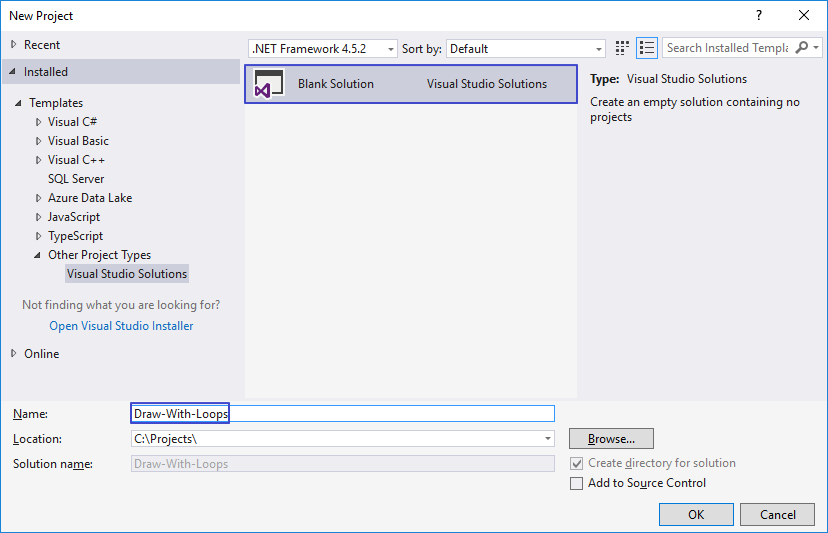
# Упражнения: Чертане с цикли

Задачи за упражнение в клас и за домашно към курса [„Основи на програмирането“ @ СофтУни](https://softuni.bg/trainings/1874/programming-basics-with-javascript-february-2018/internal).

## Празно Visual Studio решение (Blank Solution)

1. Създайте празно решение (**Blank Solution**) във Visual Studio за да организирате кода от задачите за упражнение. Целта на този **blank solution** e да съдържа **по един проект за всяка задача** от упражненията.



1. Задайте **да се стартира по подразбиране текущия проект** (не първият в решението). Кликнете с десен бутон на мишката върху **Solution 'Draw-with-Loops'** 🡪 **[Set StartUp Projects…]** 🡪 **[Current selection]**.

## Правоъгълник от 10 x 10 звездички

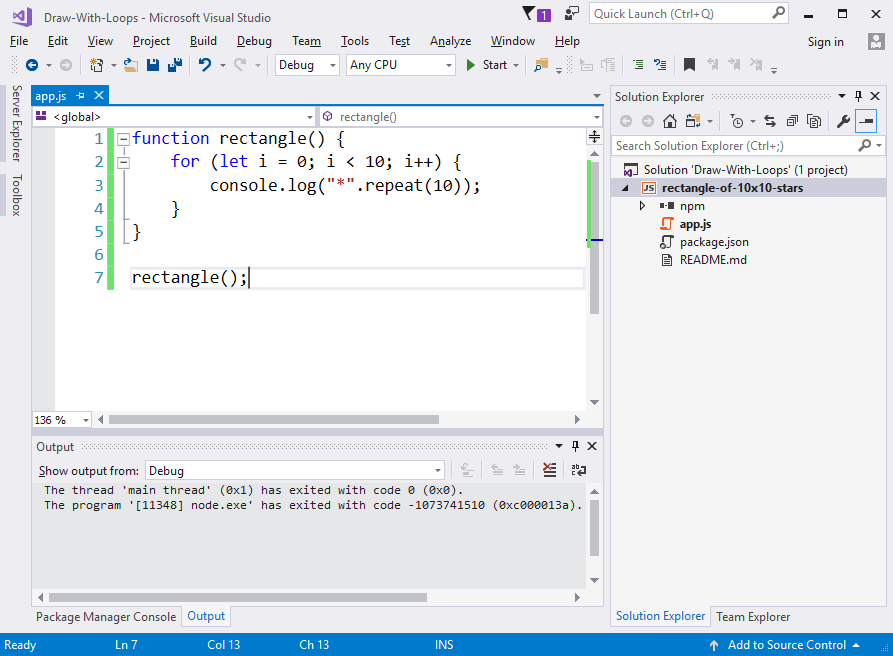
Напишете програма, която чертае на конзолата **правоъгълник от 10 x 10 звездички**:

