# Решения на задачите от глава 8 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

**Иван Стоянов – <mad\_van@abv.bg>**

**Krasin Georgiev - <gkrasin@gmail.com>**

Angel Venchev - <angelvenchev@yahoo.com>

Tsvetomir Kirchev - <tsvetomir.kirchev@gmail.com>

Stefan Stoyanov - <stefan.asenov\_@abv.bg>

|  |  |
| --- | --- |
| 1. ConvertSomeNumbers | |
| **Условие**  Превърнете числата 151, 35, 43, 251 и -0,41 в двоична бройна система. | |
| **Описание на входа**  Няма вход. | |
| **Описание на изхода**  Срещу всяко число се изписва съответното му в двоична бройна система. | |
| **Анализ на задачата**  Поставяме целите числа в масив и ги обхождаме с един цикъл for. В него има цикъл while, който дели числото на 2. Остатъка (1 или 0) се записва в стек. Накрая принтираме елементите от стека. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_01.ConverSomeNumbers  {  class Program  {  static void Main(string[] args)  {  int[] nums = new int[] { 151, 35, 43, 251 };  for (int i = 0; i < nums.Length ; i++)  {  //write the number to convert  Console.Write(nums[i] + " - ");  Stack<int> binNum = new Stack<int>();  //convert number and put 1and0 to stack  while(nums[i]>0)  {  if (nums[i] % 2 == 1) binNum.Push(1);  else binNum.Push(0);  nums[i] /= 2;  }  //print the number backwards  while (binNum.Count > 0)  {  Console.Write(binNum.Pop());  }  Console.WriteLine();  }  }  }  } | |
| **Тестове** | |
| **Вход** | **Изход** |
|  | 151 - 10010111  35 - 100011  43 - 101011  251 - 11111011 |
|  | |
| 02. ConvertBinNumberToDecAndHex | |
| **Условие**  Превърнете числото 1111010110011110**(2)**в шестнадесетична и в десе­тична бройна система. | |
| **Описание на входа**  Няма вход. | |
| **Описание на изхода**  На първия ред изписваме числото в десетичен вид, а на втория в шестнайсетичен. | |
| **Анализ на задачата**  Ползваме две функции за конвертиране. Функцията за десетично число работи като четем числто отзад напред и умножаваме цифрата (0 или 1) по 2 на степен позицията на която сме и всеки път събираме резултата с една променлива. Функцията зашеснайстично число работи като пак отзад напред взимаме по 4 цифри превръщаме ги в десетично число и по резултата избираме символ от 0 до F. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_02.ConvertBinNumberToDecAndHex  {  class Program  {  static void Main(string[] args)  {  string number = "1111010110011110";  int numberInDec = 0;  string numberInHex = null;  numberInDec = ToDec(number);  Console.WriteLine(numberInDec);  numberInHex = ToHex(number).ToString();  Console.WriteLine(numberInHex);  }  private static int ToDec(string number)  {  int stepen = 0;  int numberInDec = 0;  for (int i = number.Length - 1; i >= 0; i--)  {  numberInDec += ((int)number[i] - 48) \* (int)Math.Pow(2, stepen++);  }  return numberInDec;  }  private static string ToHex(string number)  {  char[] hexNum = new char[4];  short pos = 0;  for (int i = 0; i < 4; i++)  {  int stepen = 3;  int num = 0;  for (int j = i \* 4; j < (i \* 4) + 4; j++)  {  num += ((int)number[j] - 48) \* (int)Math.Pow(10, stepen--);  }  if (num == 0) hexNum[pos++] = '0';  else if (num == 1) hexNum[pos++] = '1';  else if (num == 10) hexNum[pos++] = '2';  else if (num == 11) hexNum[pos++] = '3';  else if (num == 100) hexNum[pos++] = '4';  else if (num == 101) hexNum[pos++] = '5';  else if (num == 110) hexNum[pos++] = '6';  else if (num == 111) hexNum[pos++] = '7';  else if (num == 1000) hexNum[pos++] = '8';  else if (num == 1001) hexNum[pos++] = '9';  else if (num == 1010) hexNum[pos++] = 'A';  else if (num == 1011) hexNum[pos++] = 'B';  else if (num == 1100) hexNum[pos++] = 'C';  else if (num == 1101) hexNum[pos++] = 'D';  else if (num == 1110) hexNum[pos++] = 'E';  else if (num == 1111) hexNum[pos++] = 'F';  }  return new string(hexNum);  }  }  } | |
| **Тестове** | |
| **Вход** | **Изход** |
|  | 62878  F59E |
|  | |

|  |  |
| --- | --- |
| 03. ConvertSomeHexToDecAndBin | |
| **Условие**  Превърнете шестнайсетичните числа 2A3E, FA, FFFF, 5A0E9 в двоична и десетична бройна система. | |
| **Описание на входа**  Няма вход. | |
