputStr);  greater = Math.Max(thirdNum, greater);  inputStr = Console.ReadLine();  decimal fourthNum = decimal.Parse(inputStr);  greater = Math.Max(fourthNum, greater);  inputStr = Console.ReadLine();  decimal fifthNum = decimal.Parse(inputStr);  greater = Math.Max(fifthNum, greater);  Console.WriteLine(greater);  }  }  } | |
| **Тестове**  Тъй като за тази задача ползваме единствено Math.Max, то интересни случаи за тестване няма. | |
| **Вход** | **Изход** |
| 1  2  3  4  5 | 5 |
| **Вход** | **Изход** |
| 5  1  2  3  4 | 5 |
| **Вход** | **Изход** |
| 1  5  2  3  4 | 5 |
| **Вход** | **Изход** |
| 1  2  5  3  4 | 5 |
| **Вход** | **Изход** |
| 1  2  3  5  4 | 5 |
| **Вход** | **Изход** |
| 5  5  5  5  5 | 5 |
| **Вход** | **Изход** |
| 5.5  4  5  -1  -100.4 | 5.5 |
| **Вход** | **Изход** |
| 0  -1  -2  -3  -4 | 0 |

|  |  |
| --- | --- |
| **Задача 09. Квадратно уравнение** | |
| **Условие**  Напишете програма, която чете коефициентите a, b и c от конзолата и решава уравнението: ax2+bx+c=0. Програмата трябва да принтира реалните решения на уравнението на конзолата. | |
| **Описание на входа**  Входът се състои от три реда. На всеки ред въвеждаме по от коефициентите на квадратното уравнение. Първо коефициент а, после – b и накрая – c. | |
| **Описание на изхода**  Ако уравнението няма решение печатаме на конзолата „No real roots.”. Ако уревнението има един корен – печатаме „One real root.” И на следващия ред печатаме самия корен. Ако уравнениет има два реални корена печатаме „Two real roots.” И на следващите два реда ги печаме и двата корена, като първо печатаме този които е , след това - . | |
| **Анализ на задачата**  Записваме всеки ред от входа в string променлива.  За да запазим трите коефициента ползваме променливи от тип double, съответно coeffA, coeffB и coeffC.  В нова променлива от тип double discriminantD, пресмятаме дискриминантата по формулата. След това ако дискриминантата е по-голяма от 0 пресмятаме двата корена по формулата , печатаме „Two real roots.” и след това и самите корените. За пресмятането на корена във формулата ползваме метода Math.Sqrt. Ако дискриминанта е равна на 0, тогава печатаме „One real root.” и след това пресметнатия по формулата корен. Ако дискриминантата е по-малка от нула печатаме „No real roots.” | |
| **Решение (сорс код)** | |
| using System;  namespace \_QuadraticEquation  {  class QuadraticEquation  {  static void Main()  {  string inputStr = Console.ReadLine();  double coeffA = double.Parse(inputStr);  inputStr = Console.ReadLine();  double coeffB = double.Parse(inputStr);  inputStr = Console.ReadLine();  double coeffC = double.Parse(inputStr);  double discriminantD = (coeffB \* coeffB) - (4 \* coeffA \* coeffC);  double root1;  double root2;  if (discriminantD > 0)  {  root1 = (-coeffB + Math.Sqrt(discriminantD)) / (2 \* coeffA);  root2 = (-coeffB - Math.Sqrt(discriminantD)) / (2 \* coeffA);  Console.WriteLine("Two real roots.");  Console.WriteLine(root1);  Console.WriteLine(root2);  }  else if (discriminantD == 0)  {  root1 = root2 = (-coeffB - Math.Sqrt(discriminantD)) / (2 \* coeffA);  Console.WriteLine("One real root.");  Console.WriteLine(root1);  }  else  {  Console.WriteLine("No real roots.");  }  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • Когато дискриминантата е нула – очаква се единствен корен.  • Когато дискриминантата е по-голяма от нула – очакват се два корена.  • Когато дискриминантата е по-малка от нула – не се очакват корени.  • Когато коефициентите b и/или c са 0 се очакват коректни резултати. | |
| **Вход** | **Изход** |
| 1  0  -4 | Two real roots.  2  -2 |
| **Вход** | **Изход** |
| 3  4  5 | No real roots. |
| **Вход** | **Изход** |
| 1  4  -5 | Two real roots.  1  -5 |
| **Вход** | **Изход** |
| 1  4  4 | One real root.  -2 |
| **Вход** | **Изход** |
| 1  -13  12 | Two real roots.  12  1 |
| **Вход** | **Изход** |
| 8  -30  7 | Two real roots.  3.5  0.25 |