### Примерен вход и изход:

|  |  |
| --- | --- |
| **вход** | **изход** |
| (няма) | \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\* |

**Подсказки**:

1. Създайте **нов проект** в съществуващото Visual Studio решение – конзолна Node.js програма. Задайте подходящо име на проекта, например “rectangle-of-10x10-stars”.
2. Отидете в тялото на функцията rectangle() и напишете решението на задачата. Можете да си помогнете с кода от картинката по-долу:



1. **Стартирайте** програмата с [F5] и я **тествайте**:



1. **Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#0>. Трябва да получите **100 точки** (напълно коректно решение).

## Правоъгълник от N x N звездички

Напишете програма, която чете цяло положително число n, въведено от потребителя, и печата на конзолата **правоъгълник от** n **\*** n-на брой **звездички**.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | \*\*  \*\* | 3 | \*\*\*  \*\*\*  \*\*\* | 4 | \*\*\*\*  \*\*\*\*  \*\*\*\*  \*\*\*\* |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#1>.

**Подсказка**: отпечатайте n звездички в цикъл n пъти, точно както в предната задача.

## Квадрат от звездички

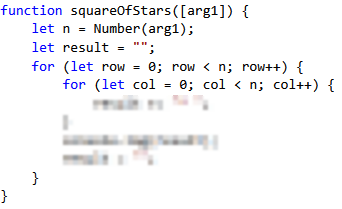
Напишете програма, която чете число n, въведено от потребителя, и чертае **квадрат от** n **\*** n-на брой **звездички**. Разликата с предходната задача е, че между всеки две звездички има по един интервал.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | \* \*  \* \* | 3 | \* \* \*  \* \* \*  \* \* \* | 4 | \* \* \* \*  \* \* \* \*  \* \* \* \*  \* \* \* \* |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#2>.

**Подсказка**: завъртете два вложени цикъла. Може да си помогнете с кода по-долу (част от кода е нарочно замъглен, за да не преписвате механично, а да се замислите малко):



Задачата може да се реши и без вложен цикъл. Проверете метода [**repeat()**](https://www.w3schools.com/jsref/jsref_repeat.asp).

## Триъгълник от долари

Да се напише програма, която чете число n, въведено от потребителя, и печата **триъгълник от долари**.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | $  $ $ | 3 | $  $ $  $ $ $ | 4 | $  $ $  $ $ $  $ $ $ $ | 5 | $  $ $  $ $ $  $ $ $ $  $ $ $ $ $ |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#3>.

## Квадратна рамка

Напишете програма, която чете цяло положително число n, въведено от потребителя,и чертае на конзолата **квадратна рамка** с размер n \* n.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3 | + - +  | - |  + - + | 4 | + - - +  | - - |  | - - |  + - - + | 5 | **+ - - - +**  **| - - - |**  **| - - - |**  **| - - - |**  **+ - - - +** | 6 | + - - - - +  | - - - - |  | - - - - |  | - - - - |  | - - - - |  + - - - - + |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#4>.

**Подсказки**:

* Отпечатайте горната част: знак “+ ”, n-2 пъти знак “- ”, знак “+”.
* Отпечатайте средната част: в цикъл n-2 пъти печатайте знак “| ”, n-2 пъти знак “- ”, знак “|”.
* Отпечатайте долната част: знак “+ ”, n-2 пъти знак “- ”, знак “+”.

## Ромбче от звездички

Напишете програма, която чете цяло положително число n, въведено от потребителя, и печата **ромбче от звездички** с размер n.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 1 | \* | 2 | \*  \* \*  \* | 3 | \*  \* \*  \* \* \*  \* \*  \* | 4 | \*  \* \*  \* \* \*  \* \* \* \*  \* \* \*  \* \*  \* |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#5>.

