# Решения на задачите от глава 02 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

**Boyan Antonov Желязков**

Контакти: 0883-48-17-08 ([b.zhelyazkov@yahoo.com](mailto:b.zhelyazkov@yahoo.com)) – Course Number 301\_BOZH

**E. Yusein**

Контакти: (по желание може да се остави уеб сайт, мейл, Facebook, LinkedIn, Twitter профил, …) Отказал се поради твърде голяма заетост на работа.

**T. Lilyanova**

Контакти: (по желание може да се остави уеб сайт, мейл, Facebook, LinkedIn, Twitter профил, …) Отказва се от академията имала си работа. Но беше с готовност да ми помогне. Аз не потърсих помощ

**I. Kolchagov**

Контакти: (kolchagov@gmail.com) Не взима участие в теам работа. Пише само по email, не ползвал skype с емайл искал да си пишем. Последно съм му писал емайл на 22.03 на който не е отговорил.

Но нямам против да свърша работата вместо него.

# Задача 1. Choose all possible pimitive type for Value / Избиране на всички възможни типове за дадени число

|  |  |
| --- | --- |
| Задача 1. Choose all possible pimitive type for Value / Избиране на всички възможни типове за дадени число | |
| **Условие**  На първия ред получаваме какво число ще получим за положително символа “+” за отрицателно символа “-”  Получваме едно цяло число N от конзолата и да кажем кои са всички целочислени типовете **sbyte**, **byte**, **short**, **ushort**, **int**, **uint**, **long** и **ulong** в които може да бъде съхранена променливата. Типовете в сортиран в азбучен ред вид изглеждат така „byte, int, long, sbyte, short, uint, ulong, ushort“  **Като условие на същата задача с малка предисторийка следното условие:**  Питър е много палав ученик и не може да преценя от какъв тип е дадена число, което учителката му казва затова той моли вас за помощ. Всеки път когато учителката пита Питър за дадено число от какъв типове може да е преди това му казва дали числото е положително или отрицателно. Задачата на Питър е да намери всички целочислени типовете **sbyte**, **byte**, **short**, **ushort**, **int**, **uint**, **long** и **ulong**, в които може да бъде записано числото и ги изведете на козолата в сортирани по азбучен ред разледени с “, “ (запетая и интервал). Типовете в сортиран в азбучен ред вид изглеждат така „byte, int, long, sbyte, short, uint, ulong, ushort“ | |
| **Описание на входа**  На първия ред получавате символа + или – които определя знака на числото.  На втория ред получвате числото **N**  **Ограничения:**  –9223372036854775808 <= **N** <= 18446744073709551615.  **Условия:**  Данните са винаги валидни.  Числото 0 считаме за положително число.  **Примерен вход:**  +  1553 | |
| **Описание на изхода**  На единствения изход извеждаме стринг с всички променливи в азбучен ред в които може да бъде записана променливата.  **Примерен изход:**  int, long, short, uint, ulong, ushort | |
| **Анализ на задачата**  За да може да вкючим долната граница на long и горната на ulong ние в началото правим проверка дали числото е положително и ако е така декларираме ulong и прочитаме число от конзолата което може да е максималното за типа ulong и ако сме избрали друг тип няма да можем да продължим.  Ако числото е отрицателно така декларираме long и прочитаме число от конзолата което може да е минималното за типа long и ако сме избрали друг тип няма да можем да продължим.  За да изкраме типовете в подреден вид без да изполваме масиви и сортирания, когато видим къде сме просто от условието премахваме по малките типове от данни и извеждаме останалите типове който са в сортиран вид за числото.  За да разберем всички типове на дадена стойност започваме с една проверка дали стойността е положителна за да знаем дали да вкючваме променливите типове без знак. След това продължаваме проверка от най-малкия интервал на оста ако той съовества добаваме всички типове наляво за отрицателните ( надясно за положителните ) числа. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace AllTypeOfANumber  {  class AllTypeOfANumber  {  static void Main(string[] args)  {  char sign = char.Parse( Console.ReadLine());  if (sign == '+')  {  ulong inputNumber = ulong.Parse(Console.ReadLine());  if (inputNumber <= 127)  {  Console.WriteLine("byte, int, long, sbyte, short, uint, ulong, ushort");  }  else if(inputNumber<=255)  {  Console.WriteLine("byte, int, long, short, uint, ulong, ushort");  }  else if(inputNumber<=32767)  {  Console.WriteLine("int, long, short, uint, ulong, ushort");  }  else if (inputNumber <= 65535)  {  Console.WriteLine("int, long, uint, ulong, ushort");  }  else if (inputNumber <= 2147483647)  {  Console.WriteLine("int, long, uint, ulong");  }  else if (inputNumber <= 4294967295)  {  Console.WriteLine("long, uint, ulong");  }  else if (inputNumber <= 9223372036854775807)  {  Console.WriteLine("long, ulong");  }  else if (inputNumber <= 18446744073709551615)  {  Console.WriteLine("ulong");  }    }  else  {  long inputNumber = long.Parse(Console.ReadLine());  if (inputNumber >= -128)  {  Console.WriteLine("int, long, sbyte, short");  }  else if (inputNumber >= -32768)  {  Console.WriteLine("int, long, short");  }  else if (inputNumber >= -2147483648)  {  Console.WriteLine("int, long");  }  else if (inputNumber >= -9223372036854775808)  {  Console.WriteLine("long");  }  }  }  }  } | |
| **Тестове**   1. Тестваме с номинален случай 2. Тестваме с 0 3. Тестваме с -Минимална стойност за byte 4. Тестваме с - Минимална стойност за short 5. Тестваме с - Минимална стойност за int 6. Тестваме с - Минимална стойност за long 7. Тестваме с - Mаксимална стойност за byte 8. Тестваме с - Mаксимална стойност за sbyte 9. Тестваме с - Mаксимална стойност за short 10. Тестваме с - Mаксимална стойност за ushort 11. Тестваме с - Mаксимална стойност за int 12. Тестваме с - Mаксимална стойност за uint 13. Тестваме с - Mаксимална стойност за long 14. Тестваме с - Mаксимална стойност за ulong 15. Тестваме с - Минимална стойност за sbyte -1 16. Тестваме с - Mаксимална стойност за sbyte+1 17. Тестваме с -Mинимална стойност за short-1 18. Тестваме с - Mаксимална стойност за short+1 19. Тестваме с -Mинимална стойност за int-1 20. Тестваме с - Mаксимална стойност за int+1 21. Тестваме с - Mаксимална стойност за long+1 22. Тестваме с - Mаксимална стойност за byte+1 23. Тестваме с - Mаксимална стойност за ushort+1 24. Тестваме с - Mаксимална стойност за int+1 25. Тестваме с - Mаксимална стойност за long+1 | |
| **Вход** | **Изход** |
| +  1553 | int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  0 | byte, int, long, sbyte, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| -  -128 | int, long, sbyte, short |
| **Вход** | **Изход** |
| -  -32768 | int, long, short |
| **Вход** | **Изход** |
| -  -2147483648 | int, long |
| **Вход** | **Изход** |
| -  -9223372036854775808 | long |
| **Вход** | **Изход** |
| +  127 | byte, int, long, sbyte, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  100 | byte, int, long, sbyte, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  255 | byte, int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  32767 | int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  65535 | int, long, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  2147483647 | int, long, uint, ulong |
| **Вход** | **Изход** |
| +  4294967295 | long, uint, ulong |
| **Вход** | **Изход** |
| +  9223372036854775807 | long, ulong |
| **Вход** | **Изход** |
| +  18446744073709551615 | ulong |
| **Вход** | **Изход** |
| -  -129 | int, long, short |
| **Вход** | **Изход** |
| +  128 | byte, int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| -  -32769 | int, long |
| **Вход** | **Изход** |
| +  32768 | int, long, uint, ulong, ushort |
| **Вход** | **Изход** |
| -  -2147483649 | long |
| **Вход** | **Изход** |
| +  2147483648 | long, uint, ulong |
| **Вход** | **Изход** |
| +  +9223372036854775808 | ulong |
| **Вход** | **Изход** |
| +  256 | int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  65536 | int, long, uint, ulong |
| **Вход** | **Изход** |
| +  4294967296 | long, ulong |
| **Вход** | **Изход** |
| +  123456789123456789 | long, ulong |