| **Описание на изхода**  На всеки ред се изписва първо шестнайсетичното число след това десетичното и накрая двоичното. | |
| **Анализ на задачата**  За превръщане в десетично взимаме първата цифра от числото и я кастваме в int ако е по-голяма от 9 изваждаме 7 (за да получим кога на цифрата А-10 В-11 и т.н) и резултата умножаваме по 16 на степен позицията от която сме взели числото като на всяка итерация сабираме резултата в една променлива. За превръщане в двоично число взимаме цифрите отзад напред превръщаме ги в десетично число и го слагаме в един цикъл for 4 пъти (F(16) = 1111(2)). Резултата пазим в масив и накрая го ревърсваме. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_03.ConvertSomeHexToDecAndBin  {  class Program  {  //2A3E, FA, FFFF, 5A0E9  static void Main(string[] args)  {  string[] hex = new string[] { "2A3E", "FA", "FFFF", "5A0E9"};  int[] dec = new int[hex.Length];  int[][] bin = new int[hex.Length][];  //to dec  for (int i = 0; i < hex.Length; i++)  {  dec[i] = hexToDec(hex[i]);  }  //to bin  for (int i = 0; i < hex.Length; i++)  {  bin[i] = hexToBin(hex[i]);  }  //print  for (int i = 0; i < hex.Length; i++)  {  Console.Write(hex[i] + " " + dec[i] + " ");  int j = 0;  while (bin[i][j] == 0) j++;  for (; j < bin[i].Length; j++)  {  Console.Write(bin[i][j]);  }  Console.WriteLine();  }  }  private static int hexToDec(string hex)  {  int dec = 0;  int cnt = hex.Length - 1;  int stepen = 0;  while (cnt >= 0)  {  int stepNumber = hex[cnt] - 48;  if (stepNumber > 9) stepNumber -= 7;  dec += stepNumber \* (int)Math.Pow(16, stepen++);  cnt--;  }  return dec;  }  private static int[] hexToBin(string hex)  {  int dec = 0;  int cnt = hex.Length - 1;  int[] theNum = new int[hex.Length \* 4];  int stepen = 0;  while (cnt >= 0)  {  int stepNumber = hex[cnt] - 48;  if (stepNumber > 9) stepNumber -= 7;  dec = stepNumber;  int step = 4 \* stepen;  for (int i = 0; i < 4; i++)  {  theNum[step] = dec % 2;  dec /= 2;  step++;  }  cnt--;  stepen++;  }  Array.Reverse(theNum);  return theNum;  }  }  } | |
| **Тестове** | |
| **Вход** | **Изход** |
|  | 2A3E 10814 10101000111110  FA 250 11111010  FFFF 65535 1111111111111111  5A0E9 368873 1011010000011101001 |
|  | |
| 04. DecToBin | |
| **Условие**  Да се напише програма, която преобразува десетично число в дво­ично. | |
| **Описание на входа**  Въвеждаме кое да е int число | |
| **Описание на изхода**  На екрана се изписва превърнатото число в двоична броина система | |
| **Анализ на задачата**  Делим десетичното число на 2 и остатъка записваме в масив като накрая ревърсваме масива. Ако числото е отрицателно от 1111 (колкото са нужни единици) изваждаме отрицателното число минус едно | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_09\_numericalSystems  {  class Program  {  static void Main(string[] args)  {  int decNum = int.Parse(Console.ReadLine());  string theBinNum = null;  if (decNum >= 0) theBinNum = positiveBin(decNum);  else  {  string tmpBinNum = positiveBin((int)(-decNum - 1));  char[] binNegativ = new char[(((tmpBinNum.Length - 1) / 4) + 1) \* 4];  for (int i = 0; i < binNegativ.Length; i++)  {  binNegativ[i] = '1';  }  for (int i = tmpBinNum.Length-1, j = binNegativ.Length-1; i >= 0; i--)  {  if (tmpBinNum[i] == '1') binNegativ[j--] = '0';  else binNegativ[j--] = '1';  }  theBinNum = new string(binNegativ);  }  Console.WriteLine(theBinNum);  }  private static string positiveBin(int decNum)  {  StringBuilder binNum = new StringBuilder();  do  {  if (decNum % 2 == 1)  {  binNum.Append(1);  }  else  {  binNum.Append(0);  }  decNum /= 2;  }  while (decNum > 0);  return ReverseString(binNum.ToString());  }  public static string ReverseString(string s)  {  char[] arr = s.ToCharArray();  Array.Reverse(arr);  return new string(arr);  }  }  } | |
| **Тестове**  Интересни са няколко случая  0 , 1, -1, 2147483647 | |
| **Вход** | **Изход** |
| 0  1  -1  2147483647 | 0  1  1111  1111111111111111111111111111111 |
|  | |

|  |  |
| --- | --- |
| 05. BinToDec | |
| **Условие**  Да се напише програма, която преобразува двоично число в десе­тично. | |
| **Описание на входа**  Въвеждаме произволно дълго число съставено само от 0 и 1 | |
