# 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 задачите, описание на използва­ните идеи, алгоритми, подходи за решаване и тестове.

## Авторски колектив

**Вася Георгиева Станкова**

Контакти:

* e-mail: [vasya.stankova@gmail.com](mailto:vasya.stankova@gmail.com),
* LinkedIn профил: <http://www.linkedin.com/pub/vasya-stankova/30/b11/3b>

**Димитър Бакалов**

Контакти:

* e-mail: [dpbakalov@abv.bg](mailto:dpbakalov@abv.bg)

**Данаил Василев**

Контакти:

* e-mail: [massaro@mail.bg](mailto:massaro@mail.bg)

|  |  |  |
| --- | --- | --- |
| Задача 1. Четно или Нечетно | | |
| **Условие**  Напишете израз, който да проверява дали дадено цяло число е четно или нечетно.  Пример: **2** 🡪 **True**  **3** 🡪 **False** | | |
| **Описание на входа**  Входните данни се въвеждат като цяло число, на един единствен ред. | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако числото е четно се извежда **True**, в противен случай – **False**. | | |
| **Анализ на задачата**  Вземете остатъкът от деленето на числото на 2 и проверете дали е 0 или 1 (четно, нечетно). Използвайте оператора % за пресмятане на остатък от целочислено деление. | | |
| **Решение (сорс код)** | | |
| using System;  class EvenOrNot  {  static void Main()  {  int number = int.Parse(Console.ReadLine());  bool isEven = number % 2 == 0;  Console.WriteLine(isEven);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * int.MinValue (-2147483648) * int.MaxValue (2147483647) * 0 * -1 * -2 * -2147483647 * 1 * 2147483646 | | |
| **Вход** | **Изход** | |
| 2 | True | |
| **Вход** | **Изход** | |
| 3 | False | |
| **Вход** | **Изход** | |
| -2147483648 | True | |
| **Вход** | **Изход** | |
| 2147483647 | False | |
| **Вход** | **Изход** | |
| 0 | True | |
| **Вход** | **Изход** | |
| -1 | False | |
| **Вход** | **Изход** | |
| -2 | True | |
| **Вход** | **Изход** | |
| -2147483647 | False | |
| **Вход** | **Изход** | |
| 1 | False | |
| **Вход** | **Изход** | |
| 100000 | True | |
| **Вход** | **Изход** | |
| 2147483646 | True | |
| Задача 2. Делимо на 5 и 7 | | |
| **Условие**  Напишете булев израз, който да проверява дали дадено цяло число се дели на 5 и на 7 без остатък.  Пример: **1225** 🡪 **True**  **2000** 🡪 **False** | | |
| **Описание на входа**  Входните данни се въвеждат като цяло число, на един единствен ред. | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако числото е делимо на 5 и на 7 едновременно, се извежда **True**, в противен случай – **False**. | | |
| **Анализ на задачата**  Ползвайте логическо "И" и операцията % за остатък при деление. Можете да решите задачата и чрез само една проверка – за деление на 35 (помислете защо). | | |
| **Решение (сорс код)** | | |
| using System;  class IsDivisibleBy5and7  {  static void Main()  {  int number = int.Parse(Console.ReadLine());  bool isDivisibleBy5and7 = number % 35 == 0;  Console.WriteLine(isDivisibleBy5and7);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 0 * -5 * -7 * -35 * -2147483625 * -2147483647 | | |
| **Вход** | **Изход** | |
| 1225 | True | |
| **Вход** | **Изход** | |
| 2000 | False | |
| **Вход** | **Изход** | |
| 0 | True | |
| **Вход** | **Изход** | |
| -5 | False | |
| **Вход** | **Изход** | |
| -7 | False | |
| **Вход** | **Изход** | |
| -35 | True | |
| **Вход** | **Изход** | |
| -2147483625 | True | |
| **Вход** | **Изход** | |
| -2147483647 | False | |
| **Вход** | **Изход** | |
| 5 | False | |
| **Вход** | **Изход** | |
| 7 | False | |
| **Вход** | **Изход** | |
| 35 | True | |
| **Вход** | **Изход** | |
| 2147483625 | True | |
| **Вход** | **Изход** | |
| 2147483647 | False | |
| Задача 3. Третата цифра | | |
