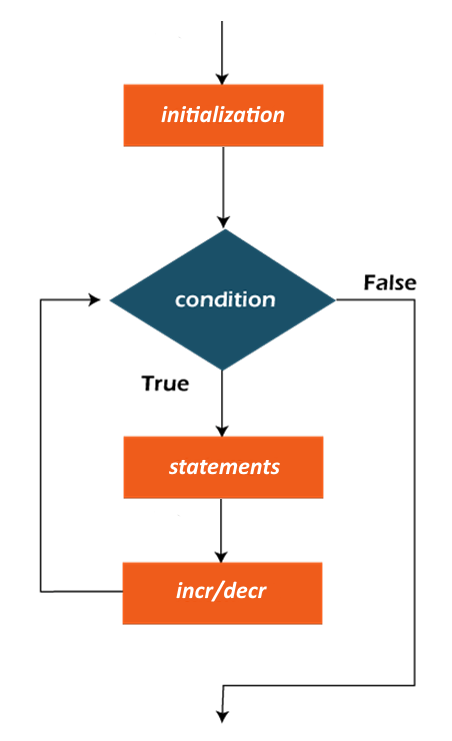
**Lesson 7. For Loop**

Цикл **while** дійсно можна застосовувати практично у будь-яких випадках, де необхідно мати повторюваний алгоритм. Дещо зручнішим для виконання повторень з лічильником є саме цикл **for**, який по суті може робити все те саме, однак має трохи компактніший запис.

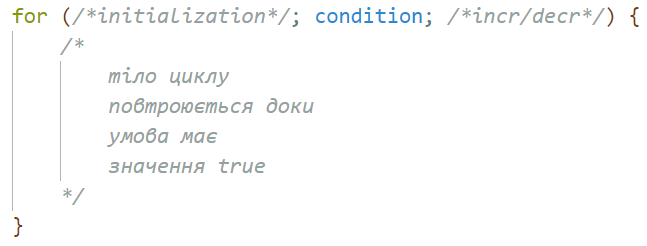
Саме його вивченню та використанню ми присвятимо дане заняття.

**For loop**

Отже, цикл **for** структурно складається із 4 блоків, які визначають як саме він спрацює:

1. **Ініціалізація** – блок, що виконається при вході в цикл незалежно від жодних умов. Зазвичай у цій частині виконують створення змінної-лічильника та присвоєння їй початкового значення.
2. **Умова** – саме від умови залежить продовження виконання циклу, як і у випадку з while (якщо true – продовжуємо, false – вихід).
3. **Тіло циклу** – основний алгоритм, що повинен повторюватись циклом. Записується у фігурних дужках після визначення циклу.
4. **Приріст** – частина, що виконується кожного разу після виконання тіла. Зазвичай у цій частині задається як буде змінюватись (збільшуватись або зменшуватись) значення змінної-лічильника.

Синтаксис запису **for** трохи відрізняється від **while**. Ініціалізація, умова і приріст записуються у круглих дужках після слова **for** розділені крапкою з комою. Далі у фігурних дужках записується тіло циклу.



З попередніх занять ви знаєте, що достатньо часто бувають ситуації, коли значення змінною необхідно певним чином змінити, збільшити чи зменшити. Однак не просто виконати додавання, а саме змінити поточне значення. Одним із варіантів є очевидно запис операції присвоєння у наступному вигляді:

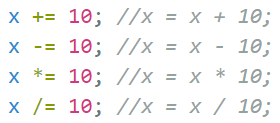
**Operators**



Однак існує також скорочений запис подібних операцій за допомогою операторів наведених у таблиці нижче:

|  |  |
| --- | --- |
| Оператор | Значення |
| += | Збільшити на |
| -= | Зменшити на |
| \*= | Домножити на (збільшити в певну кількість раз) |
| /= | Зменшити в певну кількість раз |

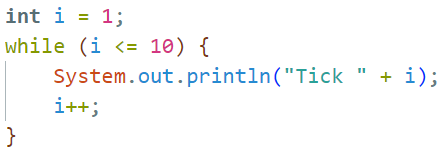
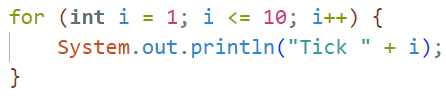
Приклади використання наведено нижче. У коментарях можна побачити ту ж саму дію, але через оператор присвоєння:



Такі оператори часто використовуються і в циклах, для виконання приросту лічильника.