| **Описание на изхода**  Изписва се числото в десетичен формат | |
| **Анализ на задачата**  Записваме двоичното число в BigInteger(няма ограничение за големината) и го разглеждаме отзад напред като текущата цифра умножаваме с две на степен текущата позиция (броим от нула) и резултата винаги събираме в една променлива, която накрая е резултата ни. Ако дължината на двоичното се е кратна на 8 то се взима предвид отрицателната стойност на числото  (11111111(2) = -1(10)) | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Numerics;  namespace BinToDec  {  class Program  {  static void Main(string[] args)  {  string input = Console.ReadLine();  BigInteger bin = BigInteger.Parse(input);  BigInteger theDec = 0;    string tmpbin = bin.ToString();  if ((input.Length % 8 == 0) && tmpbin[0] == 1 + '0')  {  BigInteger justOnes = new BigInteger();  for (int i = 0; i < tmpbin.Length; i++)  {  justOnes += 1 \* (BigInteger)Math.Pow(10, i);  }  bin = justOnes - bin + 1;  theDec = -toDec(bin);  }  else  {  theDec = toDec(bin);  }  Console.WriteLine(theDec);  }  private static BigInteger toDec(BigInteger bin)  {  BigInteger decNum = 0;  int cnt = 0;  while (bin > 0)  {  decNum += (BigInteger)(bin % 10) \* (BigInteger)Math.Pow(2, cnt);  bin /= 10;  cnt++;  }  return decNum;  }  }  } | |
| **Тестове**  Интересе е случаят с нулата както и двата вариянта  1111 1111(2) = -1(10)  0 1111 1111(2) = 255(10) | |
| **Вход** | **Изход** |
| 0  1  10  1111  11111111  011111111  1010101111001011010101010010101001011001001  1010101111001011010101010010101001011001 | 0  1  2  15  -1  255  5902806110921  -361683393967 |
|  | |
| 06. DecToHex | |
| **Условие**  Да се напише програма, която преобразува десетично число в шестнадесетично. | |
| **Описание на входа**  Въвеждаме кое да е int число | |
| **Описание на изхода**  На екрана се изписва превърнатото число в шестнайсетична броина система | |
| **Анализ на задачата**  Делим десетичното число на 16 и остатъка записваме в масив като накрая ревърсваме масив(ако остатъка е по-голям от 16 изваждаме 7, за да си отговарят буквите на съответните числа 10 - А, 11 - В и т.н.). Ако числото е отрицателно от FFFF (колкото са нужни F-ове) изваждаме отрицателното число минус едно. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_03.DecToHex  {  class Program  {  static void Main(string[] args)  {  int decNum = int.Parse(Console.ReadLine());  char[] hex = new char[8];  if (decNum >= 0)  {  hex = ToHex(decNum);  }  else  {  char[] tmpHex = new char[8];  tmpHex = ToHex(-decNum - 1);  hex = Minus(tmpHex);  }  //print  int i=0;  while(hex[i]==0) i++;  for (int j = i; j < hex.Length; j++)  {  Console.Write(hex[j]);  }  Console.WriteLine();  }  private static char[] ToHex(int decNum)  {  char[] tmp = new char[8];  short pos = 0;  do  {  if (((decNum % 16) >= 0) && ((decNum % 16 <= 9)))  {  tmp[pos] = (char)((decNum % 16) + 48);  }  else  {  tmp[pos] = (char)((decNum % 16) + 48 + 7);  }  decNum = decNum / 16;  pos++;  }  while (decNum > 0);  Array.Reverse(tmp);  return tmp;  }  private static char[] Minus(char[] positivHex)  {  Array.Reverse(positivHex);  char[] lastHex = new char[8];  for (int i = 0; i < 8; i++)  {  if (positivHex[i] == '0') lastHex[i] = 'F';  else if (positivHex[i] == '1') lastHex[i] = 'E';  else if (positivHex[i] == '2') lastHex[i] = 'D';  else if (positivHex[i] == '3') lastHex[i] = 'C';  else if (positivHex[i] == '4') lastHex[i] = 'B';  else if (positivHex[i] == '5') lastHex[i] = 'A';  else if (positivHex[i] == '6') lastHex[i] = '9';  else if (positivHex[i] == '7') lastHex[i] = '8';  else if (positivHex[i] == '8') lastHex[i] = '7';  else if (positivHex[i] == '9') lastHex[i] = '6';  else if (positivHex[i] == 'A') lastHex[i] = '5';  else if (positivHex[i] == 'B') lastHex[i] = '4';  else if (positivHex[i] == 'C') lastHex[i] = '3';  else if (positivHex[i] == 'D') lastHex[i] = '2';  else if (positivHex[i] == 'E') lastHex[i] = '1';  else if (positivHex[i] == 'F') lastHex[i] = '0';  else if (positivHex[i] == 0) lastHex[i] = 'F';  }  Array.Reverse(lastHex);  return lastHex;  }  }  } | |
| **Тестове**  Интересни са няколко случая  0 , 1, -1, 2147483647 | |