# Задача 2. Choose all possible pimitive type for Value with floating point/ Избиране на всички възможни типове за дадени число с десетична част

|  |  |  |
| --- | --- | --- |
| Задача 2. Choose all possible pimitive type for Value with floating point/ Избиране на всички възможни типове за дадени число с десетична част | | |
| **Условие**  Получваме едно дробно число N от конзолата и да кажем кои са всички floating point типовете **float, double,decimal** в които може да бъде съхранена променливата. Типовете в сортиран в азбучен ред вид изглеждат така „**decimal**, **decimal**, **float**“  **Като условие на същата задача с малка предисторийка следното условие:**  Джорд е много талантлив по иформатика и може да преценя много добре от какъв тип (float, double, decimal) е дадено число с плаваща запетая и при това успява да ги подреди типовете в азбучен ред. Изпитваща комисия от Гинес иска да прецени дали наистина Джордж е толкова талантлив в това което смята и дали заслужава да влезе в книгата на рекорди Гинес. Но комисията няма и наидея затова какво са типове данни в информатиката и кога дадено число може да бъде (float,double и decimal), (double и decimal) или само (decimal). Затова комисията Ви плаща за да и направите програма, която при дадено число **N** с плаваща запетая казва от какви типове може да бъде и при да ги изпечатва на конзолата сортиране по азбучен ред разделени с “; ” .  Възможните типове подредени по азбучен ред са decimal double float. | | |
| **Описание на входа**  На единствения ред се въвежда числото **N**  **Ограничения:**  -100.0000000000000000001 <= **N** <= 100.0000000000000000001  **Примерен вход:**  5.3 | | |
| **Описание на изхода**  На единствения ред от изхода трябва да бъдат изписани възможните типове за числото в азбучен ред разделени с “; ” (точка и запета последвана от точно 1 интервал)  **Примерен изход:**  decimal; double; float | | |
| **Анализ на задачата**   Четем числото от конзолата като string. Наример string inputStringNum  Взимаме и си прави три променливи  floatNum=float.Parse(inputStringNum);  doubleNum=double.Parse(inputStringNum);  decimalNum=decimal.Parse(inputStringNum);  Ако decimalNum - floatNum = 0 това не е достатчъно защото float режи завършващите нули трява да се провери дали inputStringNum [ без десетичната точка и ако (inputStringNum има ‚+‘ или ‚-‚ и той трява да бъде махнат)] дължината му е по малка от < 8 , то това число може да бъде присвоено към float double decimal без да загуби точност. И извеждаме Console.WriteLine(decimal; double; float);  Ако не. И decimalNum - floatNum = 0 това не е достатчъно защото double режи завършващите нули трява да се провери дали inputStringNum [ без десетичната точка и ако (inputStringNum има ‚+‘ или ‚-‚ и той трява да бъде махнат)] дължината му е по малка от < 15 , то това число може да бъде присвоено към double decimal без да загуби точност. И извеждаме Console.WriteLine(decimal; double);  Ако не то може да бъде само decimal  И извеждаме Console.WriteLine(decimal); | | |
| **Решение (сорс код)** | | |
| using System;  namespace AllPossibleTypesFloatingPointNumber  {  class AllPossibleTypesFloatingPointNumber  {  static void Main(string[] args)  {  decimal inputNum = decimal.Parse(Console.ReadLine());    decimal newNum = inputNum \* 10000000;  if (newNum % (long)newNum == 0)  {  Console.WriteLine("decimal; double; float");  }  else  {  newNum = inputNum \* 1000000000000000;  if (newNum % (long)newNum == 0)  {  Console.WriteLine("decimal; double");  }  else  {  Console.WriteLine("decimal");  }  }  }  }  } | | |
| **Тестове**  **От 1ви до 15ти тест тестват когато числото съдържа по-малко от 8 символа без десетичната точка и знака най отпред числото може да бъде** decimal; double; float  **От 16ти до 62ри тест тестват когато числото съдържа повече от 8 символа и по-мако от 16 символа (без десетичната точка и знака най отпред числото може да бъде)** double; float  **От 63ти до 95тен тест тестват когато числото съдържа повече от 8 символа без десетичната точка и знака най отпред числото може да бъде** decimal; double; float  **От 95ти до 100тен тест тестват когато числото има над 15 нули след десетината точка. Или има само 15 символа без десетична точка. Или само 0.000000000000000000** | | |
| **Вход** | **Изход** | |
| 10 | decimal; double; float | |
| **Вход** | **Изход** | |
| -18.0 | decimal; double; float | |
| **Вход** | **Изход** | |
| 19.9 | decimal; double; float | |
| **Вход** | **Изход** | |
| -9.000000 | decimal; double; float | |
| **Вход** | **Изход** | |
| 9.956986 | decimal; double; float | |
| **Вход** | **Изход** | |
| 99.00000 | decimal; double; float | |
| **Вход** | **Изход** | |
| 99.42389 | decimal; double; float | |
| **Вход** | **Изход** | |
| 131.1234 | decimal; double; float | |
| **Вход** | **Изход** | |
| -129.1999 | decimal; double; float | |
| **Вход** | **Изход** | |
| +1593.222 | decimal; double; float | |
| **Вход** | **Изход** | |
| -1999.999 | decimal; double; float | |
| **Вход** | **Изход** | |
| -89567.99 | decimal; double; float | |
| **Вход** | **Изход** | |
| 89754.14 | decimal; double; float | |
| **Вход** | **Изход** | |
| 123456.0 | decimal; double; float | |
| **Вход** | **Изход** | |
| 654321.9 | decimal; double; float | |
| **Вход** | **Изход** | |
| 9988998 | decimal; double; float | |
| **Вход** | **Изход** | |
| -7788998 | decimal; double; float | |
| **Вход double** | **Изход** | |
| 1.12345678123456 | decimal; double | |
| **Вход** | **Изход** | |
| -9.12345678123456 | decimal; double | |
