**Numerals Table**

|  |  |
| --- | --- |
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

**ValidateInput**

**Test – 1 = True**

return integer == 1;

**Test – 2 = True**

return integer >= 1;

**Test – 1-3999 = True** (No Change)

**Test – -999999-0 = False** (No Change)

**Test – 4000-999999 = False**

return ((integer >= 1) && (integer <= 3999));

**Convert**

**Test – 1 = I**

return "I";

**Test – 2 = II**

if (integer == 1)

{

return "I";

}

return "II";

**Test – 3 = III**

if (integer == 1)

{

return "I";

}

if (integer == 2)

{

return "II";

}

return "III";

*Refactor*

StringBuilder sb = new StringBuilder();

for (int i = 1; i <= integer; i++)

{

sb.Append("I");

}

return sb.ToString();

**Test – 4 = IV**

StringBuilder sb = new StringBuilder();

if (integer <= 3)

{

for (int i = 1; i <= integer; i++)

{

sb.Append("I");

}

}

else

{

sb.Append("IV");

}

return sb.ToString();

**Test – 5 = V**

StringBuilder sb = new StringBuilder();

if (integer <= 3)

{

for (int i = 1; i <= integer; i++)

{

sb.Append("I");

}

}

else if (integer == 4)

{

sb.Append("IV");

}

else

{

sb.Append("V");

}

return sb.ToString();

**Test – 6 = VI**

if (integer <= 3)

{

for (int i = 1; i <= integer; i++)

{

sb.Append("I");

}

}

else if (integer == 4)

{

sb.Append("IV");

}

else if (integer == 5)

{

sb.Append("V");

}

else

{

sb.Append("VI");

}

**Test – 7 = VII**

else if (integer == 4)

{

sb.Append("IV");

}

else if (integer == 5)

{

sb.Append("V");

}

else if (integer == 6)

{

sb.Append("VI");

}

else

{

sb.Append("VII");

}

*Refactor*

else if (integer == 5)

{

sb.Append("V");

}

else if (integer >= 6)

{

sb.Append("V");

sb.Append(Convert(integer - 5));

}

*Refactor*

else if (integer >= 5)

{

sb.Append("V");

sb.Append(Convert(integer - 5));

}

**Test – 8 = VIII** (No Change)

**Test – 9 = IX** (Initial Result = VIV)

else if (integer >= 5 && integer < 9)

{

sb.Append("V");

sb.Append(Convert(integer – 5));

}

else

{

sb.Append("IX");

}

**Test – 10 = X**

else if (integer >= 5 && integer < 9)

{

sb.Append("V");

sb.Append(Convert(integer – 5));

}

else if (integer == 9)

{

sb.Append("IX");

}

else

{

sb.Append("X");

}

*Refactor*

StringBuilder sb = new StringBuilder();

if (integer >= 10)

{

sb.Append("X");

integer -= 10;

}

if (integer == 9)

{

sb.Append("IX");

integer -= 9;

}

if (integer >= 5)

{

sb.Append("V");

integer -= 5;

}

if (integer == 4)

{

sb.Append("IV");

integer -= 9;

}

while (integer >= 1)

{

sb.Append('I');

integer--;

}

return sb.ToString();

*Refactor*

StringBuilder sb = new StringBuilder();

AddDigitToString(ref integer, 10, "X", sb);

AddDigitToString(ref integer, 9, "IX", sb);

AddDigitToString(ref integer, 5, "V", sb);

AddDigitToString(ref integer, 4, "IV", sb);

while (integer >= 1)

{

AddDigitToString(ref integer, 1, "I", sb);

}

return sb.ToString();

private static void AddDigitToString(ref int integer, int value, string digit, StringBuilder sb)

{

if (integer >= value)

{

sb.Append(digit);

integer -= value;

}

}

**Test – 11 = XI** (No Change)

**Test – 12 = XII** (No Change)

**Test – 13 = XIII** (No Change)

**Test – 14 = XIV** (No Change)

**Test – 15 = XV** (No Change)

**Test – 16 = XVI** (No Change)

**Test – 17 = XVII** (No Change)

**Test – 18 = XVIII** (No Change)

**Test – 19 = XIX** (No Change)

**Test – 20 = XX** (Initial Result = XIXI)

while (integer >= 10)

{

AddDigitToString(ref integer, 10, "X", sb);

}

AddDigitToString(ref integer, 9, "IX", sb);

AddDigitToString(ref integer, 5, "V", sb);

AddDigitToString(ref integer, 4, "IV", sb);

while (integer >= 1)

{

AddDigitToString(ref integer, 1, "I", sb);