| **Вход** | **Изход** |
| 0  1  -1  -2  128  2147483647  -2147483647  -2147483648 | 0  1  FFFFFFFF  FFFFFFFE  80  7FFFFFFF  80000001  80000000 |
|  | |

|  |  |
| --- | --- |
| 07. HexToDec | |
| **Условие**  Да се напише програма, която преобразува шестнадесетично число в десетично. | |
| **Описание на входа**  Въвеждаме произволно дълъг стринг с главни букви и цифри от 1до9 | |
| **Описание на изхода**  Изписва се числото в десетичен формат | |
| **Анализ на задачата**  Взимаме една по една цифрите от стринга отзад напред като ги кастваме в int като ги умножаваме по 16 на степен съответната позиция и резултата пазим в променливата dec | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Numerics;  namespace \_03.HexToDec  {  class Program  {  static void Main(string[] args)  {  string hex = Console.ReadLine();  BigInteger dec = 0;    dec = hexToDec(hex, dec);  Console.WriteLine(dec);  }  private static BigInteger hexToDec(string hex, BigInteger dec)  {  int pos = hex.Length - 1;  int stepen = 0;  while (pos >= 0)  {  int stepNumber = hex[pos] - 48;  if (stepNumber > 9) stepNumber -= 7;  dec += stepNumber \* (BigInteger)Math.Pow(16, stepen++);  pos--;  }  return dec;  }  }  } | |
| **Тестове**  Числото винаги е положително. Тестваме с гранични случаи 0, 1, F, FFFF | |
| **Вход** | **Изход** |
| 0  1  2  A  F  AF  34  FFFF  FF  FFF  FFFFFFF  FFFFFFFFFFFFFFF | 0  1  2  10  15  175  52  65535  255  4095  268435455  1152921504606846975 |
|  | |
| 08. HexToBin | |
| **Условие**  Да се напише програма, която преобразува шестнадесетично число в двоично. | |
| **Описание на входа**  Въвеждаме произволно дълъг стринг с главни букви от АдоF(главни букви) и цифри от 1до9 | |
| **Описание на изхода**  На екрана се изписва превърнатото число в двоична броина система | |
| **Анализ на задачата**  Взимаме цифрите отзад напред и една по една ги вкарваме в цикъл и с 4 итерации (като преди това кастваме дадената цифра в int) В цикъла превръщаме шестнайсетичните цифри във вид на четири символно бинарно число (0(16) = 0000(2), 3(16) = 0011(2), F(16) = 1111(2) ) | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_05.HexToBin  {  class Program  {  static void Main(string[] args)  {  string hex = Console.ReadLine();  int[] bin = new int[4\*hex.Length];  bin = hexToBin(hex);  Array.Reverse(bin);  //print  foreach (var item in bin)  {  Console.Write(item);  }  }  private static int[] hexToBin(string hex)  {  int dec = 0;  int cnt = hex.Length-1;  int[] theNum = new int[hex.Length\*4];  int stepen = 0;  while (cnt >= 0)  {  int stepNumber = hex[cnt] - 48;  if (stepNumber > 9) stepNumber -= 7;  dec = stepNumber;  int step = 4\*stepen;  for (int i = 0; i < 4; i++)  {  theNum[step] = dec % 2;  dec /= 2;  step++;  }  cnt--;  stepen++;  }  return theNum;  }  }  } | |
| **Тестове**  0-ла е граничен случай, нямаме лимит за дължина и нямаме отрицателни числа.  0 , 1, -1, А | |
| **Вход** | **Изход** |
| 0  1  A  F  1F  FFFABC3  FFFF  0F  EF | 0000  0001  1010  1111  00011111  1111111111111010101111000011  1111111111111111  00001111  11101111 |
|  | |

|  |  |
| --- | --- |
| 09. BinToHex | |
| **Условие**  Да се напише програма, която преобразува двоично число в шестна­десетично. | |
| **Описание на входа**  Въвеждаме произволно дълъг стринг с цифри от 1 и 0 | |
| **Описание на изхода**  Изписва се числото в шестнайсетичен формат | |
| **Анализ на задачата**  Взимаме 4 поредни цифри от стринга залепя ме ги една го друга (от вида 0101) и таблично ги свиряваме с техните шестнайсетични еквиваленти. (1010(2) = А(16), 0010(2) = 2(16)) | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace BinToHex  {  class Program  {  static void Main(string[] args)  {  string binNum1 = Console.ReadLine();  char[] binNum = binNum1.ToCharArray();  Array.Reverse(binNum);  if (binNum1.Length < 4 && int.Parse(binNum1) == 0) Console.WriteLine(0);  else  {  char[] hex = new char[(binNum.Length / 4) + 1];  hex = ToHex(binNum);  //print  int i = 0;  while (hex[i] == 0) i++;  if (hex[i] == '0') i++;  for (int j = i; j < hex.Length; j++)  {  Console.Write(hex[j]);  }  Console.WriteLine();  }  }  private static char[] ToHex(char[] binNum)  {  char[] tmpBig = new char[((binNum.Length / 4) + 1) \* 4];  short step = 0;  int length = ((binNum.Length / 4) + 1);  for (int i = 0; i < length; i++)  {  int little=0;  int num = 0;  for (int j = i\*4; j < (i\*4)+4; j++)  {  if (j >= binNum.Length) break;  num += (int)(binNum[j] - 48) \* (int)Math.Pow(10, little++);  }  if (num == 0) tmpBig[step++] = '0';  else if (num == 1) tmpBig[step++] = '1';  else if (num == 10) tmpBig[step++] = '2';  else if (num == 11) tmpBig[step++] = '3';  else if (num == 100) tmpBig[step++] = '4';  else if (num == 101) tmpBig[step++] = '5';  else if (num == 110) tmpBig[step++] = '6';  else if (num == 111) tmpBig[step++] = '7';  else if (num == 1000) tmpBig[step++] = '8';  else if (num == 1001) tmpBig[step++] = '9';  else if (num == 1010) tmpBig[step++] = 'A';  else if (num == 1011) tmpBig[step++] = 'B';  else if (num == 1100) tmpBig[step++] = 'C';  else if (num == 1101) tmpBig[step++] = 'D';  else if (num == 1110) tmpBig[step++] = 'E';  else if (num == 1111) tmpBig[step++] = 'F';  }  Array.Reverse(tmpBig);  return tmpBig;  }  }  } | |
| **Тестове**  Числото винаги е положително. Тестваме с гранични случаи 0, 1, 10, 1111, 11111 | |
| **Вход** | **Изход** |
| 0  1  10  11  1111  11111  11111111111111111111111111111111  110101111111110100111111111111010011111111101 | 0  1  2  3  F  1F  FFFFFFFF  1AFFA7FFA7FD |
|  | |
| 10. BinToDecHorner | |
| **Условие**  Да се напише програма, която преобразува двоично число в десетично по схемата на Хорнер. | |
| **Описание на входа**  Въвеждаме произволно дълъг стринг с цифри от 1 или 0 | |
| **Описание на изхода**  На екрана се изписва превърнатото число в десетична броина система | |
| **Анализ на задачата**  Схема на Хорнер - при тази схема се извършва умножение на най-лявата цифра по две и събиране със съседната й вдясно. Този резултат се умножава по две и се прибавя следващата съседна цифра от числото (цифрата вдясно). Това продължава до изчерпване на всички цифри в числото, като последната цифра от числото се добавя без умножаване. Пример:  **1001**(2) = ((1 × 2 + 0) × 2 + 0) × 2 + 1 = 2 × 2 × 2 + 1 = **9** | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Numerics;  namespace \_10.BinToDecHorner  {  class Program  {  static void Main(string[] args)  {  string binNum = Console.ReadLine();  BigInteger decNumber = 0;  if (binNum == "0") { }  else if (binNum == "1") decNumber = 1;  else  {  decNumber = ((BigInteger)(binNum[0] - '0') \* 2) + (BigInteger)(binNum[1] - '0');  for (int i = 1; i < binNum.Length - 1; i++)  {  decNumber = (decNumber \* 2) + (BigInteger)(binNum[i + 1] - 48);  }  }  Console.WriteLine(decNumber);  }  }  } | |
| **Тестове**  0-ла е граничен случай, нямаме лимит за дължина и нямаме отрицателни числа.  0 , 1, 10, 1111 | |
| **Вход** | **Изход** |
| 0  1  10  10100100  111  1111  10000  10111111111111111111111111111001111111111111 | 0  1  2  164  7  15  16  13194139508735 |
|  | |

А

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11. RomanToArabic | | | | |
| **Условие**  Да се напише програма, която преобразува римските числа в арабски. | | | | |
| **Описание на входа**  Въвеждаме стринг със символите 'i', 'v', 'x', 'l', 'c', 'd' или 'm' | | | | |
| **Описание на изхода**  Изписва се числото в арабски цифри | | | | |
| **Анализ на задачата**  Взимат се символите подред като всеки отговаря на конкретна стойност. Ако до края на низа има символ с по-голяма стойност то тази стойност се взима като отрицателна. На всяка итерация събираме резулатата в променлижата arabic. | | | | |
| **Решение (сорс код)** | | | | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_11.RomanToArabic  {  class Program  {  static char[] rimNums = { 'i', 'v', 'x', 'l', 'c', 'd', 'm' };  static void Main(string[] args)  {  string roman = Console.ReadLine();  int arabic = 0;    for (int i = 0; i < roman.Length; i++)  {  if (!CheckToEnd(roman, i, convertToValue(roman[i])))  {  arabic -= GetValue(roman[i]);  }  else arabic += GetValue(roman[i]);  }  Console.WriteLine(arabic);  }  static int GetValue(char c)  {  if (c == 'm') return 1000;  else if (c == 'd') return 500;  else if (c == 'c') return 100;  else if (c == 'l') return 50;  else if (c == 'x') return 10;  else if (c == 'v') return 5;  else if (c == 'i') return 1;  return 0;  }  static bool CheckToEnd(string roman, int pos, int value)  {  for (int i = pos+1; i < roman.Length; i++)  {  if (value < convertToValue(roman[i])) return false;  }  return true;  }  static int convertToValue(char c)  {  for (int i = 0; i < rimNums.Length; i++)  {  if (c == rimNums[i]) return i;  }  return -1;  }  }  } | | | | |