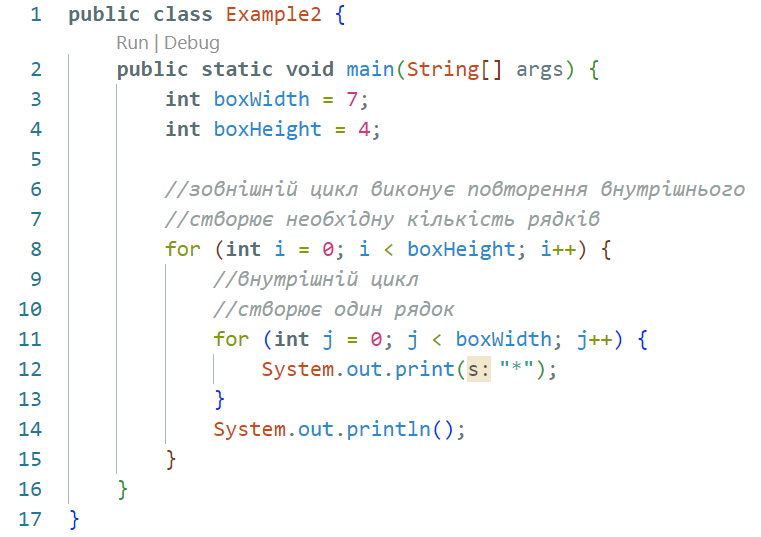
**Examples**

**Приклад 1**. Фрагмент коду на рисунку (ліворуч) нижче виводить у консоль прямий відлік від 1 до 10. Те ж саме виконує і код праворуч із використанням **while**. Порівняйте фрагменти, який вам більш зрозумілий, який подобається більше?



**Приклад 2**. У даному прикладі розглянемо так званий **вкладений цикл**. Якщо говорити простими словами, то це цикл в циклі, тобто цикл, що повторюється певну кількість раз, з допомогою іншого зовнішнього циклу.

Завдання – вивести в консоль прямокутник (бокс) із зірочок заданої ширини та висоти як показано на рисунку ліворуч.



**Tasks**

**Завдання 1**. Створіть програму, що буде отримувати від користувача одне числове значення (int). Програма повинна виводити у консоль усі парні числа від 0 до вказаного значення.

Тестові дані:

*Enter numbers upper bound: 7*

Очікуваний результат:

*0 2 4 6*

**Завдання 2**. Створіть програму, що виведе в консоль трикутник чисел як показано праворуч.

Тестові дані:



Очікуваний результат:



Варіації до завдання:

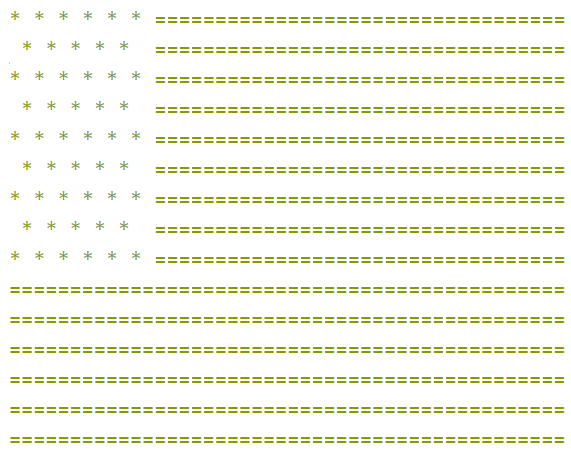
1. Розширте пірамідку відповідно до рисунка нижче.
2. Переверніть зображення пірамідки.



**Завдання 3**. Напишіть програму, що буде виводити на екран символічне зображення американського прапора. У нижній лінії прапора 46 символів «=».

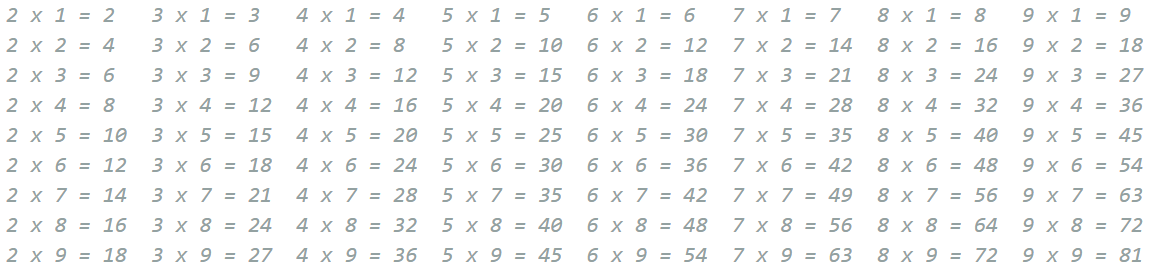
**Homework**

Очікуваний результат:



**Завдання 1**. Створіть програму, що буде виводити у консоль таблицю множення на 2 – 9. Зверніть увагу на вирівнювання стовпців таблиці.

Очікуваний результат:



**Завдання 2**. Додайте у попередню таблицю множення кожного числа на 10. Постарайтесь зберегти вирівнювання стовпців.