# Решения на задачите от глава 3 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

**Кристиан Ташков**

Контакти: ktashkov@uni-sofia.bg

**Ивайло Дянков**

Контакти: idyankov@gmail.com

**Виктор Владимиров Букуров**

Контакти: vbookie@gmail.com

**Неджати Неджати Мехмед**

Контакти: nedjatymehmed@gmail.com

# Шаблон за описание на задачите

|  |  |
| --- | --- |
| Задача 01. Четно или нечетно | |
| **Условие**   Напишете израз, който да проверява дали дадено цяло число е четно или нечетно. | |
| **Описание на входа**  Като входни данни се дава едно цяло число. | |
| **Описание на изхода**  Програмата трябва да отпечатва “Even” или “Odd“ | |
| **Анализ на задачата**  Четем от конзолата даденото число и го записваме в променливата **inputValue.** Правим проверка с делене на 2 дали дава остатък 1, ако дава остатък значи е нечетно, в противен случай е четно. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace OddOrEvenNumber  {  class Program  {  static void Main(string[] args)  {  // Read the input number  int inputValue = int.Parse(Console.ReadLine());  // Checks if the number is even or odd  Console.WriteLine((0 == inputValue % 2) ? "Even" : "Odd");  }  }  } | |
| **Тестове**   * **Тестове с отрицателни числа** * **Тест с 0** * **Тестове с долната и горната граница на променливата от типа int.** | |
| **Вход** | **Изход** |
| 1 | Odd |
| **Вход** | **Изход** |
| 10 | Even |
| **Вход** | **Изход** |
| 0 | Even |
| **Вход** | **Изход** |
| 1111111 | Odd |
| **Вход** | **Изход** |
| -100 | Even |
| **Вход** | **Изход** |
| 123456789 | Odd |
| **Вход** | **Изход** |
| -123123123 | Odd |
| **Вход** | **Изход** |
| 666 | Even |
| **Вход** | **Изход** |
| -2147483648 | Even |
| **Вход** | **Изход** |
| 2147483647 | Odd |

|  |  |
| --- | --- |
| Задача 02. Цели числа, които се делят на 5 и на 7 | |
| **Условие**   Напишете булев израз, който да проверява дали дадено цяло число се дели на 5 и на 7 без остатък. | |
| **Описание на входа**  Като входни данни се дава едно цяло число. | |
| **Описание на изхода**  Изхода трябва да връща съответно „True” или „False“ | |
| **Анализ на задачата**  Четем от конзолата даденото число и го записваме в променливата **inputValue,** правим проверка дали числото едновременно се дели без остатък на 5 и на 7 като използваме логическо „и“ („&&“). | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace DivisibleBy5And7Simultaneously  {  class Program  {  static void Main(string[] args)  {  // Read the input number  int inputValue = int.Parse(Console.ReadLine());  // Check if the number is divisible  Console.WriteLine(((0 == inputValue % 5) && (0 == inputValue % 7)));  }  }  } | |
| **Тестове**   * **Тест с число което се дели само на 7 или само на 5** * **Тест с 0** * **Тест с близки до долната и горната граница на числата от типа int.** * **Тест с отрицателни числа** | |
| **Вход** | **Изход** |
| 35 | True |
| **Вход** | **Изход** |
| 5 | False |
| **Вход** | **Изход** |
| 7 | False |
| **Вход** | **Изход** |
| 0 | True |
| **Вход** | **Изход** |
| -105 | True |
| **Вход** | **Изход** |
| 123456789 | False |
| **Вход** | **Изход** |
| -123123123 | False |
| **Вход** | **Изход** |
| 575757 | False |
| **Вход** | **Изход** |
| -2147483648 | False |
| **Вход** | **Изход** |
| 214748625 | True |