| **Тестове**  Тестваме с различни вариянти. | | | | |
| **Вход** | **Изход** | | | |
| i  ii  iii  iv  v  xiv  mdc  mdlc | 1  2  3  4  5  14  1600  1550 | | | |
|  | | | | |
| 12. ArabicToRoman | | | | |
| **Условие**  Да се напише програма, която преобразува арабските числа в римски. | | | | |
| **Описание на входа**  Въвеждаме число м/у 1 и 3999 | | | | |
| **Описание на изхода**  На екрана се изписва числото в римски цифри. | | | | |
| **Анализ на задачата**  Подаваме числото на функция, която проверя всички възможни случай от наи-големия към най-малкия и постепенно започва да оформя числото. Първо ако е по голямо от 1000 записва ”М” в стринг и изважда 1000 от числото 1000 повтаря това докато числото е по-голямо от 1000 и продължава да изпълнява свички случаи. | | | | |
| **Решение (сорс код)** | | | | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_12.ArabicToRoman  {  class Program  {  static string ToRoman(uint arabic)  {  StringBuilder result = new StringBuilder();  for (int i = 0; i < arabic; i++)  {  while (arabic >= 1000)  {  //check for thousands place  result.Append("M");  arabic = arabic - 1000;  }  while (arabic >= 900)  {  //check for nine hundred place  result.Append("CM");  arabic = arabic - 900;  }  while (arabic >= 500)  {  //check for five hundred place  result.Append("D");  arabic = arabic - 500;  }  while (arabic >= 400)  {  //check for four hundred place  result.Append("CD");  arabic = arabic - 400;  }  while (arabic >= 100)  {  //check for one hundred place  result.Append("C");  arabic = arabic - 100;  }  while (arabic >= 90)  {  //check for ninety place  result.Append("XC");  arabic = arabic - 90;  }  while (arabic >= 50)  {  //check for fifty place  result.Append("L");  arabic = arabic - 50;  }  while (arabic >= 40)  {  // check for forty place  result.Append("XL");  arabic = arabic - 40;  }  while (arabic >= 10)  {  // check for tenth place  result.Append("X");  arabic = arabic - 10;  }  while (arabic >= 9)  {  //check for nineth place  result.Append("IX");  arabic = arabic - 9;  }  while (arabic >= 5)  {  //check for fifth place  result.Append("V");  arabic = arabic - 5;  }  while (arabic >= 4)  {  //check for fourth place  result.Append("IV");  arabic = arabic - 4;  }  while (arabic >= 1)  {  //check for first place  result.Append("I");  arabic = arabic - 1;  }  }  return result.ToString();  }  static void Main(string[] args)  {  uint n = uint.Parse(Console.ReadLine());  if (n >= 4000 || n==0) Console.WriteLine("Error");  else Console.WriteLine(ToRoman(n));  }  }  } | | | | |
| **Тестове**  Числата трябва да са от 1 до 3999 | | | | |
| **Вход** | **Изход** | | | |
| 0  1  3  6  9  45  450  2345  3999 | Error  I  III  VI  IX  XLV  CDL  MMCCCXLV  MMMCMXCIX | | | |
|  | | | | |
| 13. ConvertoToAll | | | | |
| **Условие**  Да се напише програма, която по зададени N, S, D (2 ≤ S, D ≥ 16) преобразува числото N от бройна система с основа S към бройна система с основа D. | | | | |
| **Описание на входа**  Първо въвеждаме две числа, всяко на нов ред. Първото е от коя бройна система искаме да преминем, а второто в коя бройна система искаме да преминем. На третия ред въвеждаме самото число. | | | | |
| **Описание на изхода**  На екрана се изписва конвертираното число. | | | | |
| **Анализ на задачата**  Имаме две функции. Едната преобразува подаденото число от съответната бройна система в десетично. А втората функция преобразува десетичното в желаната бройна система. Първата функция | | | | |