**Подсказки**:

* Разделете ромба на горна и долна част и ги печатайте с два отделни цикъла.
* За **горната част** завъртете цикъл за row от 1 то n:
  + Отпечатайте n-row интервала.
  + Отпечатайте “\* ” row пъти.
  + Отпечатайте n-row интервала.
* **Долната част** отпечатайте аналогично на горната с цикъл в обратен ред, от n-1 до 1.

## Коледна елха

Напишете програма, която чете число n (1 ≤ n ≤ 100), въведено от потребителя, и печата **коледна елха** с размер n.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 1 | |  \* | \* | 2 | |  \* | \*  \*\* | \*\* | 3 | |  \* | \*  \*\* | \*\*  \*\*\* | \*\*\* | 4 | |  \* | \*  \*\* | \*\*  \*\*\* | \*\*\*  \*\*\*\* | \*\*\*\* |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#6>.

**Подсказки**:

* В цикъл за i от 0 до n печатайте (за лявата част на елхата):
  + n-i интервала; i звездички; вертикална черта.
* Аналогично довършете дясната част на елхата.

## Слънчеви очила

Напишете програма, която чете цяло число n (3 ≤ n ≤ 100), въведено от потребителя, и печата **слънчеви очила** с размер 5\*n x n.

### Примерен вход и изход:

|  |  |
| --- | --- |
| **вход** | **изход** |
| 3 | \*\*\*\*\*\* \*\*\*\*\*\*  \*////\*|||\*////\*  \*\*\*\*\*\* \*\*\*\*\*\* |
| 4 | \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*  \*//////\*||||\*//////\*  \*//////\* \*//////\*  \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* |
| 5 | \*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*  \*////////\* \*////////\*  \*////////\*|||||\*////////\*  \*////////\* \*////////\*  \*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\* |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#7>.

**Подсказки**:

* Отпечатайте **най-горния ред** от очилата:
  + 2\*n звездички; n интервала; 2\*n звездички
* Отпечатайте **средните** **n-2** **реда**:
  + звездичка; 2\*n-2 наклонени черти; звездичка; n интервала; звездичка; 2\*n-2 наклонени черти; звездичка
  + когато редът е (n-1) / 2 *(целочислено деление – Math.floor() или parseInt())*, печатайте n вертикални черти вместо n интервала
* Отпечатайте **най-долния ред** от очилата:
  + 2\*n звездички; n интервала; 2\*n звездички

## Къщичка

Напишете програма, която чете число n (2 ≤ n ≤ 100), въведено от потребителя, и печата **къщичка** с размер n x n.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | \*\*  || | 3 | -\*-  \*\*\*  |\*| | 4 | -\*\*-  \*\*\*\*  |\*\*|  |\*\*| | 5 | --\*--  -\*\*\*-  \*\*\*\*\*  |\*\*\*|  |\*\*\*| | 6 | --\*\*--  -\*\*\*\*-  \*\*\*\*\*\*  |\*\*\*\*|  |\*\*\*\*|  |\*\*\*\*| |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#8>.

**Подсказки**:

* Отпечатайте в цикъл **покрива** на къщичката:
  + Той съдържа (n + 1) / 2 *(целочислено деление - Math.floor() или parseInt())* реда.
  + На първия си ред съдържа 1 звездичка при нечетно n или 2 звездички при четно n.
  + На всеки следващ ред съдържа с 2 звездички повече.
* Отпечатайте в цикъл основата на къщичката: n / 2 - 1 реда *(целочислено деление - Math.floor() или parseInt())*.