| **Условие**  Напишете израз, който да проверява дали дадено цяло число съдържа 7 за трета цифра (отдясно на ляво).  Пример: **1766** 🡪 **True**  **1677** 🡪 **False** | | |
| **Описание на входа**  Входните данни се въвеждат като цяло число, на един единствен ред. | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако даденото цяло число съдържа 7 за трета цифра (отдясно на ляво), се извежда **True**, в протиквен случай – **False**. | | |
| **Анализ на задачата**  Разделете числото на 100 и го запишете в нова променлива. Нея разделете на 10 и вземете остатъкът. Остатъкът от делението на 10 е третата цифра от първоначалното число. Проверете равна ли е на 7. | | |
| **Решение (сорс код)** | | |
| using System;  class ThirdDigit  {  static void Main()  {  int number = int.Parse(Console.ReadLine());  int numberDividedByHundred = number / 100;  int thirdDigit = Math.Abs(numberDividedByHundred % 10);  bool isThirdDigit7 = thirdDigit == 7;  Console.WriteLine(isThirdDigit7);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 0 * -700 * -2147482748 * 2147483647 | | |
| **Вход** | **Изход** | |
| 1766 | True | |
| **Вход** | **Изход** | |
| 1677 | False | |
| **Вход** | **Изход** | |
| 0 | False | |
| **Вход** | **Изход** | |
| -700 | True | |
| **Вход** | **Изход** | |
| 700 | True | |
| **Вход** | **Изход** | |
| -2147483648 | False | |
| **Вход** | **Изход** | |
| -2147482748 | True | |
| **Вход** | **Изход** | |
| 2147483647 | False | |
| **Вход** | **Изход** | |
| 2147482747 | True | |
| **Вход** | **Изход** | |
| 1777777777 | True | |
| Задача 4. Третият бит | | |
| **Условие**  Напишете израз, който да проверява дали третия бит на дадено число е 1 или 0. | | |
| **Описание на входа**  Входните данни се въвеждат като цяло число, на един единствен ред. | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако даденото цяло число има 1 в третия си бит, се извежда **True**, в противен случай – **False**. | | |
| **Анализ на задачата**  Използвайте побитово "И" върху числото и число, което има 1 само в третия си бит (т.е. числото 8, ако броенето на битовете започне от 0). Ако върнатият резултат е различен от 0, то третия бит е 1. | | |
| **Решение (сорс код)** | | |
| using System;  class ThirdBit  {  static void Main()  {  int number = 3;  int mask = 1 << number;  int integer = int.Parse(Console.ReadLine());  int result = integer & mask;  bool isThirdBit1 = result > 0;  Console.WriteLine(isThirdBit1);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 8 * 0 * 1 * 64 * 1111 | | |
| **Вход** | **Изход** | |
| 1111 | False | |
| **Вход** | **Изход** | |
| 8 | True | |
| **Вход** | **Изход** | |
| 0 | False | |
| **Вход** | **Изход** | |
| 1 | False | |
| **Вход** | **Изход** | |
| 64 | False | |
| **Вход** | **Изход** | |
| 255 | True | |
| Задача 5. Площ на трапец | | |
| **Условие**  Напишете израз, който изчислява площта на трапец по дадени a, b и h. | | |
| **Описание на входа**  Входните данни се въвеждат като цeли или реални числа на три реда. На първия ред се въвежда стойността на **а**, на втория - стойността на **b** и на третия - стойността на **h**. | | |
| **Описание на изхода**  Изходът се извежда като реално число. На един ред се извежда площта на трапеца. | | |
| **Анализ на задачата**  Формулата за лице на трапец: S = (a + b) / 2 \* h. | | |
| **Решение (сорс код)** | | |
| using System;  class TrapezoidArea  {  static void Main()  {  float a = float.Parse(Console.ReadLine());  float b = float.Parse(Console.ReadLine());  float h = float.Parse(Console.ReadLine());  float area = (a + b) / 2 \* h;  Console.WriteLine(area);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 4\8\10 * 10\4\2 * 4\4\4 * 8\8\4 | | |