| **Вход** | **Изход** | |
| 1.1234567 | decimal; double | |
| **Вход** | **Изход** | |
| -9.5369587 | decimal; double | |
| **Вход** | **Изход** | |
| 12.2345678123456 | decimal; double | |
| **Вход** | **Изход** | |
| -99.1234578123456 | decimal; double | |
| **Вход** | **Изход** | |
| 11.234567 | decimal; double | |
| **Вход** | **Изход** | |
| -95.369587 | decimal; double | |
| **Вход** | **Изход** | |
| 412.235678123456 | decimal; double | |
| **Вход** | **Изход** | |
| -599.134578123456 | decimal; double | |
| **Вход** | **Изход** | |
| 151.23567 | decimal; double | |
| **Вход** | **Изход** | |
| -975.36587 | decimal; double | |
| **Вход** | **Изход** | |
| +8412.23567812456 | decimal; double | |
| **Вход** | **Изход** | |
| -5969.13457812356 | decimal; double | |
| **Вход** | **Изход** | |
| 1521.2367 | decimal; double | |
| **Вход** | **Изход** | |
| -9275.3657 | decimal; double | |
| **Вход** | **Изход** | |
| +84512.2356812456 | decimal; double | |
| **Вход** | **Изход** | |
| -59669.1387812356 | decimal; double | |
| **Вход** | **Изход** | |
| 17521.237 | decimal; double | |
| **Вход** | **Изход** | |
| -79275.357 | decimal; double | |
| **Вход** | **Изход** | |
| +845142.236812456 | decimal; double | |
| **Вход** | **Изход** | |
| -596569.138781356 | decimal; double | |
| **Вход** | **Изход** | |
| 175251.27 | decimal; double | |
| **Вход** | **Изход** | |
| -794275.37 | decimal; double | |
| **Вход** | **Изход** | |
| +8453142.23612456 | decimal; double | |
| **Вход** | **Изход** | |
| -5956569.13878156 | decimal; double | |
| **Вход** | **Изход** | |
| 1755251.8 | decimal; double | |
| **Вход** | **Изход** | |
| -7942775.2 | decimal; double | |
| **Вход** | **Изход** | |
| +84453142.2361246 | decimal; double | |
| **Вход** | **Изход** | |
| -59256569.1387816 | decimal; double | |
| **Вход** | **Изход** | |
| 17552751 | decimal; double | |
| **Вход** | **Изход** | |
| -79452775 | decimal; double | |
| **Вход** | **Изход** | |
| +436175693.231246 | decimal; double | |
| **Вход** | **Изход** | |
| -298463066.137816 | decimal; double | |
| **Вход** | **Изход** | |
| 4361745693.23246 | decimal; double | |
| **Вход** | **Изход** | |
| -2984630676.13716 | decimal; double | |
| **Вход** | **Изход** | |
| 43617456493.2326 | decimal; double | |
| **Вход** | **Изход** | |
| -29846340676.1316 | decimal; double | |
| **Вход** | **Изход** | |
| 436613745693.232 | decimal; double | |
| **Вход** | **Изход** | |
| -298546306676.137 | decimal; double | |
| **Вход** | **Изход** | |
| 4364613745693.22 | decimal; double | |
| **Вход** | **Изход** | |
| -2985463064676.13 | decimal; double | |
| **Вход** | **Изход** | |
| 43646313745693.2 | decimal; double | |
| **Вход** | **Изход** | |
| -29854630649676.1 | decimal; double | |
| **Вход** | **Изход** | |
| 436463137456937 | decimal; double | |
| **Вход** | **Изход** | |
| -298546306496763 | decimal; double | |
| **Вход** | **Изход** | |
| -4827568395869386 | decimal | |
| **Вход** | **Изход** | |
| 4857385849278459 | decimal | |
| **Вход** | **Изход** | |
| 1.485738584927843 | decimal | |
| **Вход** | **Изход** | |
| -9.485738584927843 | decimal | |
| **Вход** | **Изход** | |
| 41.45738584927843 | decimal | |
| **Вход** | **Изход** | |
| -95.45738584927843 | decimal | |
| **Вход** | **Изход** | |
| 41.45738584927843 | decimal | |
| **Вход** | **Изход** | |
| -795.4573584927843 | decimal | |
| **Вход** | **Изход** | |
| 491.4578584927848 | decimal | |
| **Вход** | **Изход** | |
| -795.4573584927843 | decimal | |
| **Вход** | **Изход** | |
| 491.4578584927848 | decimal | |
| **Вход** | **Изход** | |
| -8695.457358927843 | decimal | |
| **Вход** | **Изход** | |
| 4931.457854927848 | decimal | |
| **Вход** | **Изход** | |
| -87645.45735927843 | decimal | |
| **Вход** | **Изход** | |
| 14786.45785927848 | decimal | |
| **Вход** | **Изход** | |
| -876475.4573592784 | decimal | |
| **Вход** | **Изход** | |
| 514786.4578592787 | decimal | |
| **Вход** | **Изход** | |
| -8766475.457352784 | decimal | |
| **Вход** | **Изход** | |
| 5147866.457859277 | decimal | |
| **Вход** | **Изход** | |
| -87666475.45752784 | decimal | |
| **Вход** | **Изход** | |
| 51447866.45785277 | decimal | |
| **Вход** | **Изход** | |
| -876676475.4575284 | decimal | |
| **Вход** | **Изход** | |
| 514478966.4578527 | decimal | |
| **Вход** | **Изход** | |
| -8766766475.455284 | decimal | |
| **Вход** | **Изход** | |
| 5194478966.457857 | decimal | |
| **Вход** | **Изход** | |
| -87663766475.45284 | decimal | |
| **Вход** | **Изход** | |
| 51946478966.45857 | decimal | |
| **Вход** | **Изход** | |
| -876643766475.4284 | decimal | |
| **Вход** | **Изход** | |
| 519464478966.4587 | decimal | |
| **Вход** | **Изход** | |
| -8766443766475.428 | decimal | |
| **Вход** | **Изход** | |
| 5139464478966.458 | decimal | |
| **Вход** | **Изход** | |
| -87616443766475.42 | decimal | |
| **Вход** | **Изход** | |
| 51390464478966.45 | decimal | |
| **Вход** | **Изход** | |
| -876160443766475.4 | decimal | |
| **Вход** | **Изход** | |
| 513904644708966.7 | decimal | |
| **Вход** | **Изход** | |
| -8761607443766475 | decimal | |
| **Вход** | **Изход** | |
| 5139046944708966 | decimal | |
| **Вход** | | **Изход** |
| 0.00000000000000000000 | | decimal |
| **Вход** | | **Изход** |
| 1000000000000000000.00000000000000000000 | | decimal |
| **Вход** | | **Изход** |
| 100000000000000000000 | | decimal |
| **Вход** | | **Изход** |
| 1.00000000000000000000000000 | | decimal |
| **Вход** | | **Изход** |
| 1.000000000000000000000000001 | | decimal |
|  | |  |