|  |  |
| --- | --- |
| **Задача 10. Сумиране на N брой числа** | |
| **Условие**  Напишете програма, която прочита едно цяло число n от конзолата. След това прочита още n на брой числа от конзолата и отпечатва тяхната сума. | |
| **Описание на входа**  На първия ред от входа въвеждаме броя на числато които искаме да сумираме N. На следващите N реда въвеждаме самите числа. | |
| **Описание на изхода**  Изхода се състои от един ред, на който е сумата от въведените N числа. | |
| **Анализ на задачата**  Записваме всеки ред от входа в string променлива.  След парсване на първия ред от входа получаваме броя на числата които искаме да сумираме N.  Заделяме памет за една променлива от тип long sum в която ще натрупваме сумата и една променлива от тип int currentNum в която ще пазим всяко едно тукущо въведено чило.  Стартираме един for цикъл от нула до N, който представлъва броя на числата които искаме да добавим. За всяка итерация на този цикъл четем ред от козолата парсваме го и го приравняваме на currentNum. След това текущата сума я събираме с currentNum и резултата го присвояваме на sum.  Накрая се изпечатва резултата на конзолата. | |
| **Решение (сорс код)** | |
| using System;  namespace \_07.\_1SumOfNNumbers  {  class SumOfNNUmbers  {  static void Main()  {  string inputLine = Console.ReadLine();  int numberN = int.Parse(inputLine);  int currentNum = 0;  long sum = 0;  for (int i = 0; i < numberN; i++)  {  inputLine = Console.ReadLine();  currentNum = int.Parse(inputLine);  sum = sum + currentNum;  }  Console.WriteLine(sum);  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • Когато някои от Nте числа са с максималната стоиност на типа int – тогава се очаква резултата да надхвърли типа int.  • Когато някои от Nте числа са с минималната стоиност на типа int – тогава се очаква резултата да прехвърли типа int. | |
| **Вход** | **Изход** |
| 3  32768  32768  32768 | 98304 |
| **Вход** | **Изход** |
| 3  -32769  -32769  -32769 | -98307 |
| **Вход** | **Изход** |
| 5  1  2  3  4  5 | 15 |
| **Вход** | **Изход** |
| 5  -1  -2  -3  -4  -5 | -15 |
| **Вход** | **Изход** |
| 5  2147483647  2147483647  2147483647  2147483647  2147483647 | 10737418235 |
| **Вход** | **Изход** |
| 5  -2147483648  -2147483648  -2147483648  -2147483648  -2147483648 | -10737418240 |
| **Вход** | **Изход** |
| 15  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | 120 |

|  |  |
| --- | --- |
| **Задача 11. Печатане на числата от 1 до N** | |
| **Условие**  Напишете програма, която прочита цяло число n от конзолата и отпечатва на конзолата всички числа в интервала [1…n], всяко на отделен ред. | |
| **Описание на входа**  На първия ред от входа въвеждаме числото N. | |
| **Описание на изхода**  Изхода се състои от N на брой реда, като на всеки ред има по едно число. Първото число винаги е 1 а последното винаги е числото N. | |
| **Анализ на задачата**  Записваме единствения ред от входа в string променлива.  След това стартираме един for цикъл, като първата итерация i е равна на 1 а последната N.  На всяка една итерация на цикъла печатаме на цял ред стойност на i. | |
| **Решение (сорс код)** | |
| using System;  namespace NumbersFromOneToN  {  class OneToN  {  static void Main()  {  string inputStr = Console.ReadLine();  int numberN = int.Parse(inputStr);  for (int i = 1; i <= numberN; i++)  {  Console.WriteLine(i);  }  }  }  } | |
| **Тестове**  Тъй като задачата е сравнително проста, затова не смятаме че има интересни случай за тестване. | |
| **Вход** | **Изход** |
| 10 | 1  2  3  4  5  6  7  8  9  10 |
| **Вход** | **Изход** |
| 100 | 1  2  3  ...  100 |
| **Вход** | **Изход** |
| 1000 | 1  2  3  ...  1000 |
| **Вход** | **Изход** |
| 10000 | 1  2  3  ...  10000 |
| **Вход** | **Изход** |
| 100000 | 1  2  3  ...  100000 |