|  |  |
| --- | --- |
| Задача 03. Третото число е Седем | |
| **Условие**  Напишете израз, който да проверява дали третата цифра (отдясно на ляво) на дадено цяло число е 7. | |
| **Описание на входа**  Като входни данни се дава едно цяло число. | |
| **Описание на изхода**  Изхода трябва да връща съответно „True” или „False“ | |
| **Анализ на задачата**  Четем от конзолата даденото число и го записваме в променливата **inputValue.**  Разделяме числото на 100 (за да преместим третата цифра в края на числото). След това използваме оператора за пресмятане на остатък от целочислено делене („%“). Делим на десет, за да получим последната цифра и проверяваме дали тя е 7. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace CheckIfTheThirdDigitOfANumberIs7  {  class Program  {  static void Main(string[] args)  {  // Read the input number  int inputNumber = int.Parse(Console.ReadLine());    // Get and check if the third digit of the number is 7  Console.WriteLine(7 == ((inputNumber / 100) % 10));  }  }  } | |
| **Тестове**   * **Проверяваме за по-малко от трицифрено число дали работи правилно програмата (например с числото „7“)** * **Проверяваме с отрицателни числа** * **Проверяваме с 0** * **Проверяваме граничните случаи за типа int.** | |
| **Вход** | **Изход** |
| 777 | True |
| **Вход** | **Изход** |
| 677 | False |
| **Вход** | **Изход** |
| 7 | False |
| **Вход** | **Изход** |
| 0 | False |
| **Вход** | **Изход** |
| 77777677 | False |
| **Вход** | **Изход** |
| 123456789 | True |
| **Вход** | **Изход** |
| -77 | False |
| **Вход** | **Изход** |
| 7171 | False |
| **Вход** | **Изход** |
| -2147483648 | False |
| **Вход** | **Изход** |
| 111111711 | True |

|  |  |
| --- | --- |
| Задача 04. Третият бит е 1 или 0 | |
| **Условие**   Напишете израз, който да проверява дали третия бит на дадено число е 1 или 0. | |
| **Описание на входа**  Като входни данни се дава едно цяло число. | |
| **Описание на изхода**  Изхода съответно трябва да бъде „0“ или „1“ | |
| **Анализ на задачата**  Четем от конзолата числото и го записваме в променливата **inputNumber**, вземаме третия бит от числото чрез побитово „и“, проверяваме дали е 0 или 1 и съответно отпечатваме отговора. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace FindTheThirdBitOfANumber  {  class Program  {  static void Main(string[] args)  {  // Read the input number  int inputNumber = int.Parse(Console.ReadLine());  // Get the third bit (counting from 0)  int mask = 0x8;  Console.WriteLine(0 != (inputNumber & mask) ? 1 : 0);  }  }  } | |
| **Тестове**   * **Тестове с отрицателни числа** * **Тестове с граничните случаи за типа int.** | |
| **Вход** | **Изход** |
| 0 | 0 |
| **Вход** | **Изход** |
| 8 | 1 |
| **Вход** | **Изход** |
| 1 | 0 |
| **Вход** | **Изход** |
| -1 | 1 |
| **Вход** | **Изход** |
| -9 | 0 |
| **Вход** | **Изход** |
| 123456789 | 0 |
| **Вход** | **Изход** |
| -1431655766 | 1 |
| **Вход** | **Изход** |
| 4 | 0 |
| **Вход** | **Изход** |
| -2147483648 | 0 |
| **Вход** | **Изход** |
| 2147483647 | 1 |

|  |  |
| --- | --- |
| Задача 05. Лице на трапец | |
| **Условие**   Напишете израз, който изчислява площта на трапец по дадени a, b и h. | |
| **Описание на входа**  Първо като вход се дава едната страна на трапеца, после втората и накрая височината. Всяко от числата е на отделен ред. Също так могат да са дробни и ще са по-големи от 0. Всички входни данни ще бъдат валидни, не е нужно да проверявате изрично. | |
| **Описание на изхода**  Изхода трябва да изписва лицето на трапеца с точност до 2 цифри след десетичната запетая. | |
| **Анализ на задачата**  Четем от конзолата числата и ги записваме в променливи от тип **double**, след което използваме формулата за намиране на лице на трапец и отпечатваме резултата на конзолата с точност до втората цифра след десетичната запетая. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace TrapezoidArea  {  class Program  {  static void Main(string[] args)  {  // Read the measurements of the trapezoid  double baseA = double.Parse(Console.ReadLine());  double baseB = double.Parse(Console.ReadLine());  double height = double.Parse(Console.ReadLine());  // Calculate the trapezoid's area  Console.WriteLine("{0:f2}", (height \* (baseA + baseB) / 2));  }  }  } | |
| **Тестове**   * **Тестове с входни данни, при които лицето се получа по малко от 0.01** * **Тестове с големи числа близки до граници на типа double** * **Тест с дробни числа, с много цифри след десетичната запетая** | |
| **Вход** | **Изход** |
| 1  1  1 | 1,00 |
| **Вход** | **Изход** |
| 123,123  456,456  789,789 | 228872,56 |
| **Вход** | **Изход** |
| 1  9999  0,1 | 500,00 |
| **Вход** | **Изход** |
| 0,1  0,1  0,1 | 0,01 |
| **Вход** | **Изход** |
| 0,1  0,1  0,01 | 0,00 |
| **Вход** | **Изход** |
| 1111,111111111111  7777,777777777777  3333,333333333333 | 14814814,81 |
| **Вход** | **Изход** |
| 1  2  3 | 4,50 |
| **Вход** | **Изход** |
| 9999,99  9999,99  9999,99 | 99999800,00 |
| **Вход** | **Изход** |
| 9999,9999  9999  9999,99 | 99994899,51 |
| **Вход** | **Изход** |
| 10000  10000  10000 | 100000000,00 |

