

Homework: C Loops

This document defines the homework assignments from [the "C Programming" Course @ Software University](#). Please submit as homework a single **zip / rar / 7z** archive holding the solutions (source code) of all below described problems.

Problem 1. Numbers from 1 to N

Write a program that enters from the console a positive integer **n** and **prints all the numbers from 1 to n**, on a single line, separated by a space. Examples:

| n | output |
|---|-----------|
| 3 | 1 2 3 |
| 5 | 1 2 3 4 5 |

Problem 2. Numbers Not Divisible by 3 and 7

Write a program that enters from the console a positive integer **n** and prints all the **numbers from 1 to n not divisible by 3 and 7**, on a single line, separated by a space. Examples:

| n | output |
|----|--------------|
| 3 | 1 2 |
| 10 | 1 2 4 5 8 10 |

Problem 3. Min, Max, Sum and Average of N Numbers

Write a program that reads from the console a sequence of **n** integer numbers and returns the **minimal**, the **maximal** number, the sum and the average of all numbers (displayed with 2 digits after the decimal point). The **input** starts by the number **n** (alone in a line) followed by **n lines**, each holding an integer number. The **output** is like in the examples below. Examples:

| input | output | input | output |
|-------|------------|-------|-------------|
| 3 | min = 1.00 | 2 | min = -1.00 |
| 2 | max = 5.00 | -1 | max = 4.00 |
| 5 | sum = 8.00 | 4 | sum = 3.00 |
| 1 | avg = 2.67 | | avg = 1.50 |

Problem 4. Print a Deck of 52 Cards

Write a program that generates and prints **all possible cards from a standard deck of 52 cards** (without the jokers). The cards should be printed using the classical notation (like 5S (♠), AH (♥), 9C (♣) and KD (♦)). The card faces should start from 2 to A. Print each card face in its four possible suits: clubs, diamonds, hearts and spades. Use 2 nested **for**-loops and a **switch-case** statement.

| output |
|-------------|
| 2C 2D 2H 2S |
| 3C 3D 3H 3S |
| ... |
| KC KD KH KS |

Problem 5. Calculate $1 + 1!/x + 2!/x^2 + \dots + N!/x^N$

Write a program that, for a given two integer numbers **n** and **x**, calculates the sum $S = 1 + 1!/x + 2!/x^2 + \dots + n!/x^n$. Use only one loop. Print the result with 5 digits after the decimal point.

| n | x | S |
|---|----|---------|
| 3 | 2 | 2.75000 |
| 4 | 3 | 2.07407 |
| 5 | -4 | 0.75781 |

Problem 6. Calculate $N! / K!$

Write a program that calculates $n! / k!$ for given **n** and **k** ($1 < k < n < 100$). Use only one loop. Examples:

| n | k | $n! / k!$ |
|---|---|-----------|
| 5 | 2 | 60 |
| 6 | 5 | 6 |
| 8 | 3 | 6720 |

Problem 7. Calculate $N! / (K! * (N-K)!)$

In combinatorics, the number of ways to choose **k** different members out of a group of **n** different elements (also known as the number of [combinations](#)) is calculated by the following formula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

For example, there are 2598960 ways to withdraw 5 cards out of a standard deck of 52 cards. Your task is to write a program that calculates $n! / (k! * (n-k)!)$ for given **n** and **k** ($1 < k < n < 100$). Try to use only two loops. Examples:

| n | k | $n! / (k! * (n-k)!)$ |
|----|---|----------------------|
| 3 | 2 | 3 |
| 4 | 2 | 6 |
| 10 | 6 | 210 |
| 52 | 5 | 2598960 |

Problem 8. Catalan Numbers

In combinatorics, the [Catalan numbers](#) are calculated by the following formula:

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} = \prod_{k=2}^n \frac{n+k}{k} \quad \text{for } n \geq 0.$$

Write a program to calculate the **nth Catalan number** by given **n** ($1 < n < 100$). Examples:

| n | Catalan(n) |
|----|------------|
| 0 | 1 |
| 5 | 42 |
| 10 | 16796 |

| | |
|----|---------|
| 15 | 9694845 |
|----|---------|

Problem 9. Matrix of Numbers

Write a program that reads from the console a positive integer number **n** ($1 \leq n \leq 20$) and **prints a matrix** like in the examples below. Use two nested loops. Examples:

| n | matrix |
|---|--|
| 2 | 1 2 2 3 |
| n | matrix |
| 3 | 1 2 3 2 3 4 3 4 5 |
| n | matrix |
| 4 | 1 2 3 4 2 3 4 5 3 4 5 6 4 5 6 7 |

Problem 10. Odd and Even Product

You are given **n** integers (given in a single line, separated by a space). Write a program that checks whether the product of the odd elements is equal to the product of the even elements. Elements are counted from **1** to **n**, so the first element is odd, the second is even, etc. Examples:

| numbers | result |
|--------------|---|
| 2 1 1 6 3 | yes product = 6 |
| 3 10 4 6 5 1 | yes product = 60 |
| 4 3 2 5 2 | no odd_product = 16 even_product = 15 |

Problem 11. Random Numbers in Given Range

Write a program that enters 3 integers **n**, **min** and **max** ($\text{min} \leq \text{max}$) and prints **n** random numbers in the range [**min...max**]. Examples:

| n | min | max | random numbers |
|----|-----|-----|-------------------------------|
| 5 | 0 | 1 | 1 0 0 1 1 |
| 10 | 10 | 15 | 12 14 12 15 10 12 14 13 13 11 |

Note that the above output is just an example. Due to randomness, your program most probably will produce different results.