# Задача 3. Number precision 0.000001 / Изчисляване на числа с прецизност 0.000001

|  |  |
| --- | --- |
| Задача 3. Number precision 0.000001 / Изчисляване на числа с прецизност 0.000001 | |
| **Условие**  На входа получавате две числа P и Q с точност до 0.000001  На стандартния изход трябва да отпечатате сумата на тези числа с точност до 0.000001 | |
| **Описание на входа**  На първия ред се въвежда **P**  На втория ред ред се въвежда **Q**  **Ограничения:**  0 <= **P** <= 2000000000  0 <= **Q** <= 2000000000  **Примерен вход:**  12.758751  18.464645 | |
| **Описание на изхода**  На единствения ред да се изведе сумата с тояност до 0.000001  **Примерен изход:**  31.223396 | |
| **Анализ на задачата**  Резултата хубаво е с точност до 6ти знак, но когато числото има 2 знака преди точката то не може да се побере в float.  Резултата хубаво е с точност до 6ти знак, но когато числото има 10 знака преди точката то не може да се побере в float и double.    Използвайме типа данни **decimal** защото той е най-точен за подобни операции | |
| **Решение (сорс код)** | |
| using System;  namespace NumberPrecision  {  class Program  {  static void Main(string[] args)  {  decimal p = decimal.Parse(Console.ReadLine());  decimal q = decimal.Parse(Console.ReadLine());  decimal sum = p + q;  Console.WriteLine(sum);  }  }  } | |
| **Тестове**   1. **Тестване с примерен вход** 2. **Тестване с номинален случай** 3. **Тестване с номинален случай** 4. **Тестване с номинален случай** 5. **Тестване с случай който числата могат да бъда float double и decimal но трябва да се иползва форматиране защото има завършващи нули и резултата също може да бъде от трите възможни типа** 6. **Тестване с случай който числата могат да бъда float double и decimal но трябва да се иползва форматиране защото има завършващи нули и резултата също може да бъде от double или decimal** 7. **Тестване с случай който числата могат да бъда float double и decimal защото има завършващи нули и резултата също може да бъде от double или decimal** 8. **Тестване с случай който числата могат да бъда double и decimal но трябва да се иползва форматиране защото има завършващи нули и резултата също може да бъде от double или decimal** 9. **Тестване с случай който числата могат да бъда double и decimal и резултата също може да бъде от double или decimal** 10. **Тестване с случай който числата могат да бъда double и decimal но трябва да се иползва форматиране защото има завършващи нули и резултата също може да бъде от decimal** 11. **Тестване с случай който числата могат да бъда double и decimal резултата също може да бъде от decimal** 12. **Тестване с случай който числата могат да бъда decimal но трябва да се иползва форматиране защото има завършващи нули и резултата също може да бъде от decimal** 13. **Тестване с случай който числата могат да бъда decimal резултата също може да бъде от decimal** 14. **Тестване с случай който числата първото е горна граница и резултата има нужда форматиране** 15. **Тестване с случай който числата първото е горна граница и резултата** 16. **Тестване с случай който числата второто е горна граница и резултата има нужда форматиране** 17. **Тестване с случай който числата второто е горна граница и резултата** 18. **Тестване и двете числа да са горна граница** 19. **Тестване с случай който числата първото е горна граница, а второто долна граница** 20. **Тестване с случай който числата първото е долна граница, а второто горна граница** | |
| **Вход** | **Изход** |
| 12.758751  18.464645 | 31.223396 |
| **Вход** | **Изход** |
| 4.123456  3.123440 | 7.246896 |
| **Вход** | **Изход** |
| 4.123453  3.123443 | 7.246896 |
| **Вход** | **Изход** |
| 4.123451  3.123441 | 7.246892 |
| **Вход** | **Изход** |
| 1.580000  2.430000 | 4.010000 |
| **Вход** | **Изход** |
| 1.580000  9.580000 | 11.160000 |
| **Вход** | **Изход** |
| 1.581234  9.581234 | 11.162468 |
| **Вход** | **Изход** |
| 13.610000  42.420000 | 56.030000 |
| **Вход** | **Изход** |
| 19.151741  43.432668 | 62.584409 |
| **Вход** | **Изход** |
| 100000000.010000  950000000.070000 | 1050000000.080000 |
| **Вход** | **Изход** |
| 100000000.010000  950000000.070001 | 1050000000.080001 |
| **Вход** | **Изход** |
| 1050000000.080000  1250000000.080000 | 2300000000.160000 |
| **Вход** | **Изход** |
| 1050000000.080000  1250000000.080001 | 2300000000.160001 |
| **Вход** | **Изход** |
| 2000000000.000000  1.000000 | 2000000001.000000 |
| **Вход** | **Изход** |
| 2000000000.000000  1.123456 | 2000000001.123456 |
| **Вход** | **Изход** |
| 1.000000  2000000000.000000 | 2000000001.000000 |
| **Вход** | **Изход** |
| 1.123456  2000000000.000000 | 2000000001.123456 |
| **Вход** | **Изход** |
| 2000000000.000000  2000000000.000000 | 4000000000.000000 |
| **Вход** | **Изход** |
| 2000000000.000000  0.000000 | 2000000000.000000 |
| **Вход** | **Изход** |
| 0.000000  2000000000.000000 | 2000000000.000000 |
| 0.000000  0.000000 | 0.000000 |
|  |  |
|  |  |

# Задача 4. Heximal to Decimal Format / Превръщане от шестнайсетичен в десетичен вид

|  |  |
| --- | --- |
| Задача 4. Heximal to Decimal Format / Превръщане от шестнайсетичен в десетичен вид | |
| **Условие**  Напишете програма която при въведено число **N** шестнайсетичен вид го изписва на конзолата в десетичен.  **Като условие на същата задача с малка предисторийка следното условие:**  Иван и Георги били най-добри приятели и имали огромно доверие. Иван бил банкер, а Георги бизнесмен. Георги държал парите си в банката на Иван. Но един ден когато Георги пазарувал и след като платил в магазина с картата си той видял, че сумата по неговата сметка се различава драстично. Ядосал се той на Иван защото си мислил, че Иван го е прецакал. Но Иван казал че те са минали в 16 запис на числата на свойте клиенти защото сумите иначе били много големи и трудни за пресмятане. Но Георги не рабира от 16-тични числа и затова ви плаща на вас да му разработите софтуер, който да превъща неговата сметка в нормален 10-тичен запис за да знае колко пари има. | |
| **Описание на входа**  На единствения ред се въвежда **N** в шестнайсетичен вид  **Ограничения:**  0 <= **N** <= 7FFFFFFF  **Примерен вход:**  1F | |
| **Описание на изхода**  На единствения ред се извежда числото в десетичен формат  **Примерен изход:**  31 | |
| **Анализ на задачата**  Четем шестнасетичното числото като стринг  Използваме от библиотеката System.Convert  Convert.ToInt32(strng,16);  Което преобразува от шестнасетичното число записано в стринг десетично | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace HexToDec  {  class HexToDec  {  static void Main(string[] args)  {  string hex = Console.ReadLine();  int num = Convert.ToInt32(hex, 16);  Console.WriteLine(num);  }  }  } | |
| **Тестове**   1. Тестваме с 1F 2. Тестваме с 0 3. Тестваме с 7FFFFFF 4. Тестваме с 1234567 5. Тестваме с 200 6. Тестваме с 5000 7. Тестваме с 15000 8. Тестваме с 999 9. Тестваме с 1000 10. Тестваме с 0800 11. Тестваме с 800 | |
| **Вход** | **Изход** |
| 1F | 31 |
| **Вход** | **Изход** |
| 0 | 0 |
| **Вход** | **Изход** |
| 7FFFFFF | 134217727 |
| **Вход** | **Изход** |
| 1234567 | 19088743 |
| **Вход** | **Изход** |
| 100 | 256 |
| **Вход** | **Изход** |
| 5000 | 20480 |
| **Вход** | **Изход** |
| 15000 | 86016 |
| **Вход** | **Изход** |
| 999 | 2457 |
| **Вход** | **Изход** |
| 1000 | 4096 |
| **Вход** | **Изход** |
| 0800 | 2048 |
| **Вход** | **Изход** |
| 800 | 2048 |
|  |  |