| **Вход** | **Изход** | |
| 4  8  10 | 60 | |
| **Вход** | **Изход** | |
| 10  4  2 | 14 | |
| **Вход** | **Изход** | |
| 4  4  4 | 16 | |
| **Вход** | **Изход** | |
| 8  8  4 | 32 | |
| **Вход** | **Изход** | |
| 999999,14  555555,23  1 | 777777,2 | |
| **Вход** | **Изход** | |
| 1111  22222  333 | 3884945 | |
| **Вход** | **Изход** | |
| 1,234  5,678  9,123 | 31,52909 | |
| **Вход** | **Изход** | |
| 5678  7891  14 | 94983 | |
| **Вход** | **Изход** | |
| 1  1  1 | 1 | |
| **Вход** | **Изход** | |
| 10  10  10 | 100 | |
| Задача 6. Периметър 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**  Входните данни се въвеждат като цели числа на два реда. На първия ред се въвежда стойността на **а**, на втория - стойността на **b**. | | |
| **Описание на изхода**  Изходът се извежда като цяло число. На първия ред се извежда стойността на **периметъра**, на втория - стойността на **площта**. | | |
| **Анализ на задачата**  Потърсете в Интернет как се въвеждат цели числа от конзолата и използвайте формулата за лице на правоъгълник. Ако се затруднявате погледнете упътването на следващата задача. | | |
| **Решение (сорс код)** | | |
| using System;  class RectanglePerimeterAndArea  {  static void Main()  {  int a = int.Parse(Console.ReadLine());  int b = int.Parse(Console.ReadLine());  int perimeter = (a + b) \* 2;  Console.WriteLine(perimeter);  int area = a \* b;  Console.WriteLine(area);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 4\6 * 9876\54321 * 77777\77777 | | |
| **Вход** | **Изход** | |
| 4  6 | 20  24 | |
| **Вход** | **Изход** | |
| 9876  54321 | 128394  536474196 | |
| **Вход** | **Изход** | |
| 54321  189 | 109020  10266669 | |
| **Вход** | **Изход** | |
| 0  0 | 0  0 | |
| **Вход** | **Изход** | |
| 46340  46340 | 185360  2147395600 | |
| **Вход** | **Изход** | |
| 109  12345 | 24908  1345605 | |
| **Вход** | **Изход** | |
| 20  24 | 88  480 | |
| **Вход** | **Изход** | |
| 2147483647  2147483647 | -4  1 | |
| **Вход** | **Изход** | |
| 0  0 | 0  0 | |
| **Вход** | **Изход** | |
| 2147483647  1 | 0  2147483647 | |
| Задача 7. Тегло на 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 се въвеждат като цяло число. На един единствен ред се въвежда теглото на човек на Земята. | | |
| **Описание на изхода**  Изходът се извежда като реално число с двойна точност. | | |
| **Анализ на задачата**  Използвайте следния код, за да прочетете число от конзолата, след което го умножете по 0.17 и го отпечатайте:   |  | | --- | | int number = Convert.ToInt32(Console.ReadLine(); | | | |
| **Решение (сорс код)** | | |
| using System;  class WeightOnTheMoon  {  static void Main()  {  int number = int.Parse(Console.ReadLine());  double weightOnTheMoon = number \* 0.17;  Console.WriteLine(weightOnTheMoon);  }  } | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 57 * 100 * 111 * 71 * 0 | | |
| **Вход** | **Изход** | |
| 57 | 9,69 | |
| **Вход** | **Изход** | |
| 100 | 17 | |
| **Вход** | **Изход** | |
| 111 | 18,87 | |
| **Вход** | **Изход** | |
| 71 | 12,07 | |
| **Вход** | **Изход** | |
| 0 | 0 | |
| **Вход** | **Изход** | |
| 222 | 37,74 | |
| **Вход** | **Изход** | |
| 2147483647 | 365072219.99 | |
| **Вход** | **Изход** | |
| 1 | 0.17 | |
| **Вход** | **Изход** | |
| 999999999 | 169999999.83 | |
| Задача 8. Точка вътре 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**  Входните данни се въвеждат като цeли числа – на два реда. На първия ред се въвежда стойността на координатата **x**, на втория - стойността на координатата **y**. | | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако точкота е вътре в окръжността, се извежда **True**, в противен случай – **False**. | | | |