Problem 12.* Randomize the Numbers 1...N

Write a program that enters in integer **n** and prints the numbers 1, 2, ..., **n** in random order. Examples:

| n | randomized numbers 1...n |
|----|--------------------------|
| 3 | 2 1 3 |
| 10 | 3 4 8 2 6 7 9 1 10 5 |

Note that the above output is just an example. Due to randomness, your program most probably will produce different results. You might need to use [arrays](#).

Problem 13. Binary to Decimal Number

Using loops write a program that converts a [binary integer](#) number to its decimal form. The input is entered as **string**. The output should be a variable of type **long**. Do not use the built-in .NET functionality. Examples:

| binary | decimal |
|------------------------------|-----------|
| 0 | 0 |
| 11 | 3 |
| 10101010101011 | 43691 |
| 1110000110000101100101000000 | 236476736 |

Problem 14. Decimal to Binary Number

Using loops write a program that converts an integer number to its [binary representation](#). The input is entered as **long**. The output should be a variable of type **string**. Do not use the built-in .NET functionality. Examples:

| decimal | binary |
|-----------|------------------------------|
| 0 | 0 |
| 3 | 11 |
| 43691 | 10101010101011 |
| 236476736 | 1110000110000101100101000000 |

Problem 15. Hexadecimal to Decimal Number

Using loops write a program that converts a [hexadecimal integer](#) number to its decimal form. The input is entered as **string**. The output should be a variable of type **long**. Do not use the built-in .NET functionality. Examples:

| hexadecimal | decimal |
|-------------|--------------|
| FE | 254 |
| 1AE3 | 6883 |
| 4ED528CBB4 | 338583669684 |

Problem 16. Decimal to Hexadecimal Number

Using loops write a program that converts an integer number to its [hexadecimal representation](#). The input is entered as **long**. The output should be a variable of type **string**. Do not use the built-in .NET functionality. Examples:

| decimal | hexadecimal |
|--------------|-------------|
| 254 | FE |
| 6883 | 1AE3 |
| 338583669684 | 4ED528CBB4 |

Problem 17. * Calculate GCD

Write a program that calculates the [greatest common divisor](#) (GCD) of given two integers **a** and **b**. Use the **Euclidean algorithm** (find it in Internet). Examples:

| a | b | GCD(a, b) |
|----|-----|-----------|
| 3 | 2 | 1 |
| 60 | 40 | 20 |
| 5 | -15 | 5 |

Problem 18.* Trailing Zeroes in N!

Write a program that calculates with how many zeroes the factorial of a given number **n** has at its end. Your program should work well for very big numbers, e.g. $n=100000$. Examples:

| n | trailing zeroes of n! | explanation |
|--------|-----------------------|-----------------------------|
| 10 | 2 | 36288 00 |
| 20 | 4 | 243290200817664 0000 |
| 100000 | 24999 | think why |

Problem 19.** Spiral Matrix

Write a program that reads from the console a positive integer number **n** ($1 \leq n \leq 20$) and **prints a matrix** holding the numbers from **1** to **n*n** in the form of **square spiral** like in the examples below. Examples:

| n | matrix | | | | |
|---|---|---|---|---|---|
| 2 | <table><tr><td>1</td><td>2</td></tr><tr><td>4</td><td>3</td></tr></table> | 1 | 2 | 4 | 3 |
| 1 | 2 | | | | |
| 4 | 3 | | | | |

| n | matrix | | | | | | | | | |
|---|--|---|---|---|---|---|---|---|---|---|
| 3 | <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>8</td><td>9</td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table> | 1 | 2 | 3 | 8 | 9 | 4 | 7 | 6 | 5 |
| 1 | 2 | 3 | | | | | | | | |
| 8 | 9 | 4 | | | | | | | | |
| 7 | 6 | 5 | | | | | | | | |

| n | matrix | | | | | | | | | | | | | | | | |
|----|--|----|---|---|---|----|----|----|---|----|----|----|---|----|---|---|---|
| 4 | <table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>12</td><td>13</td><td>14</td><td>5</td></tr><tr><td>11</td><td>16</td><td>15</td><td>6</td></tr><tr><td>10</td><td>9</td><td>8</td><td>7</td></tr></table> | 1 | 2 | 3 | 4 | 12 | 13 | 14 | 5 | 11 | 16 | 15 | 6 | 10 | 9 | 8 | 7 |
| 1 | 2 | 3 | 4 | | | | | | | | | | | | | | |
| 12 | 13 | 14 | 5 | | | | | | | | | | | | | | |
| 11 | 16 | 15 | 6 | | | | | | | | | | | | | | |
| 10 | 9 | 8 | 7 | | | | | | | | | | | | | | |