# Lab: Nested Loops

Test your tasks in the Judge system: <https://judge.softuni.org/Contests/4416>

## Numbers From N to 1

Write a program that:

* **Reads an integer number** **N** from the console
* **Prints the numbers** from **N to 1**, each on separate line

ANSWER:

int N = int.Parse(Console.ReadLine());

// Print the numbers from N to 1, each on a separate line

for (int i = N; i >= 1; i--)

{

Console.WriteLine(i);

}

### **Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2 | 2  1 | 4 | 4  3  2  1 |

## Even Powers of 2

Write aprogram that:

* Reads an **integer** **number** n from the console
* Prints on the console the number two on **even powers in the range [0; n]**

**2** **≤** **2n**: **20**, **22**, **24**, **26**, …, **2n**.

ANSWER:

int n = int.Parse(Console.ReadLine());

// Print the even powers of 2 in the range [0; n]

for (int i = 0; i <= n; i += 2)

{

// Calculate 2^i

int powerOfTwo = (int)Math.Pow(2, i);

// Print the result

Console.WriteLine($"2^{i} = {powerOfTwo}");

}

### **Example**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 3 | 1  4 | 4 | 1  4  16 | 5 | 1  4  16 | 6 | 1  4  16  64 | 7 | 1  4  16  64 |

## Triangle of Stars

Write a program to print a **triangle of stars** like shown in the examples:

* Read the **size (integer number)** of a triangle from the console
* Print a **triangle of stars**

ANSWER;

int size = int.Parse(Console.ReadLine());

// Print the triangle of stars

for (int i = 1; i <= size; i++)

{

for (int j = 1; j <= i; j++)

{

Console.Write("\*");

}

Console.WriteLine();

}

### **Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\* | 7 | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*\* |

## Building

Write a program to **print a table**, representing a **building**:

* Reads **two integer numbers** from the console: **floors count** and **estates count** per floor
* Identifiers consist of: **{type}{floor}{number}**, e.g. **L65**, **A12**, **O24**
* **Odd** floors hold **apartments** (type **A**), e.g. **A10**, **A11**, **A12**, …
* **Even** floors hold **offices** (type **O**), e.g. **O20**, **O21**, **O22**, …
* The **last floor** holds large apartments (type **L**), e.g. **L60**, **L61**, **L62**

**int estatesPerFloor = int.Parse(Console.ReadLine());**

int floorsCount = int.Parse(Console.ReadLine());

int estatesPerFloor = int.Parse(Console.ReadLine());

// Loop through each floor

for (int floor = floorsCount; floor >= 1; floor--)

{

// Determine the type of estate based on the floor number

char type;

if (floor == floorsCount)

{

type = 'L'; // Last floor holds large apartments

}

else if (floor % 2 == 0)

{

type = 'O'; // Even floors hold offices

}

else

{

type = 'A'; // Odd floors hold apartments

}

// Loop through each estate on the current floor

for (int number = 0; number < estatesPerFloor; number++)

{

// Generate the identifier

string identifier = $"{type}{floor}{number}";

// Print the identifier

Console.Write($"{identifier} ");

}

// Print a new line after each floor

Console.WriteLine();

}

### **Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 6  4 | L60 L61 L62 L63  A50 A51 A52 A53  O40 O41 O42 O43  A30 A31 A32 A33  O20 O21 O22 O23  A10 A11 A12 A13 | 5  3 | L50 L51 L52  O40 O41 O42  A30 A31 A32  O20 O21 O22  A10 A11 A12 |

## Number Pyramid

Write a program that:

* Reads an integer number **n** from the console
* Prints a **pyramid of numbers** as shown in the examples

ANSWER:

int n = int.Parse(Console.ReadLine());

// Initialize a counter for the current number to print

int currentNumber = 1;

// Loop through each level of the pyramid

for (int row = 1; currentNumber <= n; row++)

{

// Loop through each position in the current row

for (int col = 1; col <= row && currentNumber <= n; col++)

{

// Print the current number and increment the counter

Console.Write(currentNumber + " ");

currentNumber++;

}

// Move to the next line after each row

Console.WriteLine();