# Задача5 Unicode symbol To char

|  |  |
| --- | --- |
| Задача 5. Unicode Symbol To Char / Преобразуване на unicode code към char | |
| **Условие**  На входа ще получите Uniccode на даден symbol .  Вие трябва да изкарате символа на конзолата | |
| **Описание на входа**  На единствения ред ще получите Uniccode на даден symbol без ‘\u’  **Ограничения:**  **unicode могат да бъдат изведени без нуждата от допълнителна настойка на козолата**  **Примерен вход:**  48 | |
| **Описание на изхода**  На единствения ред трябва да изведете символа  **Примерен изход:**  **H** | |
| **Анализ на задачата**  Когато получим даден символ от конзолата го прочитаме като string защото представянето на символ чрез unicode става с шестнайсетично число.  Преобразуваме шестнайсетичното число в десетично с Convert.  И полученото цяло число castvame към char | |
| **Решение (сорс код)** | |
| using System;  namespace charUnicode  {  class Program  {  static void Main(string[] args)  {  string unicodeSymbol = Console.ReadLine();  int numInHex = Convert.ToInt32(unicodeSymbol, 16);  char ch = (char)numInHex;  Console.WriteLine(ch);  }  }  } | |
| **Тестове**  Всички тестове са номинален случай, защото ограничението е да могат да се изпишат на конзолата без допълнителна настройка. И няма как да се тестват интересни случай   1. Тестваме с примерния вход 2. Тестваме с номинални случай 3. Тестваме с номинални случай 4. Тестваме с номинални случай 5. Тестваме с номинални случай 6. Тестваме с номинални случай 7. Тестваме с номинални случай 8. Тестваме с номинални случай 9. Тестваме с номинални случай 10. Тестваме с номинални случай 11. Тестваме с номинални случай 12. Тестваме с номинални случай 13. Тестваме с номинални случай 14. Тестваме с номинални случай 15. Тестваме с номинални случай 16. Тестваме с номинални случай 17. Тестваме с номинални случай 18. Тестваме с номинални случай 19. Тестваме с номинални случай 20. Тестваме с номинални случай 21. Тестваме с номинални случай | |
| **Вход** | **Изход** |
| 0048 | H |
| **Вход** | **Изход** |
| 005F | \_ |
| **Вход** | **Изход** |
| 0021 | ! |
| **Вход** | **Изход** |
| 0068 | h |
| **Вход** | **Изход** |
| 002F | / |
| **Вход** | **Изход** |
| 004F | O |
| **Вход** | **Изход** |
| 0021 | ! |
| **Вход** | **Изход** |
| 0022 | " |
| **Вход** | **Изход** |
| 0023 | # |
| **Вход** | **Изход** |
| 00A3 | œ |
| **Вход** | **Изход** |
| 00A9 | © |
| **Вход** | **Изход** |
| 00AE | ® |
| **Вход** | **Изход** |
| 0027 | ' |
| **Вход** | **Изход** |
| 00A7 | § |
| **Вход** | **Изход** |
| 00BC | ¼ |
| **Вход** | **Изход** |
| 00B2 | ² |
| **Вход** | **Изход** |
| 005E | ^ |
| **Вход** | **Изход** |
| 003B | ; |
| **Вход** | **Изход** |
| 007B | { |
| **Вход** | **Изход** |
| 0051 | Q |

# Задача 6 Ismale /Дали потребителя е мъж или жена

|  |  |
| --- | --- |
| Задача 6. Ismale /Дали потребителя е мъж или жена | |
| **Условие**  Напишете програма която когато въведеш своя пол ти извежда съотвено You are male. или You are female. | |
| **Описание на входа**  Ако потребителя е мъж въвежда TRUE  Ако потребителя е жена въвежда FALSE  **Ограничения:**  **Примерен вход:**  TRUE | |
| **Описание на изхода**  Ако потребителя е мъж ти извежда You are male  Ако потребителя е мъж ти извежда You are female  **Примерен изход:**  You are male. | |
| **Анализ на задачата**  При деклариране на bool променлива isMale и задаване на своя си пол. Задачата не може да бъде тествана защото не знаем дали този който пише тази задача е мъж или жена. | |
| **Решение (сорс код)** | |
| using System;  namespace IsMale  {  class IsMale  {  static void Main(string[] args)  {  bool isMale = bool.Parse(Console.ReadLine());  if (isMale == true)  {  Console.WriteLine("You are male");  }  else  {  Console.WriteLine("You are female");  }  }  }  } | |
| **Тестове**   1. Тестване примерен вход 2. Тестваме с TRUE 3. Тестваме с FALSE | |
| **Вход** | **Изход** |
| TRUE | You are male |
| **Вход** | **Изход** |
| TRUE | You are male |
| **Вход** | **Изход** |
| FALSE | You are female |

# Задача 7. String Concat To Object / Записване в Object

|  |  |
| --- | --- |
| Задача 7. String Concat To Object / Записване в Object | |
| **Условие**   Потребителя получава от входа два думи. На изхода трябва да изведете двете думи разделени с интервал | |
| **Описание на входа**  На първия ред получавате една дума  На втория ред получавате една дума  **Ограничения:**  **Примерен вход:**  Hello  World | |
| **Описание на изхода**  На единствения ред се извеждат двете думи разделени с интервал  **Примерен изход:**  Hello World | |
| **Анализ на задачата**  При декларираме две променливи от тип string hello=”Hello” , string world=”World” и още 1 променлива object helloPlusWorld = hello +” “+ world; | |
| **Решение (сорс код)** | |
| using System;  namespace ConcatWords  {  class ConcatWords  {  static void Main(string[] args)  {  string word1 = Console.ReadLine();  string word2 = Console.ReadLine();  object word1PlusWord2 = word1 + " " + word2;  Console.WriteLine(word1PlusWord2);  }  }  } | |
| **Тестове**  Тук не може да тестваме дали потребителя записва в object но може da проверим дали има интервал   1. Тестваме с две думи Hello World 2. Тестваме с две думи John Smit 3. Тестваме с две думи Patric Ivanov 4. Тестваме с две думи George Anderson 5. Тестваме с две думи The Hack 6. Тестваме само с втора дума Hack 7. Тестваме само с първа дума Hack 8. Тестваме без нито 1 дума 2 празни ред 9. Тестваме с две думи The Hack 10. Тестваме с две думи The Hack | |
| **Вход** | **Изход** |
| Hello  World | Hello World |
| **Вход** | **Изход** |
| John  Smit | John Smit |
| **Вход** | **Изход** |
| Patric  Ivanov | Patric Ivanov |
| **Вход** | **Изход** |
| George  Anderson | George Anderson |
| **Вход** | **Изход** |
| The  Hack | The Hack |
| **Вход** | **Изход** |
| Hack | Hack |
| **Вход** | **Изход** |
| Hack | Hack |
| **Вход** | **Изход** |
|  |  |
| **Вход** | **Изход** |
| The  Hack | The Hack |
| **Вход** | **Изход** |
| The  Hack | The Hack |
| **Вход** | **Изход** |
| The  Hack | The Hack |

# Задача 8. Casting type / Кастване на типове

|  |  |
| --- | --- |
| Задача 8. Casting type / Кастване на типове | |
| **Условие**    Декларирайте две променливи от тип **string** и им присвоете стойности "Hello" и "World". Декларирайте променлива от тип **object** и и присвоете стойността на конкатенацията на двете променливи от тип **string** (не изпускайте интервала по средата). Декларирайте трета променлива от тип **string** и я инициализирайте със стойността на променливата от тип**object** ( трябва да използвате type casting). | |
| **Описание на входа**  1-ви ред получавате 1 дума  2-ри ред получавате 1 дума  **Ограничения:**  Данните са винаги валидни.  **Примерен вход:**  Hello  World | |
| **Описание на изхода**  На единствения ред изведете двете думи разделени с интервал  **Примерен изход:**  Hello World | |
| **Анализ на задачата**  При декларираме две променливи от тип string hello=”Hello” , string world=”World” променлива object helloPlusWorld = hello +” “+ world; и променлива от тип string helloPlusWorldString = helloPlusWorldString = (string)helloPlusWorld;  Със (тип-променлива)променлива взимаме нова променлива от този тип – “тип-променлива” | |
| **Решение (сорс код)** | |
| using System;  namespace castObjectToString  {  class castObjectToString  {  static void Main(string[] args)  {  string word1 = Console.ReadLine();  string word2 = Console.ReadLine();  object word1PlusWord2 = word1 + " " + word2;  string word1PlusWord2String = (string)word1PlusWord2;  Console.WriteLine(word1PlusWord2String);  }  }  } | |
| **Тестове**  Тук не може да тестваме дали потребителя cast-ва от object към string но може da проверим дали има интервал   1. Тестваме с две думи Hello World 2. Тестваме с две думи John Smit 3. Тестваме с две думи Patric Ivanov 4. Тестваме с две думи George Anderson 5. Тестваме с две думи The Hack 6. Тестваме само с втора дума Hack 7. Тестваме само с първа дума Hack 8. Тестваме без нито 1 дума 2 празни 9. Тестваме с две думи The Hack 10. Тестваме с две думи The Hack | |
| **Вход** | **Изход** |
| Hello  World | Hello World |
| **Вход** | **Изход** |
| John  Smit | John Smit |
| **Вход** | **Изход** |
| Patric  Ivanov | Patric Ivanov |
| **Вход** | **Изход** |
| George  Anderson | George Anderson |
| **Вход** | **Изход** |
| The  Hack | The Hack |
| **Вход** | **Изход** |
| Hack | Hack |
| **Вход** | **Изход** |
| Hack | Hack |
| **Вход** | **Изход** |
|  |  |
| **Вход** | **Изход** |
| The  Hack | The Hack |
| **Вход** | **Изход** |
| The  Hack | The Hack |
| **Вход** | **Изход** |
| The  Hack | The Hack |