| **Решение (сорс код)** | | | | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_07.ConvertoToAll  {  class Program  {  static void Main(string[] args)  {  int from = int.Parse(Console.ReadLine());  int to = int.Parse(Console.ReadLine());  string numToConvert = Console.ReadLine();  long dec = 0;    dec = FromToDec(numToConvert,from);  char[] newNumber = new char[numToConvert.Length\*4];  newNumber = DecToTo(dec, to, numToConvert.Length \* 4);  //print  int i=0;  while (newNumber[i] == 0) i++;  for (; i < newNumber.Length; i++)  {  Console.Write(newNumber[i]);  }  Console.WriteLine();    }  private static long FromToDec(string num, int from)  {  long dec = 0;  int cnt = num.Length - 1;  int stepen = 0;  while (cnt >= 0)  {  int stepNumber = num[cnt] - 48;  if (stepNumber > 9) stepNumber -= 7;  dec += stepNumber \* (int)Math.Pow(from, stepen++);  cnt--;  }  return dec;  }  private static char[] DecToTo(long decNum, int TO, int lenght)  {  char[] tmp = new char[lenght];  short cnt = 0;  do  {  if (((decNum % TO) >= 0) && ((decNum % TO <= 9)))  {  tmp[cnt] = (char)((decNum % TO) + 48);  }  else  {  tmp[cnt] = (char)((decNum % TO) + 48 + 7);  }  decNum = decNum / TO;  cnt++;  }  while (decNum > 0);  Array.Reverse(tmp);  return tmp;  }  }  } | | | | |
| **Тестове**  Въведените числа трябва да са от от двоична до шестнайсетична бройна система, а ако има символи трябва да са с главни букви. | | | | |
| **Вход** | | | **Изход** | |
| 2  10  1010  10  2  9  2  16  1111  16  10  FA  10  3  7  21 | | | 10  1001  F  250  21 | |
|  | | | | |
| 14. DisplayDecInBin | | | | |
| **Условие**  Да се напише програма, която по дадено цяло число извежда на конзолата двоичното представяне на числото. | | | | |
| **Описание на входа**  Въвеждаме число в десетична бройна система. | | | | |
| **Описание на изхода**  На екрана се изписва конвертираното число в двоична бройна система. | | | | |
| **Анализ на задачата**  Тъй като вече имаме такава задача, за целта ще използваме системния метод за конвертиране. | | | | |
| **Решение (сорс код)** | | | | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_14.DisplayDecInBin  {  class Program  {  static void Main(string[] args)  {  int n = int.Parse(Console.ReadLine());  Console.WriteLine(Convert.ToString(n,2));  }  }  } | | | | |
| **Тестове**  Въведените различни стойности за тестване. | | | | |
| **Вход** | | | **Изход** | |
| 10  15  16  2  1  0  -1  -2 | | | 1010  1111  10000  10  1  0  11111111111111111111111111111111  11111111111111111111111111111110 | |
|  | | | | |
| 15. DifferenceBetweenDecimalAndDouble | | | | |
| **Условие**  Опитайте да сумирате 50 000 000 пъти числото 0.000001. Използвайте цикъл и събиране (не директно умножение). Опитайте с типовете float и double и след това с decimal. Забелязвате ли разликата в резултатите и в скоростта? | | | | |
| **Описание на входа** | | | | |
| **Описание на изхода**  Извеждаме на екрана времето, за което се извършва даденото действие съответно за float, double и decimal. | | | | |
| **Анализ на задачата**  Имаме три цикъла от 0 до 50 000 000, в които се събира числото 0.000001. В първия го събираме в float променлива, в втория – double, а третия – decimal като след края на всеки цикъл замерваме времето. | | | | |
| **Решение (сорс код)** | | | | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_15.DifferenceBetweenDecimalAndDouble  {  class Program  {  static void Main(string[] args)  {  int cicles = 50000000;  DateTime start = DateTime.Now;  float num1 = 0.000001f;  float result = 0;  for (int i = 0; i < cicles; i++)  {  result += num1;  }  DateTime end = DateTime.Now;  Console.WriteLine("Float seconds = {0}", end-start);  start = DateTime.Now;  double num2 = 0.000001d;  double resultD = 0;  for (int i = 0; i < cicles; i++)  {  resultD += num2;  }  end = DateTime.Now;  Console.WriteLine("Double seconds = {0}", end - start);  start = DateTime.Now;  decimal num3 = 0.000001m;  decimal resultM = 0;  for (int i = 0; i < cicles; i++)  {  resultM += num3;  }  end = DateTime.Now;  Console.WriteLine("Decimal seconds = {0}", end - start);  }  }  } | | | | |
| **Тестове** | | | | |
| **Вход** | | **Изход** | | |
|  | | Float seconds = 00:00:00.6390365  Double seconds = 00:00:00.6350363  Decimal seconds = 00:00:05.0582893 | | |