}

**Test – 40 = XL**

AddDigitToString(ref integer, 40, "XL", sb);

while (integer >= 10)

{

AddDigitToString(ref integer, 10, "X", sb);

}

**Test – 50 = L**

AddDigitToString(ref integer, 50, "L", sb);

AddDigitToString(ref integer, 40, "XL", sb);

**Test – 90 = XC** (Initial Result = LXL : See Test 9)

StringBuilder sb = new StringBuilder();

AddDigitToString(ref integer, 90, "XC", sb);

AddDigitToString(ref integer, 50, "L", sb);

AddDigitToString(ref integer, 40, "XL", sb);

while (integer >= 10)

{

AddDigitToString(ref integer, 10, "X", sb);

}

AddDigitToString(ref integer, 9, "IX", sb);

AddDigitToString(ref integer, 5, "V", sb);

AddDigitToString(ref integer, 4, "IV", sb);

while (integer >= 1)

{

AddDigitToString(ref integer, 1, "I", sb);

}

return sb.ToString();

*Refactor*

StringBuilder sb = new StringBuilder();

AddDigitRangeToString(ref integer, 100, "C", 50, "L", 10, "X", sb);

AddDigitRangeToString(ref integer, 10, "X", 5, "V", 1, "I", sb);

return sb.ToString();

private static void AddDigitRangeToString(ref int integer, int highValue, string highDigit, int middleValue, string middleDigit, int lowValue, string lowDigit, StringBuilder sb)

{

AddDigitToString(ref integer, highValue - lowValue, String.Concat(lowDigit, highDigit), sb);

AddDigitToString(ref integer, middleValue, middleDigit, sb);

AddDigitToString(ref integer, middleValue - lowValue, String.Concat(lowDigit, middleDigit), sb);

while (integer >= lowValue)

{

AddDigitToString(ref integer, lowValue, lowDigit, sb);

}

}

**Test – 100 = C**

AddDigitToString(ref integer, 100, "C", sb);

AddDigitRangeToString(ref integer, 100, "C", 50, "L", 10, "X", sb);

AddDigitRangeToString(ref integer, 10, "X", 5, "V", 1, "I", sb);

**Test – 200 = CC** (Initial Result = CXCX : See Test 20)

AddDigitRangeToString(ref integer, 1000, "M", 500, "D", 100, "C", sb);

Refactor

struct ValueDigitPair

{

public Int32 Value;

public String Digit;

public ValueDigitPair(Int32 value, String digit)

{

Value = value;

Digit = digit;

}

}

StringBuilder sb = new StringBuilder();

var thousand = new ValueDigitPair(1000, "M");

var fiveHundred = new ValueDigitPair(500, "D");

var oneHundred = new ValueDigitPair(100, "C");

var fifty = new ValueDigitPair(50, "L");

var ten = new ValueDigitPair(10, "X");

var five = new ValueDigitPair(5, "V");

var one = new ValueDigitPair(1, "I");

AddDigitRangeToString(ref integer, thousand, fiveHundred, oneHundred, sb);

AddDigitRangeToString(ref integer, oneHundred, fifty, ten, sb);

AddDigitRangeToString(ref integer, ten, five, one, sb);

return sb.ToString();

private static void AddDigitRangeToString(ref int integer, ValueDigitPair high, ValueDigitPair middle, ValueDigitPair low, StringBuilder sb)

{

AddDigitToString(ref integer, high.Value – low.Value, String.Concat(low.Digit, high.Digit), sb);

AddDigitToString(ref integer, middle.Value, middle.Digit, sb);

AddDigitToString(ref integer, middle.Value - low.Value, String.Concat(low.Digit, middle.Digit), sb);

while (integer >= low.Value)

{

AddDigitToString(ref integer, low.Value, low.Digit, sb);

}

}

**Test – 1000 = M**

var thousand = new ValueDigitPair(1000, "M");

var fiveHundred = new ValueDigitPair(500, "D");

var oneHundred = new ValueDigitPair(100, "C");

var fifty = new ValueDigitPair(50, "L");

var ten = new ValueDigitPair(10, "X");

var five = new ValueDigitPair(5, "V");

var one = new ValueDigitPair(1, "I");

AddDigitToString(ref integer, thousand.Value, thousand.Digit, sb);

AddDigitRangeToString(ref integer, thousand, fiveHundred, oneHundred, sb);

AddDigitRangeToString(ref integer, oneHundred, fifty, ten, sb);

AddDigitRangeToString(ref integer, ten, five, one, sb);

**Test – 2000 = MM**

while (integer >= 1000)

{

AddDigitToString(ref integer, thousand.Value, thousand.Digit, sb);

}

**Test – 3999 = MMMCMXCIX** (No Change)