}

### **Example**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 7 | 1  2 3  4 5 6  7 | 10 | 1  2 3  4 5 6  7 8 9 10 | 12 | 1  2 3  4 5 6  7 8 9 10  11 12 | 15 | 1  2 3  4 5 6  7 8 9 10  11 12 13 14 15 |

## Travel Savings

Write a program that calculate the **money collection** for multiple travel destinations:

* Read a **destination (string)** and **needed budget (floating-point number)** for the destination
* Read many times amounts of collected money, until they are **enough** for the destination (starting from 0)
  + Print:

"**Collected: {sum}**" where sum is formatted to 2nd digit

or

"**Going to {destination}!**"

* Read another destination and budget and collect money again
* A destination "**End**" ends the program

answer:

while (true)

{

string destination = Console.ReadLine();

if (destination.ToLower() == "end")

{

break;

}

// Read the needed budget

float budget = float.Parse(Console.ReadLine());

// Initialize collected amount

float collectedAmount = 0.0f;

// Collect money until the budget is met

while (collectedAmount < budget)

{

float amount = float.Parse(Console.ReadLine());

collectedAmount += amount;

Console.WriteLine($"Collected: {collectedAmount:F2}");

}

// Announce that we're going to the destination

Console.WriteLine($"Going to {destination}!");

}

### **Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| Bali  3500  800  1800  1000  Brazil  4600  5000  End | Collected: 800.00  Collected: 2600.00  Collected: 3600.00  Going to Bali!  Collected: 5000.00  Going to Brazil! | Spain  4000  1000  1500  1500  Greece  800  400  500  End | Collected: 1000.00  Collected: 2500.00  Collected: 4000.00  Going to Spain!  Collected: 400.00  Collected: 900.00  Going to Greece! |

## Sum of Digits Calculator

Write a program that:

* Continuously **read integers** until "**End**" is entered from the console
  + Print the **sum of digits** for each integer, use the following format:

"**Sum of digits = {sum}**"

* Finally, print "**Goodbye**"

answer:

while (true)

{

string input = Console.ReadLine();

// Check if the input is "End"

if (input.ToLower() == "end")

{

break;

}

// Convert the input to an integer

if (int.TryParse(input, out int number))

{

// Calculate the sum of the digits

int sumOfDigits = CalculateSumOfDigits(number);

// Print the sum of the digits

Console.WriteLine($"Sum of digits = {sumOfDigits}");

}

else

{

Console.WriteLine("Invalid input. Please enter an integer or 'End' to stop.");

}

}

// Print "Goodbye"

Console.WriteLine("Goodbye");

// Method to calculate the sum of digits of a number

static int CalculateSumOfDigits(int number)

{

int sum = 0;

while (number != 0)

{

sum += Math.Abs(number % 10); // Use Math.Abs to handle negative digits

number /= 10;

}

return sum;

}

### **Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 157  99  5  438  End | Sum of digits = 13  Sum of digits = 18  Sum of digits = 5  Sum of digits = 15  Goodbye | 107  345  98  23  End | Sum of digits = 8  Sum of digits = 12  Sum of digits = 17  Sum of digits = 5  Goodbye |

## Prime Numbers

Write a program that**:**

* Reads two integer numbers: **start of the range** and **end of the range**
* Print **all prime numbers** in given range

**Hint:** A prime number is a positive integer greater than 1 that has exactly two distinct positive divisors: 1 and itself.

Answer:

int start = int.Parse(Console.ReadLine());

int end = int.Parse(Console.ReadLine());

for (int i = start; i <= end; i++)

{

if (IsPrime(i))

{

Console.Write(i+" ");

}

}

// Function to check if a number is prime

static bool IsPrime(int number)

{

if (number <= 1)

{

return false;

}

if (number == 2)

{

return true;

}

if (number % 2 == 0)

{

return false;

}

int boundary = (int)Math.Floor(Math.Sqrt(number));

for (int i = 3; i <= boundary; i += 2)

{

if (number % i == 0)

{

return false;

}

}

return true;

}

### **Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  50 | 5 7 11 13 17 19 23 29 31 37 41 43 47 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20  30 | 23 29 |