|  |  |
| --- | --- |
| Задача 06. Oбиколка и Лице на правоъгълник | |
| **Условие**  Напишете програма, която за подадени от потребителя дължина и височина на право­ъгълник, пресмята и отпечатва на конзолата неговия периметър и лице. | |
| **Описание на входа**  Първо като вход се дава широчината на правоъгълника и после височината. Всички числа ще бъдат цели положителни, не е нужно да се проверява изрично. | |
| **Описание на изхода**  Изхода трябва да бъде в следния формат:  P = X  S = Y  Като X e периметъра, а Y e лицето на правоъгълника. | |
| **Анализ на задачата**  Четем параметрите на правоъгълника от конзолата и ги записваме в две променливи от типа **uint** . Правим проверка ако широчината или височината е равна на 0, ако да => отпечатваме 0, ако не => отпечатваме сбора на страните. На долния ред отпечатваме лицето на правоъгълника. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace CalculateRectanglePerimeterAndArea  {  class Program  {  static void Main(string[] args)  {  // Read the measurements of the rectangle  uint width = uint.Parse(Console.ReadLine());  uint height = uint.Parse(Console.ReadLine());  // Calculate the rectangle's perimeter and area  Console.WriteLine("P = {0}",  (width == 0 || height == 0) ? 0 : (2 \* width + 2 \* height));  Console.WriteLine("S = {0}", (width \* height));  }  }  } | |
| **Тестове**   * **Тестове с дадена стойност 0** * **Тестове с големи числа.** | |
| **Вход** | **Изход** |
| 1  1 | P = 4  S = 1 |
| **Вход** | **Изход** |
| 10  5 | P = 30  S = 50 |
| **Вход** | **Изход** |
| 0  0 | P = 0  S = 0 |
| **Вход** | **Изход** |
| 100  123 | P = 446  S = 12300 |
| **Вход** | **Изход** |
| 1  0 | P = 0  S = 0 |
| **Вход** | **Изход** |
| 0  1 | P = 0  S = 0 |
| **Вход** | **Изход** |
| 9999  1 | P = 20000  S = 9999 |
| **Вход** | **Изход** |
| 9999  9999 | P = 39996  S = 99980001 |
| **Вход** | **Изход** |
| 10000  9999 | P = 39998  S = 99990000 |
| **Вход** | **Изход** |
| 10000  10000 | P = 40000  S = 100000000 |

|  |  |
| --- | --- |
| Задача 07. Човек на луната | |
| **Условие**  Силата на гравитационното поле на Луната е приблизително 17% от това на Земята. Напишете програма, която да изчислява тежестта на човек на Луната, по дадената тежест на Земята. | |
| **Описание на входа**  Входните данни ще бъдат валидни цели положителни числа. | |
| **Описание на изхода**  Изхода трябва да бъде число с точност до втората цифра след десетичната запетая. | |
| **Анализ на задачата**  Четем от конзолата тежестта и я записваме в променливата от тип **int weight**, след което умножаваме по 0.17 и отпечатваме на конзолата, с точност до две цифри след десетичната запетая. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace ThePowerOfGravity  {  class Program  {  static void Main(string[] args)  {  // Read the input weight  int weight = int.Parse(Console.ReadLine());  // Calculate the weight on the moon  Console.WriteLine("{0:f2}", (weight \* 0.17));  }  }  } | |
| **Тестове**   * **Тестове с близост до горната граница на променливата** * **Тест с 0** | |
| **Вход** | **Изход** |
| 0 | 0,00 |
| **Вход** | **Изход** |
| 1 | 0,17 |
| **Вход** | **Изход** |
| 100 | 17,00 |
| **Вход** | **Изход** |
| 123 | 20,91 |
| **Вход** | **Изход** |
| 9999 | 1699,83 |
| **Вход** | **Изход** |
| 1111111 | 188888,87 |
| **Вход** | **Изход** |
| 17 | 2,89 |
| **Вход** | **Изход** |
| 123456789 | 20987654,13 |
| **Вход** | **Изход** |
| 2000000000 | 340000000,00 |
| **Вход** | **Изход** |
| 2147483647 | 365072219,99 |