|  | | | | |
| 16. FloatToBinary | | | | |
| **Условие**  Да се напише програма, която отпечатва стойността на мантисата, знака на мантисата и стойността на експонентата за числа тип float (32-битови числа с плаваща запетая съгласно стандарта IEEE 754). Пример: за числото -27,25 да се отпечата: знак = 1, експонента = 10000011, мантиса = 10110100000000000000000. | | | | |
| **Описание на входа**  Въвеждаме float число. | | | | |
| **Описание на изхода**  На екрана се извежда на един ред разделени с един интервал първо знака на числото (0 за положително и 1 за отрицателно), след това експонентата с дължина 8 цифри и накрая мантисата с дължина 23 цифри. | | | | |
| **Анализ на задачата**  Ако зададеното число е по-голямо или равно на 1е-126, го представяме като число (означаваме го с M) между 1 и 2, умножено по 2 на някаква степен (означаваме я с Е) (умножаваме или делим числото на две докато попадне в желания диапазон, в същото време отброяваме степените на двойката E). За да получим мантисата (в десетичен вид) от M вадим 1-ца. За да получим мантисата в двоичен вид, последователно проверяваме дали M е по-голямо или по-малко от специално натрупвана сума от отрицателни степени на двойката (2e-i) (съответно добавяме 1 или 0 към двоичното число). За получаване  на експонентата в двоичен вид, първо увеличаваме E със 127, след това я преобразуваме от десетично число в двоично. За получаване на знака на мантисата проверяваме дали числото е положително (знак 0) или отрицателно (знак 1).  Ако зададеното число е по-малко от 1е-126, тогава осемте знака на експонентата са нули, а от M не се вади 1-ца.  Алтернативен подход е да се използва специалният метод за представяне на числа с плаваща запетая с двойна точност като 64 битово цяло число System. BitConverter.DoubleToInt64Bits(<double>), след което да се приложат подходящи побитови операции (измествания и битови маски). | | | | |
| **Решение (сорс код)** | | | | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace \_12  {  class FloatToBin  {  static void Main()  {  float signedFloatNum = float.Parse(Console.ReadLine());  int exponentDec = 0;  char sign = '0';  float normalizationLimit = 1;  for (int i = 0; i < 126; i++)  {  normalizationLimit /= 2;  }  if (signedFloatNum < 0)  {  sign = '1';  signedFloatNum = -signedFloatNum;  }  string exponentBin = "";  string mantissaBin = "";  if (signedFloatNum == 0)  {  exponentBin = "00000000";  mantissaBin = "00000000000000000000000";  }  else if (signedFloatNum >= normalizationLimit)  {  if (signedFloatNum >= 2)  {  while (signedFloatNum >= 2)  {  exponentDec++;  signedFloatNum /= 2;  }  }  if (signedFloatNum < 1)  {  while (signedFloatNum < 1)  {  exponentDec--;  signedFloatNum \*= 2;  }  }  float mantissaDec = signedFloatNum - 1;  float tmpN = 0;  float dN = 0.5f;  for (int i = 0; i < 23; i++)  {  if (mantissaDec >= tmpN + dN)  {  tmpN += dN;  mantissaBin = mantissaBin + '1';  }  else  {  mantissaBin += '0';  }  dN /= 2;  }  int exponentDecShift = exponentDec + 127;  for (int i = 0; i < 8; i++)  {  if (exponentDecShift % 2 == 1)  {  exponentBin = "1" + exponentBin;  }  else  {  exponentBin = "0" + exponentBin;  }  exponentDecShift /= 2;  }  }  else  {  exponentBin = "00000000";  signedFloatNum /= normalizationLimit;  float mantissaDec = signedFloatNum;  float tmpN = 0;  float dN = 0.5f;  for (int i = 0; i < 23; i++)  {  if (mantissaDec >= tmpN + dN)  {  tmpN += dN;  mantissaBin = mantissaBin + '1';  }  else  {  mantissaBin += '0';  }  dN /= 2;  }  }  Console.WriteLine("{0} {1} {2}", sign, exponentBin, mantissaBin);  }  }  } | | | | |
| **Тестове** | | | | |
| **Вход** | | | | **Изход** |
| 1e-40  0  12345678.3214  -123.324324243423  0.12121221  -0.001011001  -1.00001  -1.000000001  0.0000000000000000000001 | | | | 0 00000000 00000010001011011000010  0 00000000 00000000000000000000000  0 10010110 01111000110000101001110  1 10000101 11101101010011000001110  0 01111011 11110000011111000011011  1 01110101 00001001000001110010000  1 01111111 00000000000000001010100  1 01111111 00000000000000000000000  0 00110101 11100011100100100000001 |