

# Lab: Data Types and Variables

Problems for exercises and homework for the [“Programming Fundamentals” course @ SoftUni](#).

You can check your solutions here: <https://judge.softuni.bg/Contests/171/Data-Types-and-Variables-Lab>.

## I. Integer and Real Numbers

### 1. Centuries to Minutes

Write program to enter an integer number of **centuries** and convert it to **years**, **days**, **hours** and **minutes**.

#### Examples

Input	Output
1	1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes
5	5 centuries = 500 years = 182621 days = 4382904 hours = 262974240 minutes

#### Hints

- Use appropriate data type to fit the result after each data conversion.
- Assume that a year has 365.2422 days at average ([the Tropical year](#)).

#### Solution

You might help yourself with the code below:

```
Console.Write("Centuries = ");
int centuries = int.Parse(Console.ReadLine());
int years = centuries * 100;
int days = (int)(years * 365.2422);
int hours = 24 * days;
int minutes = 60 * hours;
Console.WriteLine("{0} centuries = {1} years = {2} days = {3} hours = {4} minutes", centuries, years, days, hours, minutes);
```

### 2. Circle Area (12 Digits Precision)

Write program to enter a radius **r** (real number) and **print the area** of the circle with exactly **12 digits** after the decimal point. Use data type of **enough precision** to hold the results.

#### Examples

Input	Output	Input	Output
2.5	19.634954084936	1.2	4.523893421169

#### Hints

- You might use the data type **double**. It has precision of 15-16 digits.
- To print the output with exactly 12 digits after the decimal point, you might use the following code:

```
double r = double.Parse(Console.ReadLine());
Console.WriteLine("{0:f12}", Math.PI * r * r);
```

### 3. Exact Sum of Real Numbers

Write program to enter **n** numbers and calculate and print their **exact sum** (without rounding).

#### Examples

Input	Output	Input	Output
3 10000000000000000000 5 10	1000000000000000015	2 0.00000000003 33333333333.3	33333333333.30000000003

#### Hints

- If you use types like **float** or **double**, the result will lose some of its precision. Also it might be printed in scientific notation.
- You might use the **decimal** data type which holds real numbers with high precision with less loss.
- Note that **decimal** numbers sometimes hold the unneeded zeroes after the decimal point, so **0m** is different than **0.0m** and **0.00000m**.

## II. Data Types and Type Conversion

### 4. Elevator

Calculate how many courses will be needed to **elevate n persons** by using an elevator of **capacity of p persons**. The input holds two lines: the **number of people n** and the **capacity p** of the elevator.

#### Examples

Input	Output	Comments
17 3	6	5 courses * 3 people + 1 course * 2 persons
4 5	1	All the persons fit inside in the elevator. Only one course is needed.
10 5	2	2 courses * 5 people

#### Hints

- You should **divide n by p**. This gives you the number of full courses (e.g.  $17 / 3 = 5$ ).
- If **n** does not divide **p** without a remainder, you will need one additional partially full course (e.g.  $17 \% 3 = 2$ ).
- Another approach is to round up **n / p** to the nearest integer (ceiling), e.g.  $17/3 = 5.67 \rightarrow$  rounds up to 6.
- Sample code for the round-up calculation:

```
int courses = (int)Math.Ceiling((double)n / p);
```

## 5. Special Numbers

A **number** is **special** when its **sum of digits** is **5, 7 or 11**.

Write a program to read an integer **n** and for all numbers in the range **1...n** to print the number and if it is special or not (**True / False**).

### Examples

Input	Output
15	1 -> False
	2 -> False
	3 -> False
	4 -> False
	5 -> True
	6 -> False
	7 -> True
	8 -> False
	9 -> False
	10 -> False
	11 -> False
	12 -> False
	13 -> False
	14 -> True
	15 -> False

### Hints

To calculate the sum of digits of given number **num**, you might repeat the following: sum the last digit (**num % 10**) and remove it (**sum = sum / 10**) until **num** reaches **0**.

## 6. Triples of Latin Letters

Write a program to read an integer **n** and print all **triples** of the first **n** **small Latin letters**, ordered alphabetically:

### Examples

Input	Output
3	aaa
	aab
	aac
	aba
	abb
	abc
	aca
	acb
	acc
	baa
	bab
	bac
	bba
	bbb
	bbc
	bca

	bcb
	bcc
	caa
	cab
	cac
	cba
	cbb
	cbc
	cca
	ccb
	ccc

## Hints

Perform 3 nested loops from **0** to **n-1**. For each number **num** print its corresponding Latin letter as follows:

```
char letter = (char)('a' + num);
```

## 7. Greeting

Write a program that enters **first name**, **last name** and **age** and prints "**Hello, <first name> <last name>. You are <age> years old.**". Use interpolated strings.

## Examples

Input	Output
Svetlin Nakov 25	Hello, Svetlin Nakov. You are 25 years old.

## Hints

You might use the following code:

```
Console.WriteLine(
    $"Hello, {firstName} {lastName}.\r\nYou are {age} years old.");
```

# III. Variables

## 8. Refactor Volume of Pyramid

You are given a **working code** that finds the **volume of a pyramid**. However, you should consider that the variables exceed their optimum span and have improper naming. Also, search for variables that **have multiple purpose**.

## Code

### Sample Code

```
double dul, sh, V = 0;
Console.Write("Length: ");
dul = double.Parse(Console.ReadLine());
Console.Write("Width: ");
sh = double.Parse(Console.ReadLine());
Console.Write("Height: ");
V = double.Parse(Console.ReadLine());
V = (dul + sh + V) / 3;
Console.WriteLine("Pyramid Volume: {0:F2}", V);
```

## Hints

- **Reduce the span** of the variables by declaring them in the moment they receive a value, not before
- Rename your variables to **represent their real purpose** (example: "dul" should become length, etc.)
- Search for variables that have multiple purpose. If you find any, **introduce a new variable**.

## 9. Refactor Special Numbers

You are given a **working code** that is a solution to **Problem 5. Special Numbers**. However, the variables are **improperly named, declared before** they are needed and some of them are used for multiple things. Without using your previous solution, **modify the code** so that it is **easy to read and understand**.

## Code

### Sample Code

```
int kolkko = int.Parse(Console.ReadLine());
int obshto = 0; int takova = 0; bool toe = false;
for (int ch = 1; ch <= kolkko; ch++)
{
    takova = ch;
    while (ch > 0)
    {
        obshto += ch % 10;
        ch = ch / 10;
    }
    toe = (obshto == 5) || (obshto == 7) || (obshto == 11);
    Console.WriteLine($"{takova} -> {toe}");
    obshto = 0;
    ch = takova;
}
```

## Hints

- Reduce the span of the variables by declaring them in the moment they receive a value, not before
- Rename your variables to represent their real purpose (example: "dul" should become length, etc.)
- Search for variables that have multiple purpose. If you find any, introduce a new variable