|  |  |
| --- | --- |
| Задача 08. Точка в Окръжност | |
| **Условие**  Напишете програма, която проверява дали дадена точка О (x, y) е вътре в окръжността К ((0,0), 5). Пояснение: точката (0,0) е център на окръжността, а радиусът й е 5. | |
| **Описание на входа**  Входа е винаги валидно цяло число.  На първия ред се дава координатата **x,** на втория ред съответно координатата **y** | |
| **Описание на изхода**  Изхода трябва да връща съответно „within“ или „out“ | |
| **Анализ на задачата**  Четем от конзолата съответните кординати и използваме Питагоровата теорема, за да проверим дали точката е в окръжността. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace CheckIfAPointIsWithinACircle  {  class Program  {  static void Main(string[] args)  {  // Read the point's coordinates  int x = int.Parse(Console.ReadLine());  int y = int.Parse(Console.ReadLine());  int radius = 5;  // Check if the point is within or out of the circle  if ((x \* x) + (y \* y) > (radius \* radius))  {  Console.WriteLine("out");  }  else  {  Console.WriteLine("within");  }  }  }  } | |
| **Тестове**   * **Тестове с координати на границата на кръга** * **Тестове с координати близки до границата на кръга** * **Тестове с 0** * **Тестове с отрицателни координати** * **Тест с много далечни координати** | |
| **Вход** | **Изход** |
| 0  0 | within |
| **Вход** | **Изход** |
| 5  5 | out |
| **Вход** | **Изход** |
| 0  5 | within |
| **Вход** | **Изход** |
| -5  0 | within |
| **Вход** | **Изход** |
| 3  4 | within |
| **Вход** | **Изход** |
| 10000  -10000 | out |
| **Вход** | **Изход** |
| 1  1 | within |
| **Вход** | **Изход** |
| 4  4 | out |
| **Вход** | **Изход** |
| -1  -5 | out |
| **Вход** | **Изход** |
| -1  -4 | within |

|  |  |
| --- | --- |
| Задача 09. Точка в Правоъгълник | |
| **Условие**  Напишете програма, която проверява дали дадена точка О (x, y) е вътре в окръжността К ((0,0), 5) и едновременно с това извън право­ъгълника ((-1, 1), (5, 5). Пояснение: правоъгълникът е зададен чрез координатите на горния си ляв и долния си десен ъгъл. | |
| **Описание на входа**  Входните данни ще бъдат винаги валидни цели числа. | |
| **Описание на изхода**  Изхода трябва да бъде „True“, ако точката е в окръжността и извън правоъгълника, в противен случай „False“. | |
| **Анализ на задачата**  Вземаме кординатите на точката и проверяваме дали точката е в кръга, после правим и проверка дали е извън правоъгълника. Ако отговаря на и двете условия отпечатваме „True“, иначе „False“. | |
| **Решение (сорс код)**  using System;  using System.Linq;  namespace CheckIfAPointIsWithinACircleAndOutOfARectangle  {  class Program  {  static void Main(string[] args)  {  int pointX = int.Parse(Console.ReadLine());  int pointY = int.Parse(Console.ReadLine());    int radius = 5;  bool isInTheCircle = false;  if ((radius \* radius) >= ((pointX \* pointX) + (pointY \* pointY)))  {  isInTheCircle = true;  }  // Set the rectangle's coordinates  int rectBotLeftX = -1;  int rectBotLeftY = 1;  int rectTopRigthX = 5;  int rectTopRigthY = 5;  // Check if the point is out of the rectangle  bool isOutOfTheRect = false;  if ((rectBotLeftX > pointX) || (rectTopRigthX < pointX) ||  (rectBotLeftY > pointY) || (rectTopRigthY < pointY))  {  isOutOfTheRect = true;  }    Console.WriteLine(isInTheCircle && isOutOfTheRect);  }  }  } | |
| **Тестове**   * **Тестове за граничните случаи между правоъгълника и кръга** * **Тестове за точка в кръга и в правоъгълника** * **Тест с 0** * **Тестове с отрицателни координати** | |
| **Вход** | **Изход** |
| 0  0 | True |
| **Вход** | **Изход** |
| 5  5 | False |
| **Вход** | **Изход** |
| -5  5 | False |
| **Вход** | **Изход** |
| -5  0 | True |
| **Вход** | **Изход** |
| 3  4 | False |
| **Вход** | **Изход** |
| 5  1 | False |
| **Вход** | **Изход** |
| 1  1 | False |
| **Вход** | **Изход** |
| -3  4 | True |
| **Вход** | **Изход** |
| -1  -5 | False |
| **Вход** | **Изход** |
| -1  1 | False |