|  |  |
| --- | --- |
| **Задача 12. Първите 100 на Фибоначи** | |
| **Условие**  Напишете програма, която отпечатва на конзолата първите 100 числа от редицата на Фибоначи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ... | |
| **Описание на входа**  За тази задача вход не се налага да се въвежда. | |
| **Описание на изхода**  Изхода се състои от 101 на брой реда, като на всеки ред има по едно число на Фобоначи, започвайки от нулевото до 100ното. Необходимо е да има 101 реда защото по дефиниция редицата на Фибоначи има и нулев елемент който е равен на 0, следователно за да изпечатаме всички до стотния елемент от редицата е необходимо да има 101 реда. | |
| **Анализ на задачата**  Започваме като запазваме в три променливи от тип \*BigInteger(firstFibonacci, secondFibonacci, nFibonacci), първите три числа от редицата на Фибоначи, съответно 0,1,1. От тези три числа започваме да пресмятаме следващите по ред в редицата.  За пресмятането на следващите числа от редицата е необходимо да стартираме един for цикъл който се състои от 101 - 3 итераци(-3 защото първите три вече ги знаем).  При всяка итерация на for цикъла firstFibonacci става равно на secondFibonacci, secondFibonacci става равно на nFibonacci, а nFibonacci става равно на firstFibonacci + secondFibonacci. След което печатаме nFibonacci.  \*Тъй като редицата не Фибоначи расте доста бързо е необходимо резултатите да се съхраняват в променлива която може да съдържа достатъчно голям набор от цифри. | |
| **Решение (сорс код)** | |
| using System;  using System.Numerics;  namespace First100ofFibonacci  {  class Fibonacci  {  static void Main()  {  BigInteger firstFibonacci = 0;  BigInteger secondFibonacci = 1;  BigInteger nFibonacci = firstFibonacci + secondFibonacci;  Console.WriteLine(firstFibonacci);  Console.WriteLine(secondFibonacci);  Console.WriteLine(nFibonacci);  for (int i = 3; i <= 100; i++)  {  firstFibonacci = secondFibonacci;  secondFibonacci = nFibonacci;  nFibonacci = firstFibonacci + secondFibonacci;  Console.WriteLine(nFibonacci);  }  }  }  } | |
| **Тестове**  Интересните случаи за тестване са следните:  • Интересно е да се провери дали първия елемент действително е 0 и дали има точно 101 елемента изпечатани на конзолата. | |
| **Вход** | **Изход** |
|  | 0  1  1  2  3  5  8  13  21  34  55  ...  354224848179261915075 |

|  |  |  |
| --- | --- | --- |
| **Задача 13. Специална сума** | | |
| **Условие**  Напишете програма, която пресмята сумата (с точност до 0.001): 1+ 1/2 + 1/3 + 1/4 + 1/5 + ... 1/N. | | |
| **Описание на входа**  На първия ред от входа въвеждаме числото N. | | |
| **Описание на изхода**  Изхода се състои от един ред, на който е получената от гореспоменатата редица сумата . | | |
| **Анализ на задачата**  Записваме единствения ред от входа в string променлива.  Парсваме входния ред в променлива от тип int(lastElement), която представлява номера на последния елемент.  Заделяме и променлива в която ще пазим сумата(sum). Присвояваме и стойност 1(съответно първия елемент от редицата).  Стартираме for цикъл, който започва от 2(защото първия елемент вече сме го натрупалив sum) до lastElement(вкучително).  При всяка итерация в сумата натрупваме следващия елемент от редицата,който е 1.0/i.  За да форматираме изходните данни ползваме StringFormat компонентата {0:F3}, която закръгля стойноста до третия знак след десетичната запетая. | | |
| **Решение (сорс код)** | | |
| using System;  namespace \_10\_CalculateSum  {  class CalcSum  {  static void Main()  {  string inputStr = Console.ReadLine();  int lastElement = int.Parse(inputStr);  double sum = 1.0;  for (int i = 2; i <= lastElement; i++)  {  sum = sum + (1.0 / i);  }  Console.WriteLine ("{0:F3}",sum);  }  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:  • Интересно е да се провери дали дали закръглянето е действително до третия знак. Дали правилно е форматиран изхода. | | |
| **Вход** | **Изход** | |
| 5 | | 2.283 |
| **Вход** | **Изход** | |
| 15 | | 3.318 |
| **Вход** | **Изход** | |
| 25 | | 3.816 |
| **Вход** | **Изход** | |
| 35 | | 4.147 |
| **Вход** | **Изход** | |
| 45 | | 4.395 |