Home task № 5.1

Course

Programming Using Java

TOPIC: STRINGS

Task 1

Enter a string of text from the keyboard, followed by one character. Output indexes and the number of matches (look for occurrences of a character in a string) to the console. If no match is found, output the corresponding text.

Task 2

Write a program that creates a string, in which all the integers starting from 1 are written out in one line "123456789101112131415...".

The string should be no longer than 1000 characters.

For the number n (*entered from the keyboard*), output **the digit** on the n position (numbering starts with 1).

Task 3

Calculate the average word length in the sentence entered from the keyboard.

Task 4

Enter a string from the keyboard (Latin). Select all words that start with vowels and end with consonants from the entered string. Output the selected words to the console.

Home task № 5.1

Task 5

Enter a string from the keyboard. The string should contain words that can be separated by spaces or colons. It is necessary to calculate the number of words in the string that have an even number of letters.

Task 6

> In the American army, the number 13 is considered unlucky, and in Chinese army, the number 4 is considered unlucky. Before the joint exercises with the US and Chinese army, the Army Headquarters of the Ukraine decided to exclude the numbers of military equipment containing numbers 4 or 13 (for example, 40123, 13373, 12345 or 61342) in order to not confuse foreign colleagues.

Write a program that will display how many numbers will have to be excluded if the army has 100,000 units of military equipment and each military vehicle has a number from 00000 to 99999. Solve it using strings.

Task 7

The user enters any string from the keyboard. Change all capital letters to lowercase letters and lowercase letter to capital letters in the source string. If the string contains digits, replace them with an underscore and output the result to the console.

Task 8

> In *Java*, the first word in the variable name starts with a lowercase Latin letter, the next word is written with a capital letter (only the first letter of the word is capital), and the words have no separators and consist only of Latin letters. For example, the correct entries for variables in Java may look like this: **javaIdentifier**, **longAndMnemonicIdentifier**, **name**, **nEERC**.
>
> In *C ++*, only small Latin characters and the "_" character, which separates non-empty words from each other, are used to describe variables. If the string has a mixed syntax, for example, **java_Identifier**, report this. Examples: **java_identifier**, **long_and_mnemonic_identifier**, **name**, **n_e_e_r_c**.

You need to write a program that converts a variable written in one language to the format of another language. The variable identifier (name) should be entered from the keyboard. The program should determine, from which language the variable is taken and change it to a variable of another language. Output the result to the console.

Task 9

Write a program that checks whether a string is anagram of another string (the string can consist of several words and punctuation symbols). Spaces and punctuation should be ignored when analyzing. The difference in capital and lowercase letters should be ignored. Both strings should be entered from

the keyboard. The program should display Yes if the strings are anagram and No otherwise.

Example of an anagram:

Creative – Reactive,
Listen – Silent

Task 10

Write a program that will print a rhombus pattern based on a string entered from the keyboard (the maximum length is 50 characters).

Example of the output of the “testing” string:

```
      t
     te
    tes
   test
  testi
 testin
testing
 esting
  sting
   ting
    ing
     ng
      g
```

Home task № 5.1

Task 11

Words in the Mumba-Yumba language can consist only of letters **a**, **b**, **c**, and in this case:

- they never contain two letters **b** in a row,
- a word never contains three identical subwords in a row. For example, according to this rule, the following words cannot be included in the Mumba-Yumba language: **aaa** (since it contains the subword **a** three times in a row), **abababab** (since it contains the subword **ab** three times in a row), **aabcabcabca** (since it contains the subword **abc** three times in a row).

All words that satisfy the above rules are included in the Mumba-Yumba language.

Write a program that determines whether the given word (entered from the keyboard) belongs to this language.

Task 12

Write a program that will count the number of smiles in the given text.

The sequence of characters will be considered a smile if it satisfies the following conditions:

- the first character is either ; (semicolon) or : (colon) exactly once;
- further the - (minus) character can follow it any number of times (including 0 times);
- at the end, there should be some number (at least one) of the same braces from the following set: (,), [,];
- no other characters can be inside the smile.

For example, the following sequences are smiles:

:)

;---------[[[[[[[[

whereas these sequences are not smiles (although some of them contain smiles):

:-)]

;--

-)

::-(

:-()