| **Анализ на задачата**  Използвайте питагоровата теорема a2 + b2 = c2. За да е точката вътре в кръга, то x\*x + y\*y следва да е по-малко или равно на 5. | | | |
| **Решение (сорс код)** | | | |
| using System;  class IsPointInCircle  {  static void Main()  {  int x = int.Parse(Console.ReadLine());  int y = int.Parse(Console.ReadLine());  bool isPointInCircle = Math.Sqrt(x \* x + y \* y) <= 5;  Console.WriteLine(isPointInCircle);  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 2\3 * -2\-3 * 2\-3 * 5\1 * -5\1 | | | |
| **Вход** | | **Изход** | |
| 2  3 | | True | |
| **Вход** | | **Изход** | |
| -2  -3 | | True | |
| **Вход** | | **Изход** | |
| 2  -3 | | True | |
| **Вход** | | **Изход** | |
| 5  1 | | False | |
| **Вход** | | **Изход** | |
| 1  5 | | False | |
| **Вход** | | **Изход** | |
| -1  5 | | False | |
| **Вход** | | **Изход** | |
| 0  111 | | False | |
| **Вход** | | **Изход** | |
| 0  5 | | True | |
| **Вход** | | **Изход** | |
| 5  0 | | True | |
| **Вход** | | **Изход** | |
| 4  0 | | True | |
| Задача 9. Точка вътре в окръжност и извън 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 се въвеждат цяло число. На първия ред се въвежда стойността на координатата **x** на точката О, на втория - стойността на координатата **y**. | | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако точката е вътре в окръжността и едновременно с това, извън правоъгълника се извежда **True**, в противен случай – **False**. | | | |
| **Анализ на задачата**  Използвайте кода от предходната задача и добавете проверка за правоъгълника. Една точка е вътре в даден правоъгълник със стени успоредни на координатните оси, когато е вдясно от лявата му стена, вляво от дясната му стена, надолу от горната му стена и нагоре от долната му стена. | | | |
| **Решение (сорс код)** | | | |
| using System;  class IsPointInCircleAndOutsideRectangle  {  static void Main()  {  int x = int.Parse(Console.ReadLine());  int y = int.Parse(Console.ReadLine());  bool isInCircle = Math.Sqrt(x \* x + y \* y) <= 5;  bool isOutsideRectangle = x < -1 || x > 5 || y < 1 || y > 5;  bool isInCircleAndOutsideRectangle = isInCircle && isOutsideRectangle;  Console.WriteLine(isInCircleAndOutsideRectangle);  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 2\3 * 5\4 * -1\1 * -2\-3 | | | |
| **Вход** | | **Изход** | |
| 2  3 | | False | |
| **Вход** | | **Изход** | |
| 5  4 | | False | |
| **Вход** | | **Изход** | |
| -1  1 | | False | |
| **Вход** | | **Изход** | |
| -2  -3 | | True | |
| **Вход** | | **Изход** | |
| -2  -5 | | False | |
| **Вход** | | **Изход** | |
| 1  0 | | True | |
| **Вход** | | **Изход** | |
| 0  1 | | False | |
| **Вход** | | **Изход** | |
| -1  5 | | False | |
| **Вход** | | **Изход** | |
| 5  5 | | False | |
| **Вход** | | **Изход** | |
| 6  6 | | False | |
| **Вход** | | **Изход** | |
| 1  1 | | False | |
| **Вход** | | **Изход** | |
| 1  2 | | False | |
| Задача 10. Манипулации четирицифрено число | | | |
| **Условие**  Напишете програма, която приема за вход четирицифрено число във формат *abcd* (например числото 2010) и след това извършва следните действия върху него:   * + Пресмята сбора от цифрите на числото (за нашия пример 2+0+1+0 = 3).   + Разпечатва на конзолата цифрите в обратен ред: *dcba* (за нашия пример резултатът е 0102).   + Поставя последната цифра, на първо място: *dabc* (за нашия пример резултатът е 0201).   + Разменя мястото на втората и третата цифра: *acbd* (за нашия пример резултатът е 2100). | | | |
| **Описание на входа**  Входните данни се въвеждат като цяло число на един ред. На един единствен ред се въвежда четирицифрено число. | | | |