## \* Диамант

Напишете програма, която чете цяло число n (1 ≤ n ≤ 100), въведено от потребителя, и печата диамант с размер n.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 1 | \* | 2 | \*\* | 3 | -\*-  \*-\*  -\*- | 4 | -\*\*-  \*--\*  -\*\*- | 5 | --\*--  -\*-\*-  \*---\*  -\*-\*-  --\*-- |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 6 | --\*\*--  -\*--\*-  \*----\*  -\*--\*-  --\*\*-- | 7 | ---\*---  --\*-\*--  -\*---\*-  \*-----\*  -\*---\*-  --\*-\*--  ---\*--- | 8 | ---\*\*---  --\*--\*--  -\*----\*-  \*------\*  -\*----\*-  --\*--\*--  ---\*\*--- | 9 | ----\*----  ---\*-\*---  --\*---\*--  -\*-----\*-  \*-------\*  -\*-----\*-  --\*---\*--  ---\*-\*---  ----\*---- |

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/155#9>.

**Подсказки**:

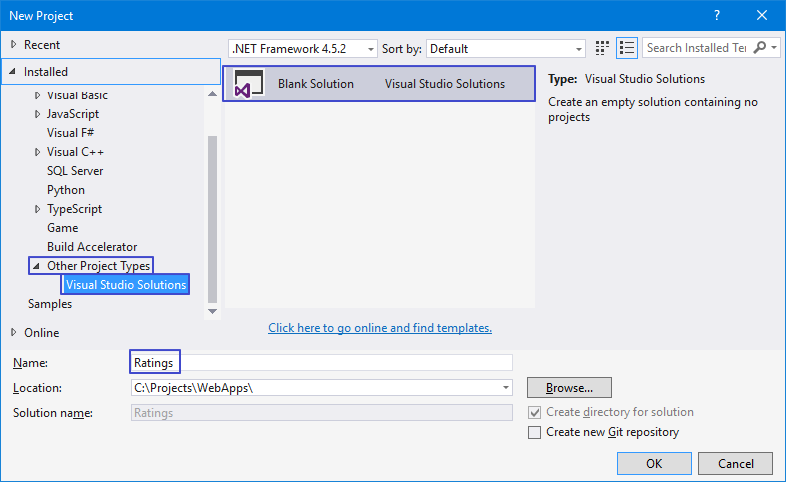
* Всички редове съдържат точно по n символа.
* Първият ред съдържа **отляво и отдясно** точно leftRight = **(**n **- 1) / 2** тирета.
* Всеки следващ ред **до средния** съдържа отляво и отдясно с **1 тире по-малко** от предходния.
* Всеки следващ ред **след средния** съдържа отляво и отдясно с **1 тире повече** от предходния.
* Всеки ред съдържа **в средата** си (във вътрешността на диаманта)mid= n **- 2 \*** leftRight **- 2** тирета.
* Всеки ред съдържа **2 звездички**, освен когато mid е отрицателно (тогава има само 1 звездичка).
* За всеки ред може да се изчислят и отпечатат неговите 5 съставни части:
  + leftRight тиренца отляво
  + **1** звездичка
  + mid тиренца в средата (когато mid >= **0**)
  + **1** звездичка (когато mid >= **0**)
  + **1** звездичка
  + leftRight тиренца отляво

## Рейтинги – визуализация в уеб среда

Да се разработи уеб приложение за визуализация на рейтинг (число от 0 до 100). Чертаят се от 1 до 10 звездички (с половинки). Звездичките да се генерират с for-цикъл:

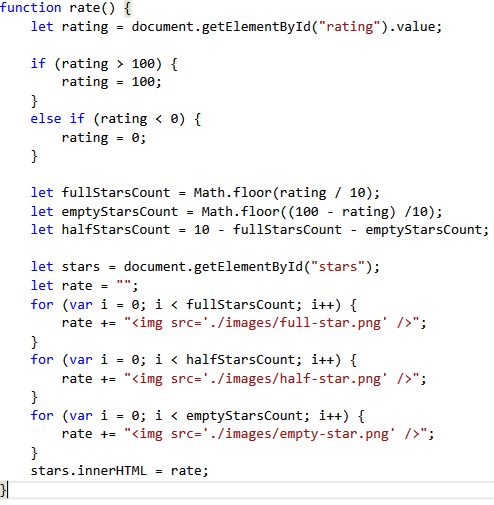


Създайте нов проект във Visual Studio с име **'Ratings':**



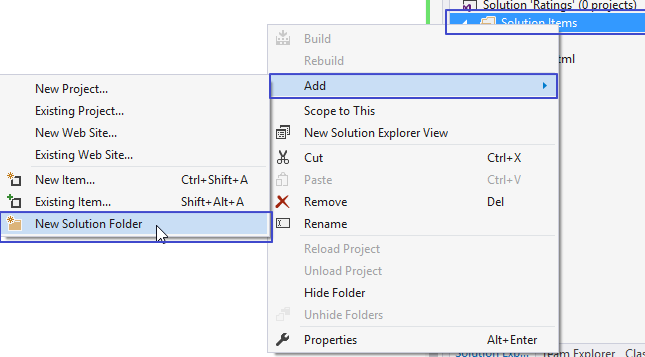


Този код създава уеб форма <form> с едно поле “rating” за въвеждане на число в интервала [0…100] и бутон **[**Rate**]** за изпращане на данните от формата към сървъра. Указан е пътят до скриптовия файл, в който е дефинирана функцията rate() . На бутона [Rate]е закачено събитието onclick, което извиква функцията rate().



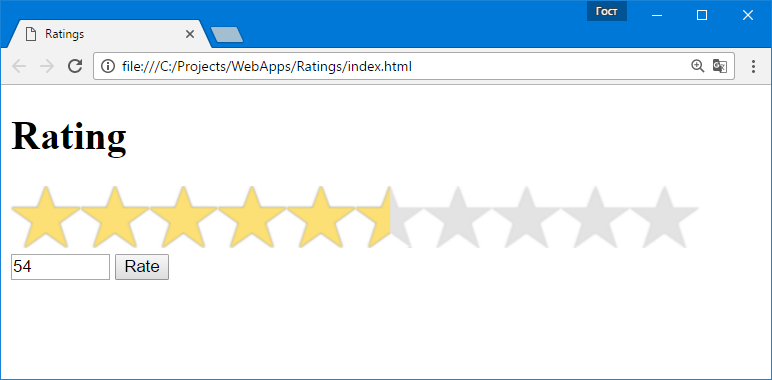
Горният код взима въведеното число rating, прави малко пресмятания и изчислява броя **пълни звездички**, броя **празни звездички** и броя **половинки звездички**, след което генерира **HTML код**, който нарежда няколко картинки със звездички една след друга, за да сглоби от тях картинката с рейтинга. Подготвеният HTML код се вкарва в <div> елемент с **id=”stars”**. Преди да се извършат пресмятанията се правят проверки за долна и горна граница на стойността на рейтинга.

1. От **[Solution Explorer]** направете папка “images” в проекта:



Сега добавете картинките със звездичките (те са част от заданието за домашно). Копирайте ги от Windows Explorer и ги поставете в папката “images” в **[Solution Explorer]** във Visual Studio с **copy / paste**.

1. Тествайте приложението:



# Изпитни задачи от минали издания на курса

## Чертане на крепост

*Пета задача от междинния изпит на 6 март 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/169#4)*.*

Да се напише програма, която прочита от конзолата **цяло число** n, въведено от потребителя, и чертае **крепост** с ширина **2 \*** n **колони** и височина n **реда** като в примерите по-долу. Лявата и дясната колона във вътрешността си са широки **n / 2**.

### Вход

Входът е **цяло число** n в интервала [**3**…**1000**].

### Изход

Да се отпечатат на конзолата nтекстови реда, изобразяващи **крепостта**, точно както в примерите.