# Задача 9 Quoted String / Цитати

|  |  |
| --- | --- |
| Задача 9 Quoted String / Цитати | |
| **Условие**    Декларирайте две променливи от тип string и им присвоете стойност "The "use" of quotations causes difficulties." (без първите и последни кавички). В едната променлива използвайте quoted string, а в другата не го използвайте. | |
| **Описание на входа**  **Ограничения:**  **Примерен вход:** | |
| **Описание на изхода**  **Примерен изход:** | |
| **Анализ на задачата**  Неможах да измисля и да променя задачата, така че да стане с някакъв вход и да бъде възможна за тестове. Затова като компенсация написах 2 задачи допълнително с малко променени условия с тези на 1ва и на 10та задача. Те са под номера 14 и 15 съответно. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace QuotedString  {  class QuotedString  {  static void Main(string[] args)  {  string someText = "The \" use \" of quotations causes difficulties.";  Console.WriteLine(someText);  }  }  } | |
| **Тестове** | |
| **Вход** | **Изход** |
|  |  |

# Задача 10. Drawing Heart / Рисуване по конзолата

|  |
| --- |
| Задача 10. Drawing Heart / Рисуване по конзолата |
| **Условие**  Напишете програма, която принтира фигура във формата на сърце със знака "o" при зададено число **N.** Ширината на сърцето е 6\***N** |
| **Описание на входа**  На единствения ред получавате **N**  **Ограничения:**  1<= **N** <=25  **Примерен вход:**  3 |
| **Описание на изхода**  Извежда се изрисуваното сърце  **Примерен изход:**  **C:\Users\Boyan Zhelyazkov\Desktop\heart3.png**  **Примерен вход :**  4  **Примерен изход:**  **C:\Users\Boyan Zhelyazkov\Desktop\heart4.png** |
| **Анализ на задачата**  **От най-широката част виждаме че 1ви ред винаги е изместан с 6 интервала.**  Забелязваме че при нечетно **N** най-долния ред има 2 О-та  А при четно **N** най-долния ред има 4 О-та  Също така че и двере сърца имат **6** особенни реда  Стъпка 1. Първи ред започва в 6 интервала + N О-та + 2\*6 интервала + N О-та  Стъпка 2. За ред 2 и 3 намаляме всеки път интервалите с **два** и увеличаване О-та с **две**  Изпълнявме стъпка 1  След 3път О-та станали с ширина **6\*N/2**  a интервалите нула затова конкатинираме стринга с О със себе си  Последната стъпка е да начертаем сърцето надолу  **От най-широката част виждаме че всеки ред намаля с по 4ри О-та и увеличава от ляво с 2 интервала.**  Това с while цикъл го правим докато О-тата са повече от 4 |
| **Решение (сорс код)** |
| using System;  using System.Linq;  namespace Heart  {  class Heart  {  static void Main(string[] args)  {  int N = int.Parse(Console.ReadLine());  string sixSpaces = " ";  string nO = new string('O', N);  for (int i = 0; i < 3; i++)  {  Console.Write(sixSpaces);  Console.Write(nO);  Console.Write(sixSpaces);  Console.Write(sixSpaces);  Console.WriteLine(nO);  sixSpaces = sixSpaces.Remove(sixSpaces.Length - 2, 2);  nO += "OOOO";  }  nO += nO;  for (int i = 0; i < 3; i++)  {  Console.WriteLine(nO);  }  while (nO.Length > 4)  {  nO = nO.Remove(nO.Length - 4, 4);  sixSpaces += " ";  Console.Write(sixSpaces);  Console.WriteLine(nO);  }  }  }  } |
| **Тестове**  Тестовете са обемни и затова са в папката с тестовете за всички задачи задача Heart |

# Задача 11 Drawing Triangle/ изчертаване на триъгълник

|  |
| --- |
| Задача 11. Drawing Triangle / изчертаване на триъгълник |
| **Условие**  Напишете програма, която принтира на конзолата равнобедрен триъгълник, като страните му са очертани от символа звездичка "©". На стандартния вход получавате **N** - височината на триъгълника. |
| **Описание на входа**  На единствения ред получавате **N**  **Ограничения:**  2 <= **N** <= 26  **Примерен вход:**  8 |
| **Описание на изхода**  Изрисувате триъгълника.  **Примерен изход:**  **C:\Users\Boyan Zhelyazkov\Desktop\triangle8.png** |
| **Анализ на задачата**  От примерния изход виждаме че започва с 1 -> @  На всеки следващ ред @ се увеличават с 2 , а празните интервали отляво намалят с 1.  С инициализащия какъв символ и колко създаваме два стринга.  Стринга за интервалите и тяхната бройката е равна на N - i, където i е текущия ред  Стринга за символите и тяхната бройка е равна на 2\*i-1, където i е текущия ред |
| **Решение (сорс код)** |
| using System;  namespace Triangle  {  class Triangle  {  static void Main(string[] args)  {  int N = int.Parse(Console.ReadLine());  for (int i = 1; i <= N; i++)  {  string spaces = new string(' ', N - i);  string symbols = new string('©', 2\*i-1);  Console.Write(spaces);  Console.WriteLine(symbols);  }  }  }  } |
| **Тестове**  Тестовете са обемни и затова са в папката с тестовете за всички задачи задача Triangle |