| **Описание на изхода**  На отделни редове се разпечатват сбора от цифрите на числото, цифрите в обратен ред, поставя се последната цифра на първо място и се разпечатва числото, разменя се мястото на втората и третата цифра и се разпечатва числото. | | | |
| **Анализ на задачата**  За да вземете отделните цифри на числото, можете да го делите на 10 и да взимате остатъка при деление на 10 последователно 4 пъти. | | | |
| **Решение (сорс код)** | | | |
| using System;  class FourDigitManipulation  {  static void Main()  {  int abcd = int.Parse(Console.ReadLine());  int a = abcd / 1000;  int b = (abcd / 100) % 10;  int c = (abcd % 100) / 10;  int d = abcd % 10;  int sum = a + b + c + d;  Console.WriteLine(sum);  int reversed = 1000 \* d + 100 \* c + 10 \* b + a;  Console.WriteLine(reversed.ToString("D4"));  int firstLastDigitChanged = 1000 \* d + 100 \* b + 10 \* c + a;  Console.WriteLine(firstLastDigitChanged.ToString("D4"));  int secondThirdDigitChanged = 1000 \* a + 100 \* c + 10 \* b + d;  Console.WriteLine(secondThirdDigitChanged.ToString("D4"));  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * 9999 * 1000 * 2112 * 1313 | | | |
| **Вход** | | **Изход** | |
| 9999 | | 36  9999  9999  9999 | |
| **Вход** | | **Изход** | |
| 1000 | | 1  0001  0001  1000 | |
| **Вход** | | **Изход** | |
| 2112 | | 6  2112  2112  2112 | |
| **Вход** | | **Изход** | |
| 1313 | | 8  3131  3311  1133 | |
| **Вход** | | **Изход** | |
| 4477 | | 22  7744  7474  4747 | |
| **Вход** | | **Изход** | |
| 9876 | | 30  6789  6879  9786 | |
| **Вход** | | **Изход** | |
| 1234 | | 10  4321  4231  1324 | |
| **Вход** | | **Изход** | |
| 1234 | | 10  4321  4231  1324 | |
| **Вход** | | **Изход** | |
| 2010 | | 3  0102  0012  2100 | |
| **Вход** | | **Изход** | |
| 1111 | | 4  1111  1111  1111 | |
| Задача 11. Бит тест на позиция p. | | | |
| **Условие**  Дадено е цяло положително 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**  Входните данни се въвеждат цяло число. На първия ред се въвежда стойността на числото **n**, на втория - стойността на позицията **р** (0 < р < 31). | | | |
| **Описание на изхода**  Изходът се извежда **1** или **0**. Ако бита на позиция **р** е 1, се извежда **1**, в противен случай – **0**. | | | |
| **Анализ на задачата**  Ползвайте побитови операции:  int n = 35; // 00100011  int p = 6;  int i = 1; // 00000001  int mask = i << p; // Move the 1st bit left by p positions  // If i & mask are positive then the p-th bit of n is 1  Console.WriteLine((n & mask) != 0 ? 1 : 0); | | | |
| **Решение (сорс код)** | | | |
| using System;  class BitTest  {  static void Main()  {  int n = int.Parse(Console.ReadLine());  int p = int.Parse(Console.ReadLine());  int i = 1;  int mask = i << p;  int result = n & mask;  if (result == 0)  {  Console.WriteLine(0);  }  else  {  Console.WriteLine(1);  }  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 15\3 * 239\4 * 257\8 * 1111\3 | | | |
| **Вход** | | **Изход** | |
| 35  6 | | 0 | |
| **Вход** | | **Изход** | |
| 239  4 | | 0 | |
| **Вход** | | **Изход** | |
| 257  8 | | 1 | |
| **Вход** | | **Изход** | |
| 1111  3 | | 0 | |
| **Вход** | | **Изход** | |
| 1111  4 | | 1 | |
| **Вход** | | **Изход** | |
| 1234  7 | | 1 | |
| **Вход** | | **Изход** | |
| 0  3 | | 0 | |
| **Вход** | | **Изход** | |
| 268435456  28 | | 1 | |
| **Вход** | | **Изход** | |
| 1073741824  30 | | 1 | |
| **Вход** | | **Изход** | |
| 32768  15 | | 1 | |
| **Вход** | | **Изход** | |
| 2147450879  15 | | 0 | |
| Задача 12. Установяване бит на позиция p в стойност v. | | | |