|  |  |
| --- | --- |
| Задача 10. Swaping Digits in A Number | |
| **Условие**  Напишете програма, която приема за вход четирицифрено число във формат *abcd*  (например числото 2011) и след това извършва следните действия върху него:  -     Пресмята сбора от цифрите на числото (за нашия пример 2+0+1+1 = 4).  -     Разпечатва на конзолата цифрите в обратен ред: *dcba* (за нашия пример резултатът е 1102).  -     Поставя последната цифра, на първо място: *dabc* (за нашия пример резултатът е 1201).  -     Разменя мястото на втората и третата цифра: *acbd* (за нашия пример резултатът е 2101). | |
| **Описание на входа**  Входа винаги ще бъде валидно цяло четирицифрено число. | |
| **Описание на изхода**  Изхода на първия ред трябва да изкара сбора на цифрите на числото.  На втория ред разпечатва в обратен ред.  Поставя последната цифра на първо място.  Разменя втората и трета цифра. | |
| **Анализ на задачата**  Четем числото от кознолата записваме го в една променлива. Раздробяваме променлива в четири по малки където пазим цифрите на числото и съответна с тях извършваме манипулациите. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace SwappingDigitsInANumber  {  class Program  {  static void Main(string[] args)  {  int abcd = int.Parse(Console.ReadLine());  // Extract the digits  int a = abcd / 1000;  int b = (abcd % 1000) / 100;  int c = (abcd % 100) / 10;  int d = abcd % 10;  Console.WriteLine(a + b + c + d);  Console.WriteLine(d \* 1000 + c \* 100 + b \* 10 + a);  Console.WriteLine(d \* 1000 + a \* 100 + b \* 10 + c);  Console.WriteLine(a \* 1000 + c \* 100 + b \* 10 + d);  }  }  } | |
| **Тестове**   * **Тестове с числа, които съдържат 0** * **Тест с число с еднакви цифри** * **Тест с огледално число** | |
| **Вход** | **Изход** |
| 2011 | 4  1102  1201  2101 |
| **Вход** | **Изход** |
| 1000 | 1  1  100  1000 |
| **Вход** | **Изход** |
| 1234 | 10  4321  4123  1324 |
| **Вход** | **Изход** |
| 1111 | 4  1111  1111  1111 |
| **Вход** | **Изход** |
| 1001 | 2  1001  1100  1001 |
| **Вход** | **Изход** |
| 1209 | 12  9021  9120  1029 |
| **Вход** | **Изход** |
| 3553 | 16  3553  3355  3553 |
| **Вход** | **Изход** |
| 1200 | 3  21  120  1020 |
| **Вход** | **Изход** |
| 9876 | 30  6789  6987  9786 |
| **Вход** | **Изход** |
| 5080 | 13  805  508  5800 |

|  |  |
| --- | --- |
| Задача 11. Get A Bit From a Number | |
| **Условие**  Дадено е число **n** и позиция **p**. Напишете поредица от операции, които да отпечатат стойността на бита на позиция **p** от числото **n** (0 или 1). Пример: n=35, p=5 -> 1. Още един пример: n=35, p=6 -> 0. | |
| **Описание на входа**  На първия ред ще бъде дадено числото, а на втория ред позицията. Входните данни ще бъдат валидни цели числа. | |
| **Описание на изхода**  На изхода съответно трябва да бъде отпечато бита, който отговаря на тази позиция. | |
| **Анализ на задачата**  Вземаме и записваме входното число в **InputNumber,** а след това и позицията на бита, който искаме да извлечем, в **bitPosition**. Изместваме единственият бит на единицата на ляво с **bitPosition** позиции и така се получава число с единствен бит 1 на позиция **bitPosition.** След което използваме логическото „и“, за да определим бита на числото **inputNumber** на позиция **bitPosition**. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace GetABitFromANumber  {  class Program  {  static void Main(string[] args)  {  // Read the input number and position  int inputNumber = int.Parse(Console.ReadLine());  int bitPosition = int.Parse(Console.ReadLine());  // Find the bit at the given position  Console.WriteLine(0 == (inputNumber & (1 << bitPosition)) ? 0 : 1);  }  }  } | |
| **Тестове**   * **Тестове с 0** * **Тестове с отрицателно число** * **Тестове с 32вия бит (този за знак)** | |
| **Вход** | **Изход** |
| 1  0 | 1 |
| **Вход** | **Изход** |
| 1  1 | 0 |
| **Вход** | **Изход** |
| -1  10 | 1 |
| **Вход** | **Изход** |
| -111  31 | 1 |
| **Вход** | **Изход** |
| 32768  15 | 1 |
| **Вход** | **Изход** |
| 123456789  28 | 0 |
| **Вход** | **Изход** |
| -123123123  15 | 0 |
| **Вход** | **Изход** |
| 1431655765  16 | 1 |
| **Вход** | **Изход** |
| -2147483648  13 | 0 |
| **Вход** | **Изход** |
| 2147483647  31 | 0 |