# Задача 12. Company Employees / Работници във фирма

|  |
| --- |
| Задача 12. Company Employees / Работници във фирма |
| **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 (‘м’ или ‘ж’) и уникален номер на служителя (27560000 до 27569999). Декларирайте необходи­мите променливи, нужни за да се запази информацията за един служи­тел, като използвате подходящи типове данни и описателни имена.  Програмата трябва да изведе цялата информация за служителя + типа данни за всяко негово поле. |
| **Описание на входа**  Получавате пет реда  1-ви ред: името на служителя  2-ри ред: фамилия на служителя  3-ти ред: години на служителя  4-ти ред: пол  5-ти ред: Unice number  **Ограничения:**  27560000 < UniceNumber < 27569999  Първото име не е празен string  Фамилията не е празен string  Годините не са повече от 60.  Пола е male или female  **Условия:**  Данните са винаги валидни.  **Примерен вход:**  Nikola  Ivanov  21  Male  27563234 |
| **Описание на изхода**  5 реда  1-ви ред: името на служителя + “ -> избрания от вас тип за име”  2-ри ред: фамилия на служителя +” -> избрания от вас тип за име”  3-ти ред: години на служителя +“-> избрания от вас тип за име”  4-ти ред: пол + “ -> избрания от вас тип за име”  5-ти ред: Uniqe number + “ -> избрания от вас тип за име”  **Примерен изход:**  Задачата не може да има нулев тест за пример защото се обезсмисля решаване на задачата за избор на най-подходящ тип. |
| **Анализ на задачата**  За първо име и фамилия избираме string  Годините са винаги по-големи от 0 и най-много 60 затова избираме тип byte  За пола на служителя го прочитаме като стринг в една променлива. Но го прехвърляме в една bool прменлива защото стойностите са само 2.  Например в променлива bool isMale=true;  правим една проверка ако прочетения пол е female променяме isMale=false;  За uniqueNumber избираме int защото и двете стойности (27560000 - 27569999) влизат в границите на int  Накрая според формата на изхода извежаме променливите + :“ техния тип“ |
| **Решение (сорс код)** |
| using System;  namespace Emploee  {  class Emploee  {  static void Main(string[] args)  {  string firstName = Console.ReadLine();  string lastName = Console.ReadLine();  byte age = byte.Parse(Console.ReadLine());  string sex = Console.ReadLine();  int number = int.Parse(Console.ReadLine());  bool isMale = true;  if (sex=="female")  {  isMale = false;  }  Console.WriteLine(firstName+" -> string");  Console.WriteLine(lastName + " -> string");  Console.WriteLine(age + " -> byte");  Console.WriteLine(isMale + " -> bool");  Console.WriteLine(number + " -> int");  }  }  } |
| **Тестове**  Тестовете са обемни и затова са в папката с тестовете за всички задачи задача Company Emploees |

# Задача 13. Swap two integers / размяна на два integers

|  |  |
| --- | --- |
| Задача 13. Swap two integers / размяна на два integers | |
| **Условие**   1. Декларирайте две променливи от тип int. Прочете стойностите им от клавиатурата. Разменете стойностите им и ги отпечатайте. | |
| **Описание на входа**  Получавате два реда  1-ви ред: първото число  2-ри ред: второто число  **Ограничения:**  **Условия:**  Данните са винаги валидни.  **Примерен вход:**  10  5 | |
| **Описание на изхода**  Извеждате на два реда  1-ви ред: първото число  2-ри ред: второто число  **Примерен изход:**  5  10 | |
| **Анализ на задачата**  Съдаваме 3та променлива от тип int. и присвояваме първата променлива.  Напрмер int third=first.  След това на първата променлива присвояваме втората променлива.  И накрая на втората променлива присвояваме третата промелива.  След това размяната е на лице. | |
| **Решение (сорс код)** | |
| using System;  namespace swap  {  class swap  {  static void Main(string[] args)  {  int fistNumber = int.Parse(Console.ReadLine());  int secondNumber = int.Parse(Console.ReadLine());  int tempNumber = fistNumber;  fistNumber = secondNumber;  secondNumber=tempNumber;  Console.WriteLine(fistNumber);  Console.WriteLine(secondNumber);  }  }  } | |
| **Тестове**   1. Тестваме с номинален случай 2. Тестваме с 0 3. Тестваме с -128 4. Тестваме с - 32768 5. Тестваме с - 2147483648 6. Тестваме с - 9223372036854775808 7. Тестваме с 127 8. Тестваме с 255 9. Тестваме с 32767 10. Тестваме с 65535 11. Тестваме с 2147483647 12. Тестваме с 4294967295 13. Тестваме с 9223372036854775807 14. Тестваме с 18446744073709551615 15. Тестваме с -129 16. Тестваме с 128 17. Тестваме с -32769 18. Тестваме с 32768 19. Тестваме с -2147483649 20. Тестваме с 2147483648 21. Тестваме с 9223372036854775808 22. Тестваме с 256 23. Тестваме с 65536 24. Тестваме с 4294967296 25. Тестваме с 123456789123456789 | |
| **Вход** | **Изход** |
| +  1553 | int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  0 | byte, int, long, sbyte, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| -  -128 | int, long, sbyte, short |
| **Вход** | **Изход** |
| -  -32768 | int, long, short |
| **Вход** | **Изход** |
| -  -2147483648 | int, long |
| **Вход** | **Изход** |
| -  -9223372036854775808 | long |
| **Вход** | **Изход** |
| +  127 | byte, int, long, sbyte, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  255 | 255  byte, int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  32767 | int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  65535 | int, long, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  2147483647 | int, long, uint, ulong |
| **Вход** | **Изход** |
| +  4294967295 | long, uint, ulong |
| **Вход** | **Изход** |
| +  9223372036854775807 | long, ulong |
| **Вход** | **Изход** |
| +  18446744073709551615 | ulong |
| **Вход** | **Изход** |
| -  -129 | int, long, short |
| **Вход** | **Изход** |
| +  128 | byte, int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| -  -32769 | int, long |
| **Вход** | **Изход** |
| +  32768 | int, long, uint, ulong, ushort |
| **Вход** | **Изход** |
| -  -2147483649 | long |
| **Вход** | **Изход** |
| +  2147483648 | long, uint, ulong |
| **Вход** | **Изход** |
| +  +9223372036854775808 | ulong |
| **Вход** | **Изход** |
| +  256 | int, long, short, uint, ulong, ushort |
| **Вход** | **Изход** |
| +  65536 | int, long, uint, ulong |
| **Вход** | **Изход** |
| +  4294967296 | long, ulong |
| **Вход** | **Изход** |
| +  123456789123456789 | long, ulong |
|  |  |

# Задача 14. Drawing Romb / Рисуване по конзолата ромб

|  |
| --- |
| Задача 14. Drawing Romb / Рисуване по конзолата ромб |
| **Условие**  Напишете програма, която принтира фигура във формата на ромб застанал на единия си ъгъл.  Използвайте ‘/’ и ‘\’  На входа ще получите **N** дължина на страната на ромба |
| **Описание на входа**  На единствения ред получавате **N**  **Ограничения:**  1<= **N** <=25 |
| **Описание на изхода**  Извежда се изрисуваното сърце  **Примерен вход :**  5  **Примерен изход:**  **C:\Users\Boyan Zhelyazkov\Desktop\romb5.png** |
| **Анализ на задачата**  Виждаме че в най-широката си част ромба има 2\***N** елемент  На първия ред има два елемента и празни места от ляво и дясно (местата от двете странни са равен брой), отук следва че **N**-1 празни места има на 1ви ред, 1 наклонено няляво и 1 нядясно.  На веки следващ ред до **N-ти** наклонините на едната и другата страна се увеличават с 1. А празнияте места от няво надясно намалят с 1.  Така се изчисляват следните три компонента до **N-ти** ред  string spaces = new string(' ', N - i);  string onLeft = new string('/', i);  string onRight = new string('\\', i);  И се изведат на екрана в последователност spaces,onLeft, onRight с нов ред накрая.  От **N-ти** до 2\*N ред нещата се обърщат. Но ние пак може да го разгледаме като поредица от 1 до N  А за втората част се изчисляват по следния начин  string spaces = new string(' ', N - i);  string onLeft = new string('/', i);  string onRight = new string('\\', i);  И се извеждат в последователнос spaces onRight onLeft с нов ред накрая. |
| **Решение (сорс код)** |
| using System;  namespace Romb  {  class Romb  {  static void Main(string[] args)  {  int N = int.Parse(Console.ReadLine());    for (int i = 1; i <= N; i++)  {  string spaces = new string(' ', N - i);  string onLeft = new string('/', i);  string onRight = new string('\\', i);  Console.Write(spaces);  Console.Write(onLeft);  Console.WriteLine(onRight);  }  for (int i = N; i > 0; i--)  {  string spaces = new string(' ', N - i);  string onLeft = new string('/', i);  string onRight = new string('\\', i);  Console.Write(spaces);  Console.Write(onRight);  Console.WriteLine(onLeft);    }  }  }  } |
| **Тестове**  Тестовете са обемни и затова са в папката с тестовете за всички задачи задача Romb |
|  |