| **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**  Входните данни се въвеждат като цели числа. На първия ред се въвежда стойността на **v** , на втория - стойността **p.** | | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако битът на позиция p на цялото число **v** е 1 извеждаме **True**, иначе извеждаме **False**. | | | |
| **Анализ на задачата**  Задачата е аналогична на предната. | | | |
| **Решение (сорс код)** | | | |
| using System;  class IsBit1  {  static void Main(string[] args)  {  int v = int.Parse(Console.ReadLine());  int p = int.Parse(Console.ReadLine());  int i = 1;  int mask = i << p;  int result = v & mask;  if (result != 0)  {  Console.WriteLine(true);  }  else  {  Console.WriteLine(false);  }  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 1 * 512 * 32768 * -2147483648 | | | |
| **Вход** | | **Изход** | |
| 1  0 | | True | |
| **Вход** | | **Изход** | |
| 512  9 | | True | |
| **Вход** | | **Изход** | |
| 512  8 | | False | |
| **Вход** | | **Изход** | |
| 1073742336  30 | | True | |
| **Вход** | | **Изход** | |
| 32768  15 | | True | |
| **Вход** | | **Изход** | |
| 32768  14 | | False | |
| **Вход** | | **Изход** | |
| 2147450879  15 | | False | |
| **Вход** | | **Изход** | |
| 2147450879  31 | | False | |
| **Вход** | | **Изход** | |
| -2147483648  31 | | True | |
| **Вход** | | **Изход** | |
| 1073741824  30 | | True | |
| Задача 13. Промяна на бит на позиция p | | | |
| **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**  Входните данни се въвеждат като цeли числа. На първия ред се въвежда стойността на **n** , на втория - стойността **p** (1 ≤ р ≤ 31), на третия - стойността на **v** (v = 0 или 1). | | | |
| **Описание на изхода**  Изходът се извежда като цяло число на нов ред. | | | |
| **Анализ на задачата**  Ползвайте побитови операции, по аналогия с предната задача. Можете да нулирате бита на позиция p в числото n по следния начин:   |  | | --- | | n = n & (~(1 << p)); |   Можете да установите в единица бита на позиция p в числото n по следния начин:   |  | | --- | | n = n | (1 << p); |   Помислете как можете да комбинирате тези две упътвания. | | | |
| **Решение (сорс код)** | | | |
| using System;  class SetBits  {  static void Main()  {  int n = int.Parse(Console.ReadLine());  int p = int.Parse(Console.ReadLine());  int v = int.Parse(Console.ReadLine());  int mask = v << p;  n = n & (~(1 << p));  int result = n | mask;  Console.WriteLine(result);  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 239\4\1 * 257\8\0 * 1111\12\1 * 0\16\1 | | | |
| **Вход** | | **Изход** | |
| 35  5  0 | | 3 | |
| **Вход** | | **Изход** | |
| 239  4  1 | | 255 | |
| **Вход** | | **Изход** | |
| 257  8  0 | | 1 | |
| **Вход** | | **Изход** | |
| 1111  12  1 | | 5207 | |
| **Вход** | | **Изход** | |
| 123456789  2  0 | | 123456785 | |
| **Вход** | | **Изход** | |
| 0  16  1 | | 65536 | |
| **Вход** | | **Изход** | |
| 2147483647  15  0 | | 2147450879 | |
| **Вход** | | **Изход** | |
| 2146435071  0  0 | | 2146435070 | |
| **Вход** | | **Изход** | |
| 0  31  1 | | -2147483648 | |
| Задача 14. Просто число | | | |
| **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**  Входните данни се въвеждат цяло положително число. На един ред се въвежда стойността на **n** (1 < n < 100). | | | |
| **Описание на изхода**  Изходът се извежда като низ **True** или **False**. Ако числото е просто, се извежда **True**, в противен случай – **False**. | | | |
