11308. ICPC strings

Define an ICPC-string as a string composed of the letters *I*, *C*, *P*, *C* such that every 4 consecutive characters contain exactly two letters *C*, one letter *I*, and one letter *P*.

For a given *n*, find the number of distinct ICPC-strings of length *n*.

**Input.** Contains one integer *n* (4 ≤ *n* ≤ 1000) – the length of the string.

**Output.** Print one integer – the number of ICPC-strings of length *n*.

Определим ICPC-строку как строку, составленную из букв *I*, *C*, *P*, *C*, такую,

|  |  |
| --- | --- |
| **Sample input** | **Sample output** |
| 5 | 12 |

## SOLUTION

**combinatorics**

**Algorithm analysis**

The number of distinct anagrams of the word “ICPC” is 4! / (2! \* 2!) = 12.

The fifth character of an ICPC-string is uniquely determined by the first four. Indeed, every 4 consecutive letters must contain two letters C, one letter I, and one letter P. Therefore, the fifth character must coincide with the first one. For the same reason, the sixth character coincides with the second, the seventh with the third, and so on.

Thus, any ICPC-string is completely determined by its first four characters. Consequently, the number of ICPC-strings of length *n* does not depend on *n* and equals the number of anagrams of the word “ICPC”, which is 12.

**Example**

Consider examples of ICPC-strings:

* **ICPC**ICPCICPCICPCICPC…
* **CCPI**CCPICCPICCPICCPI…
* **PCIC**PCICPCICPCICPCIC…

**Algorithm implementation**

To solve the problem, it is sufficient to print the number 12.

printf("12\n");