|  |  |
| --- | --- |
| Задача 12. Проверка дали даден бит е 1 | |
| **Условие**  Напишете булев израз, който проверява дали битът на позиция **p** на цялото число **v** има стойност 1. Пример v=5, p=1 -> false. | |
| **Описание на входа**  Входа е винаги валидно цяло число, като първо се дава числото, а след това позицията. | |
| **Описание на изхода**  Изхода трябва да бъде „True “, ако бита на дадената позиция е 1, във всички останали случаи връща „False“ | |
| **Анализ на задачата**  Числото се записва в **inputInteger**, а позицията в **bitPosition**. Изместваме единственият бит на единицата на ляво с **bitPosition** позиции и така се получава число с единствен бит 1 на позиция **bitPosition.** След което използваме логическото „и“, за да определим дали бита на числото **inputInteger** на позиция **bitPosition** е 1. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace CheckIfAGivenBitIs1  {  class Program  {  static void Main(string[] args)  {  int inputInteger = int.Parse(Console.ReadLine());  int bitPosition = int.Parse(Console.ReadLine());  // Check if the bit at the given position is 1  Console.WriteLine(0 != (inputInteger & (1 << bitPosition)));  }  }  } | |
| **Тестове**   * **Тестове с 0** * **Тестове с отрицателно число** * **Тестове с 32вия бит (този за знак)** * **Тестове с горната и долната граница на типа Int** | |
| **Вход** | **Изход** |
| 1  0 | True |
| **Вход** | **Изход** |
| 1  1 | False |
| **Вход** | **Изход** |
| -1  10 | True |
| **Вход** | **Изход** |
| -111  31 | True |
| **Вход** | **Изход** |
| 32768  15 | True |
| **Вход** | **Изход** |
| 123456789  28 | False |
| **Вход** | **Изход** |
| -123123123  15 | False |
| **Вход** | **Изход** |
| 1431655765  16 | True |
| **Вход** | **Изход** |
| -2147483648  13 | False |
| **Вход** | **Изход** |
| 2147483647  31 | False |

|  |  |
| --- | --- |
| Задача 13. Смяна на даден бит | |
| **Условие**  Дадено е число n, стойност **v** (**v** = 0 или 1) и позиция **p**. Напишете поредица от операции, които да променят стойността на n, така че битът на позиция p да има стойност v. Пример n=35, p=5, v=0 -> n=3. Още един пример: n=35, p=2, v=1 -> n=39. | |
| **Описание на входа**  Входа ще бъде винаги валидно цяло число като на първия ред ще е числото, а на втория бита, който трябва да променим, и на третия ред - позицията. | |
| **Описание на изхода**  Трябва да отпечатваме числото след промяната на дадения бит. | |
| **Анализ на задачата**  Четем дадените параметри от задачата правим проверка дали бита на тази позиция съвпада с даденото ни число. Ако съвпада - не го променяме и отпечатваме на конзолата. Ако не съвпада - го променяме и отпечатваме на конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace ChangeTheBitAtGivenPosition  {  class Program  {  static void Main(string[] args)  {  int input = int.Parse(Console.ReadLine());  int value = int.Parse(Console.ReadLine());  int bitPosition = int.Parse(Console.ReadLine());  // Change the bit at the given position  if (0 == value)  {  value = ~(1 << bitPosition);  value = value & input;  }  else  {  value = 1 << bitPosition;  value = value | input;  }  Console.WriteLine(value);  }  }  } | |
| **Тестове**   * **Тестове с отрицателни числа** * **Тестове с 32вия бит (този за знак)** * **Тестове с горната и долната граница на типа Int** | |
| **Вход** | **Изход** |
| 1  0  0 | 0 |
| **Вход** | **Изход** |
| 1  0  1 | 1 |
| **Вход** | **Изход** |
| -1  0  31 | 2147483647 |
| **Вход** | **Изход** |
| 35  0  5 | 3 |
| **Вход** | **Изход** |
| 35  1  2 | 39 |
| **Вход** | **Изход** |
| 128  0  7 | 0 |
| **Вход** | **Изход** |
| -123123123  1  15 | -123090355 |
| **Вход** | **Изход** |
| 1431655765  0  30 | 357913941 |
| **Вход** | **Изход** |
| -2147483648  1  15 | -2147450880 |
| **Вход** | **Изход** |
| 2147483647  1  31 | -1 |