| **Анализ на задачата**  В математиката **просто число** се нарича всяко естествено число, по-голямо от 1, което има точно два естествени делителя — 1 и самото себе си. Естествените числа, по-големи от едно, които не са прости, се наричат съставни. Числата нула и едно не са нито прости, нито съставни.  Първите няколко прости числа са:  2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107  Основната теорема на аритметиката гласи, че всяко цяло число, по-голямо от 1, може да се представи по единствен начин (с точност до реда на множителите) като произведение на прости числа.   Прочетете за цикли в Интернет. Използвайте цикъл и проверете числото за делимост на всички числа от 2 до корен квадратен от числото. В конкретната задача, тъй като ограничението е само до 100, можете предварително да намерите простите числа от 1 до 100 и да направите проверки дали даденото число **n** e равно на някое от тях. | | | |
| **Решение (сорс код)** | | | |
| using System;  class IsPrimeNumber  {  static void Main()  {  int number = Int32.Parse(Console.ReadLine());  bool isPrime = true;  for (int divisor = 2; divisor <= Math.Sqrt(number); divisor++)  {  if (number % divisor == 0)  {  isPrime = false;  break;  }  }  Console.WriteLine(isPrime);  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 2 * 3 * 89 * 97 | | | |
| **Вход** | | **Изход** | |
| 2 | | True | |
| **Вход** | | **Изход** | |
| 3 | | True | |
| **Вход** | | **Изход** | |
| 4 | | False | |
| **Вход** | | **Изход** | |
| 5 | | True | |
| **Вход** | | **Изход** | |
| 6 | | False | |
| **Вход** | | **Изход** | |
| 7 | | True | |
| **Вход** | | **Изход** | |
| 8 | | False | |
| **Вход** | | **Изход** | |
| 11 | | True | |
| **Вход** | | **Изход** | |
| 89 | | True | |
| **Вход** | | **Изход** | |
| 97 | | True | |
| **Вход** | | **Изход** | |
| 99 | | False | |
| Задача 15. Размяна на битове 3, 4, 5 и 24, 25, 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**  Входните данни се въвеждат като цяло число без знак. На един ред се въвежда стойността на **n**. | | | |
| **Описание на изхода**  Изходът се извежда като цяло число без знак. | | | |
| **Анализ на задачата**   За решението на тази задача използвайте комбинация от задачите за вземане и установяване на бит на определена позиция. | | | |
| **Решение (сорс код)** | | | |
| using System;  class BitExchange  {  static void Main(string[] args)  {  uint number = uint.Parse(Console.ReadLine());  int bitAtPosition3 = PthBit(number, 3);  int bitAtPosition4 = PthBit(number, 4);  int bitAtPosition5 = PthBit(number, 5);  int bitAtPosition24 = PthBit(number, 24);  int bitAtPosition25 = PthBit(number, 25);  int bitAtPosition26 = PthBit(number, 26);  number = ModifiedNumber(number, 3, bitAtPosition24);  number = ModifiedNumber(number, 4, bitAtPosition25);  number = ModifiedNumber(number, 5, bitAtPosition26);  number = ModifiedNumber(number, 24, bitAtPosition3);  number = ModifiedNumber(number, 25, bitAtPosition4);  number = ModifiedNumber(number, 26, bitAtPosition5);  Console.WriteLine(number);  }  private static int PthBit(uint number, int position)  {  uint pthBit = (number >> position) & 1;  return (int)pthBit;  }  private static uint ModifiedNumber(uint number, int position, int bitValue)  {  uint actualP = (uint)bitValue << position;  number = number & (~((uint)1 << position));  uint result = number | actualP;  return result;  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 8 * 56 * 117440568 * 2097151975 | | | |
| **Вход** | | **Изход** | |
| 0 | | 0 | |
| **Вход** | | **Изход** | |
| 1 | | 1 | |
| **Вход** | | **Изход** | |
| 8 | | 16777216 | |
| **Вход** | | **Изход** | |
| 24 | | 50331648 | |
| **Вход** | | **Изход** | |
| 48 | | 100663296 | |
| **Вход** | | **Изход** | |
| 56 | | 117440512 | |
| **Вход** | | **Изход** | |
| 117440568 | | 117440568 | |
| **Вход** | | **Изход** | |
| 67108896 | | 67108896 | |
| **Вход** | | **Изход** | |
| 100663344 | | 100663344 | |