### Примерен вход и изход:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3 | /^\/^\  | |  \\_/\\_/ | 4 | /^^\/^^\  | |  | |  \\_\_/\\_\_/ | 5 | /^^\\_\_/^^\  | |  | |  | \_\_ |  \\_\_/ \\_\_/ | 8 | /^^^^\\_\_\_\_/^^^^\  | |  | |  | |  | |  | |  | \_\_\_\_ |  \\_\_\_\_/ \\_\_\_\_/ |

## Пеперуда

*Пета задача от междинния изпит на 26 март 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/179#4)*.*

Да се напише програма, която прочита от конзолата **цяло число** n, въведено от потребителя, и чертае **пеперуда** с ширина **2 \*** n - 1 **колони** и височина **2 \* (**n - 2) + 1 **реда** като в примерите по-долу. **Лявата** и **дясната** ѝ **част** са **широки**

**n - 1**.

### Вход

Входът е **цяло число** n в интервала [**3**…**1000**].

### Изход

Да се отпечатат на конзолата **2 \* (**n - 2) + 1 текстови реда, изобразяващи **пеперудата**.

### Примерен вход и изход:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **вход** | **изход** | **вход** | **изход** | **вход** | **изход** |
| 3 | \*\ /\*  @  \*/ \\* | 5 | \*\*\*\ /\*\*\*  ---\ /---  \*\*\*\ /\*\*\*  @  \*\*\*/ \\*\*\*  ---/ \---  \*\*\*/ \\*\*\* | 7 | \*\*\*\*\*\ /\*\*\*\*\*  -----\ /-----  \*\*\*\*\*\ /\*\*\*\*\*  -----\ /-----  \*\*\*\*\*\ /\*\*\*\*\*  @  \*\*\*\*\*/ \\*\*\*\*\*  -----/ \-----  \*\*\*\*\*/ \\*\*\*\*\*  -----/ \-----  \*\*\*\*\*/ \\*\*\*\*\* |

## Знак СТОП

*Пета задача от междинния изпит на 24 април 2016. Тествайте решението си* [***тук***](https://judge.softuni.bg/Contests/Practice/Index/181#4)*.*

Да се напише програма, която прочита от конзолата **цяло число** N, въведено от потребителя, и чертае **предупредителен знак STOP** с размери като в примерите по-долу.

### Вход

Входът е **цяло число** N в интервала [**3**…**1000**].

### Изход

Да се отпечатат на конзолататекстови редове, изобразяващи **предупредителния знак STOP**, точно както в примерите.

### Примерен вход и изход:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **вход** | **изход** | **вход** | **изход** | **вход** | **изход** |
| 3 | ....\_\_\_\_\_\_\_....  ...//\_\_\_\_\_\\...  ..//\_\_\_\_\_\_\_\\..  .//\_\_\_\_\_\_\_\_\_\\.  //\_\_\_STOP!\_\_\_\\  \\\_\_\_\_\_\_\_\_\_\_\_//  .\\\_\_\_\_\_\_\_\_\_//.  ..\\\_\_\_\_\_\_\_//.. | 6 | .......\_\_\_\_\_\_\_\_\_\_\_\_\_.......  ......//\_\_\_\_\_\_\_\_\_\_\_\\......  .....//\_\_\_\_\_\_\_\_\_\_\_\_\_\\.....  ....//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\....  ...//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\...  ..//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\..  .//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\.  //\_\_\_\_\_\_\_\_\_STOP!\_\_\_\_\_\_\_\_\_\\  \\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//  .\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//.  ..\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//..  ...\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//...  ....\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//....  .....\\\_\_\_\_\_\_\_\_\_\_\_\_\_//..... | 7 | ........\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_........  .......//\_\_\_\_\_\_\_\_\_\_\_\_\_\\.......  ......//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\......  .....//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\.....  ....//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\....  ...//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\...  ..//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\..  .//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\\.  //\_\_\_\_\_\_\_\_\_\_\_STOP!\_\_\_\_\_\_\_\_\_\_\_\\  \\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//  .\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//.  ..\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//..  ...\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//...  ....\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//....  .....\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//.....  ......\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//...... |