# Задача 15 Most suitable type for Value / Най-подходящ тип за дадено число

|  |  |
| --- | --- |
| Задача 15. Most suitable type for Value / Най-подходящ тип за дадено число | |
| **Условие**  На първия ред получаваме какво число ще получим за положително символа “+” за отрицателно символа “-”  Получваме едно цяло число **N** от конзолата и да кажем кой от всички целочислени типовете **sbyte**, **byte**, **short**, **ushort**, **int**, **uint**, **long** и **ulong** е “най-подходящ за него”.  Като за “най-подходящ за него” се разбира най-малкия възможен в който променливата може да бъде съхранена.  Градацията на типовете е следната: **sbyte**, **byte**, **short, ushort, int, uint, long, ulong.**  **Като условие на същата задача с малка предисторийка следното условие:**  Ники е много талантлив по иформатика и може да преценя много добре кой тип (**sbyte**, **byte**, **short, ushort, int, uint, long, ulong**) е най-подходящ за дадено цяло число.Той е решил че това е талант подходящ за книгата на гинес. Изпитваща комисия от Гинес иска да прецени дали наистина Ники е толкова талантлив в това което смята и дали заслужава да влезе в книгата на рекорди Гинес. Но комисията няма и наидея затова какво са типове данни в информатиката и кой е „най-подходящия тип“ за дадено число. Затова комисията Ви плаща за да и направите програма, която при дадено цяло число **N,** кой от всички целочислени типовете **sbyte**, **byte**, **short**, **ushort**, **int**, **uint**, **long** и **ulong** е “най-подходящ за него”.  Като за “най-подходящ за него” се разбира най-малкия възможен в който променливата може да бъде съхранена.  Градацията на типовете е следната: **sbyte**, **byte**, **short, ushort, int, uint, long, ulong.** | |
| **Описание на входа**  На първия ред получавате символа + или – които определя знака на числото.  На втория ред получвате числото **N**  **Ограничения:**  –9223372036854775808 <= **N** <= 18446744073709551615.  **Условия:**  Данните са винаги валидни.  Числото 0 считаме за положително число.  **Примерен вход:**  +  1553 | |
| **Описание на изхода**  На единствения изход извеждаме стринг с всички променливи в азбучен ред в които може да бъде записана променливата.  **Примерен изход:**  short | |
| **Анализ на задачата**  За да може да вкючим долната граница на long и горната на ulong ние в началото правим проверка дали числото е положително и ако е така декларираме ulong и прочитаме число от конзолата което може да е максималното за типа ulong и ако сме избрали друг тип няма да можем да продължим.  Ако числото е отрицателно така декларираме long и прочитаме число от конзолата което може да е минималното за типа long и ако сме избрали друг тип няма да можем да продължим.  За да изкраме типовете в подреден вид без да изполваме масиви и сортирания, когато видим къде сме просто от условието премахваме по малките типове от данни и извеждаме останалите типове който са в сортиран вид за числото.  За да разберем кой тип е най-подходящ на дадена стойност започваме с една проверка дали стойността е положителна за да знаем дали да вкючваме променливите типове без знак. След това продължаваме проверка от най-малкия интервал на оста ако той съовества и извеждаме типа, който съответства на интервала в който сме. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace MostSuitableType  {  class MostSuitableType  {  static void Main(string[] args)  {  char sign = char.Parse(Console.ReadLine());  if (sign == '+')  {  ulong inputNumber = ulong.Parse(Console.ReadLine());  if (inputNumber <= 127)  {  Console.WriteLine("sbyte");  }  else if (inputNumber <= 255)  {  Console.WriteLine("byte");  }  else if (inputNumber <= 32767)  {  Console.WriteLine("short");  }  else if (inputNumber <= 65535)  {  Console.WriteLine("ushort");  }  else if (inputNumber <= 2147483647)  {  Console.WriteLine("int");  }  else if (inputNumber <= 4294967295)  {  Console.WriteLine("uint");  }  else if (inputNumber <= 9223372036854775807)  {  Console.WriteLine("long");  }  else if (inputNumber <= 18446744073709551615)  {  Console.WriteLine("ulong");  }  }  else  {  long inputNumber = long.Parse(Console.ReadLine());  if (inputNumber >= -128)  {  Console.WriteLine("sbyte");  }  else if (inputNumber >= -32768)  {  Console.WriteLine("short");  }  else if (inputNumber >= -2147483648)  {  Console.WriteLine("int");  }  else if (inputNumber >= -9223372036854775808)  {  Console.WriteLine("long");  }  }  }  }  } | |
| **Тестове**   1. Тестваме с номинален случай 2. Тестваме с 0 3. Тестваме с -Минимална стойност за byte 4. Тестваме с - Минимална стойност за short 5. Тестваме с - Минимална стойност за int 6. Тестваме с - Минимална стойност за long 7. Тестваме с - Mаксимална стойност за byte 8. Тестваме с - Mаксимална стойност за sbyte 9. Тестваме с - Mаксимална стойност за short 10. Тестваме с - Mаксимална стойност за ushort 11. Тестваме с - Mаксимална стойност за int 12. Тестваме с - Mаксимална стойност за uint 13. Тестваме с - Mаксимална стойност за long 14. Тестваме с - Mаксимална стойност за ulong 15. Тестваме с - Минимална стойност за sbyte -1 16. Тестваме с - Mаксимална стойност за sbyte+1 17. Тестваме с -Mинимална стойност за short-1 18. Тестваме с - Mаксимална стойност за short+1 19. Тестваме с -Mинимална стойност за int-1 20. Тестваме с - Mаксимална стойност за int+1 21. Тестваме с - Mаксимална стойност за long+1 22. Тестваме с - Mаксимална стойност за byte+1 23. Тестваме с - Mаксимална стойност за ushort+1 24. Тестваме с - Mаксимална стойност за int+1 25. Тестваме с - Mаксимална стойност за long+1 | |
| **Вход** | **Изход** |
| +  1553 | short |
| **Вход** | **Изход** |
| +  0 | sbyte |
| **Вход** | **Изход** |
| -  -128 | sbyte |
| **Вход** | **Изход** |
| -  -32768 | short |
| **Вход** | **Изход** |
| -  -2147483648 | int |
| **Вход** | **Изход** |
| -  -9223372036854775808 | long |
| **Вход** | **Изход** |
| +  127 | sbyte |
| **Вход** | **Изход** |
| +  100 | sbyte |
| **Вход** | **Изход** |
| +  255 | byte |
| **Вход** | **Изход** |
| +  32767 | short |
| **Вход** | **Изход** |
| +  65535 | ushort |
| **Вход** | **Изход** |
| +  2147483647 | int |
| **Вход** | **Изход** |
| +  4294967295 | uint |
| **Вход** | **Изход** |
| +  9223372036854775807 | long |
| **Вход** | **Изход** |
| +  18446744073709551615 | ulong |
| **Вход** | **Изход** |
| -  -129 | short |
| **Вход** | **Изход** |
| +  128 | byte |
| **Вход** | **Изход** |
| -  -32769 | int |
| **Вход** | **Изход** |
| +  32768 | ushort |
| **Вход** | **Изход** |
| -  -2147483649 | long |
| **Вход** | **Изход** |
| +  2147483648 | uint |
| **Вход** | **Изход** |
| +  +9223372036854775808 | ulong |
| **Вход** | **Изход** |
| +  256 | short |
| **Вход** | **Изход** |
| +  65536 | int |
| **Вход** | **Изход** |
| +  4294967296 | long |
| **Вход** | **Изход** |
| +  123456789123456789 | long |