|  |  |
| --- | --- |
| Задача 14. Prime Numbers | |
| **Условие**  Напишете програма, която проверява дали дадено число **n** (**1 < n < 100**) е просто (т.е. се дели без остатък само на себе си и на единица). | |
| **Описание на входа**  Входа винаги ще бъде валидно цяло число между 1 и 100 (Като 1 и 100 не се включват) | |
| **Описание на изхода**  Изхода трябва да бъде съответно „prime“, ако числото е просто или „not prime“ в противен случай. | |
| **Анализ на задачата**  Кои са простите числа между 1 и 10? (2,3,5,7) Всяко число по-голямо от 10 и по-малко от 100, което не се дели на тези числа е просто. Правим проверка дали даденото число не съвпада с някое от тези, ако съвпада => съответно отпечатваме „prime“, ако числото не е едно от тях правим втора проверка, където проверяваме дали се дели на някое от тези числа без остатък (2,3,5,7). Ако се дели => отпечатваме „not prime“, в противен случай числото е просто („prime“). | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace CheckIfANumberIsPrime  {  class Program  {  static void Main(string[] args)  {  int input = int.Parse(Console.ReadLine());  if (2 == input || 3 == input || 5 == input || 7 == input)  {  Console.WriteLine("prime");  }  else  {  if ((0 != (input % 2)) && (0 != (input % 3)) && (0 != (input % 5)) && (0 != (input % 7)))  {  Console.WriteLine("prime");  }  else  {  Console.WriteLine("not prime");  }  }  }  }  } | |
| **Тестове**   * **Тестовете са с различни числа от 1 до 100** * **Тестове с числа близки до 1 или 100** | |
| **Вход** | **Изход** |
| 2 | prime |
| **Вход** | **Изход** |
| 49 | not prime |
| **Вход** | **Изход** |
| 13 | prime |
| **Вход** | **Изход** |
| 53 | prime |
| **Вход** | **Изход** |
| 55 | not prime |
| **Вход** | **Изход** |
| 67 | prime |
| **Вход** | **Изход** |
| 73 | prime |
| **Вход** | **Изход** |
| 77 | not prime |
| **Вход** | **Изход** |
| 97 | prime |
| **Вход** | **Изход** |
| 99 | not prime |

|  |  |
| --- | --- |
| Задача 15. Bit Exchange (3->24, 4->25, 5->26) | |
| **Условие**  Напишете програма, която разменя стойностите на битовете на позиции 3, 4 и 5 с битовете на позиции 24, 25 и 26 на дадено цяло положително число. | |
| **Описание на входа**  Входа ще бъде цяло положително число от тип int. | |
| **Описание на изхода**  Изхода трябва да изпише на конзолата числото след промяната в битовете. | |
| **Анализ на задачата**  Четем от конзолата числото, чийто битове ще разменяме и го записваме в **input**. Дефинираме две числа **firstSeqBits** и **secondSeqBits**, които първоначално са равни на 0. След което чрез побитови операции копираме нужните битове от input числото съответно в **firstSeqBits** и **secondSeqBits.** След това изместваме битовете на **firstSeqBits** и **secondSeqBits** снеобходимите позиции и копираме техните битове обратно в **input**. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace ExchangeThreeBitsInANumber  {  class Program  {  static void Main(string[] args)  {    int input = int.Parse(Console.ReadLine());  // Set starting positions  int startExchangePosition = 3;  int exchangeBits = 3;  int exchangeFromPosition = 24;  // Get bits from the number  int firstSeqBits = 0;  int secondSeqBits = 0;  for (int i = 0; i < exchangeBits; i++)  {  if (0 != (input & (1 << (startExchangePosition + i))))  {  firstSeqBits |= (1 << i);  }  input &= ~(1 << startExchangePosition + i);  if (0 != (input & (1 << (exchangeFromPosition + i))))  {  secondSeqBits |= (1 << i);  }  input &= ~(1 << exchangeFromPosition + i);  }  // Exchange bits  input |= secondSeqBits << startExchangePosition;  input |= firstSeqBits << exchangeFromPosition;  // Print the result  Console.WriteLine(input);  }  }  } | |
| **Тестове**   * **Тестове с близост до границата на типа int.** * **Тестове с отрицателни числа** * **Тестове с числа, които имат еднакви битове на съответните позиции** | |
| **Вход** | **Изход** |
| 1 | 1 |
| **Вход** | **Изход** |
| 56 | 117440512 |
| **Вход** | **Изход** |
| 8 | 16777216 |
| **Вход** | **Изход** |
| -9 | -16777217 |
| **Вход** | **Изход** |
| -1431655766 | -1381324142 |
| **Вход** | **Изход** |
| 123456789 | 39570749 |
| **Вход** | **Изход** |
| 117440568 | 117440568 |
| **Вход** | **Изход** |
| 67108872 | 16777248 |
| **Вход** | **Изход** |
| -2122219135 | -2138996343 |
| **Вход** | **Изход** |
| 32 | 67108864 |