| **Вход** | | **Изход** | |
| 117440512 | | 56 | |
| **Вход** | | **Изход** | |
| 100663296 | | 48 | |
| **Вход** | | **Изход** | |
| 50331648 | | 24 | |
| **Вход** | | **Изход** | |
| 16777216 | | 8 | |
| **Вход** | | **Изход** | |
| 2030043079 | | 2030043079 | |
| **Вход** | | **Изход** | |
| 2046820303 | | 2046820303 | |
| **Вход** | | **Изход** | |
| 2063597527 | | 2063597527 | |
| **Вход** | | **Изход** | |
| 2097151975 | | 2097151975 | |
| Задача 16. Размяна на битове {p, p+1, …, p+k-1} с {q, q+1, …, q+k-1} | | | |
| **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**  Входните данни се въвеждат като цяли числа. На първия ред се въвежда стойността на **n** като цяло число без знак. На втория ред се въвежда стойността на  **р**, на третия ред се въвежда стойността на  **q**, и на четвъртия ред се въвежда стойността на  **k.** | | | |
| **Описание на изхода**  Изходът се извежда като цяло число без знак на един ред. | | | |
| **Анализ на задачата**   Използвайте предната задача и прочетете в интернет как се работи с цикли и масиви (в които да запишете битовете). | | | |
| **Решение (сорс код)** | | | |
| using System;  class BitSequenceExchange  {  static void Main(string[] args)  {  uint number = uint.Parse(Console.ReadLine());  int p = int.Parse(Console.ReadLine());  int q = int.Parse(Console.ReadLine());  int k = int.Parse(Console.ReadLine());  if (p > q)  {  int oldValue = p;  p = q;  q = oldValue;  }  if (p + k >= q)  {  k += p - q - 1;  q += p + k + 1;  }  number = ModifyNumber(number, p, q, k);  Console.WriteLine(number);  }  private static uint ModifyNumber(uint number, int p, int q, int k)  {  int[] pBits = new int[k];  int[] qBits = new int[k];  for (int position = p, i = 0; i < pBits.Length; position++, i++)  {  pBits[i] = PthBit(number, position);  }  for (int position = q, i = 0; i < qBits.Length; position++, i++)  {  qBits[i] = PthBit(number, position);  }  for (int position = p, i = 0; i < qBits.Length; position++, i++)  {  number = ModifiedNumber(number, position, qBits[i]);  }  for (int position = q, i = 0; i < pBits.Length; position++, i++)  {  number = ModifiedNumber(number, position, pBits[i]);  }  return number;  }  private static int PthBit(uint number, int position)  {  uint pthBit = (number >> position) & 1;  return (int)pthBit;  }  private static uint ModifiedNumber(uint number, int position, int bitValue)  {  uint actualP = (uint)bitValue << position;  number = number & (~((uint)1 << position));  uint result = number | actualP;  return result;  }  } | | | |
| **Тестове**  Интересните случаи за тестване са следните:   * Примерният вход от условието на задачата * 56\3\24\3\117440512 * 2030043079\3\24\3\2030043079 * 7\0\28\3\1879048192 * 251658255\0\24\4\251658255 | | | |
| **Вход** | | **Изход** | |
| **56**  **3**  **24**  **3** | | **117440512** | |
| **Вход** | | **Изход** | |
| **2030043079**  **3**  **24**  **3** | | **2030043079** | |
| **Вход** | | **Изход** | |
| **7**  **0**  **28**  **3** | | **1879048192** | |
| **Вход** | | **Изход** | |
| **15**  **0**  **24**  **4** | | **251658240** | |
| **Вход** | | **Изход** | |
| **251658255**  **0**  **24**  **4** | | **251658255** | |
| **Вход** | | **Изход** | |
| **1610612742**  **29**  **1**  **2** | | **1610612742** | |
| **Вход** | | **Изход** | |
| **536870904**  **29**  **0**  **3** | | **536870904** | |
| **Вход** | | **Изход** | |
| 98304  16  15  2 | | 98304 | |
| **Вход** | | **Изход** | |
| **2**  **1**  **30**  **1** | | **1073741824** | |
| **Вход** | | **Изход** | |
| 2  1  30  1 | |  | |
| **Вход** | | **Изход** | |
| **15728880**  **20**  **4**  **4** | | **15728880** | |