|  |  |
| --- | --- |
| Задача 16. Bit Exchange | |
| **Условие**   Напишете програма, която разменя битовете на позиции {p, p+1, …, p+k-1) с битовете на позиции {q, q+1, …, q+k-1} на дадено цяло положително число. | |
| **Описание на входа**  На първия ред ще ви е дадено числото, на което трябва да се сменят битовете.  На втория ред позицията, от която да взима битовете.  На третия ред колко бита да вземе.  На четвъртия от коя позиция да започне да сменя, като всички входни данни са валидни цели числа. | |
| **Описание на изхода**  Изхода трябва да бъде числото след промяната . | |
| **Анализ на задачата**  Четем от конзолата числото, чийто битове ще разменяме и го записваме в **input**. Останалите данни ги записваме в **startExchangePosition**, **exchangeBits**, **exchangeFromPosition**. Дефинираме две числа **firstSeqBits** и **secondSeqBits**, които първоначално са равни на 0. След което чрез побитови операции копираме нужните битове от input числото съответно в **firstSeqBits** и **secondSeqBits.** След това изместваме битовете на **firstSeqBits** и **secondSeqBits** снеобходимите позиции и копираме техните битове обратно в **input**. | |
| **Решение (сорс код)** | |
| using System;  using System.Linq;  namespace ExchangeThreeBitsInANumber  {  class Program  {  static void Main(string[] args)  {  int input = int.Parse(Console.ReadLine());  int startExchangePosition = int.Parse(Console.ReadLine());  int exchangeBits = int.Parse(Console.ReadLine());  int exchangeFromPosition = int.Parse(Console.ReadLine());  // Get bits from the number  int firstSeqBits = 0;  int secondSeqBits = 0;  for (int i = 0; i < exchangeBits; i++)  {  if (0 != (input & (1 << (startExchangePosition + i))))  {  firstSeqBits |= (1 << i);  }  input &= ~(1 << startExchangePosition + i);  if(0 != (input & (1 << (exchangeFromPosition + i))) )  {  secondSeqBits |= (1 << i);  }    input &= ~(1 << exchangeFromPosition + i);  }  // Exchange bits  input |= secondSeqBits << startExchangePosition;  input |= firstSeqBits << exchangeFromPosition;  // Print the result  Console.WriteLine(input);  }  }  } | |
| **Тестове**   * **Тестове с близост до границата на типа int.** * **Тестове с отрицателни числа** * **Тестове с числа, които имат еднакви битове на съответните позиции** | |
| **Вход** | **Изход** |
| 1  0  1  1 | 2 |
| **Вход** | **Изход** |
| 1  1  1  10 | 1 |
| **Вход** | **Изход** |
| 8  2  1  3 | 4 |
| **Вход** | **Изход** |
| -123123123  0  4  28 | -659994033 |
| **Вход** | **Изход** |
| -1431655766  1  10  15 | -1431655766 |
| **Вход** | **Изход** |
| 123456789  1  6  21 | 22793589 |
| **Вход** | **Изход** |
| 1  0  1  31 | -2147483648 |
| **Вход** | **Изход** |
| 65535  0  16  16 | -65536 |
| **Вход** | **Изход** |
| -2122219135  4  12  16 | -2011686895 |
| **Вход** | **Изход** |
| 32  3  3  24 | 67108864 |