



# C++ code kata: Week 2

---

## Numeric literals and types


---

Language C++17

Compiler G++7

Track Beginner

### Contents

- [Contents](#)
- [Kata this week](#)
  - [Numeric literals in C++](#)
  - [Numeric types in C++](#)
  - [Assignment operator](#)
- [Ground Statement](#)
  - [Statement 1](#)
  - [Statement 2](#)
  - [Statement 3](#)
  - [Statement 4](#)
- [Hints and Study Materials to solve the Katas](#)
- [Tools](#)
- [I need Help](#) 

Hello 🤝

Great to see you again 👍. This week we are going to practice following elements of C++ in our code kata 📖

### Kata this week

This week we are going to practice with various numeric types in C++. we will try to understand how much memory storage a numeric literal requires and what various **datatypes** C++ provides to store them.

#### Numeric literals in C++

C++ defines the following numeric literals:-

Number	Type	Comment
10	Integer	Numeric values without fractional part are integers.
-10	Integer	Integers can be positive or negative.

Number	Type	Comment
0.5	Floating Point	Numeric value with the fractional part is floating point. These can be single precision or double precision values.
2.96E-2	Floating Point	Floating point values in exponential notation. $2.96 / 100 = 0.0296$

Numeric literals can be written in binary, decimal, hexadecimal and octal notation.

## Numeric types in C++

**Typed** language like C++ has concept of datatype. Datatype helps in allocating bytes of memory so that values can be stored in them. The **most commonly type** are presented in the table below ( For much detailed description visit [Hints](#) section ):-

Data type	Memory Usage
__int8	1 byte
__int16, short, unsigned short	2 bytes
float, __int32, int, unsigned int, long, unsigned long	4 bytes.
double, __int64, long double, long long	8 bytes.

## Assignment operator

The assignment operator is basically part of the assignment statement. Its job is to update the contents of the memory location.

It is written as:-

*memory\_location = expression;*

In the above statement:-

- **memory location** is the unique location where value gets updated. It is mostly referred to using an identifier known as a variable. Also known as **lvalue**, this should be assignable. The important thing here is to choose the amount of memory we want to allocate.
- **expression** is any combination of operators and operands that produces a valid result, such that it can be assigned to **lvalue**. This is also known as **rvalue**. On our case, it will be numeric literals for now.
- **=** is assignment operator whose job is to update **lvalue**.

**i** Check links giving you information on what is variable and how to assign value to a variable in [hints](#) section.

## Ground Statement

## Statement 1

Suppose you have an integer variable that stores value for one billion and displays it onto the screen. See below 

```
1: #include <iostream>
2:
3: int main (int argc, char * argv[])
4: {
5:     int one_billion = 1000000000;
6:     std::cout << "One billion: " << one_billion << std::endl;
7:
8:     return EXIT_SUCCESS;
9: }
```


```
One billion: 1000000000
```

What will be the output if you change **line 05** like this? Please support your answer with an explanation.

```
5:     int three_billion = 3000000000;
```

If we have to store value **3,000,000,000** into memory what size of an integer will be required?

## Statement 2

The **oldest human** as per records lived up to the age of 122. Considering that fact, what is the problem with **line 05** in code written below 

```
1: #include <iostream>
2:
3: int main (int argc, char * argv[])
4: {
5:     int age = 21;
6:     std::cout << age << std::endl;
7:     return EXIT_SUCCESS;
8: }
```

## Statement 3

Revisiting our first problem, someone changed the code like this 

```
1: #include <iostream>
2:
3: int main (int argc, char * argv[])
4: {
5:     unsigned three_billion = 3000000000;
6:     std::cout << "Three billion: " << three_billion << std::endl;
7:
8:     return EXIT_SUCCESS;
9: }
```

Three billion: 3000000000

Why it is coming correct. Try to support your answer with an explanation.

## Statement 4

Area of a circle is can be computed as follows. See below 

```
01: #define _USE_MATH_DEFINES
02:
03: #include <iostream>
06: #include <cmath>
07:
08: int main (int argc, char *argv[])
09: {
10:     area = M_PI * radius * radius;
11:     return EXIT_SUCCESS;
12: }
```

What **datatype** you will prefer to store the value of **radius** and **area**?

## Hints and Study Materials to solve the Katas

- [Know more about C++ data types](#)
- [How to write a variable in C++](#)

---

## Tools

---

To write solutions you can download tools from:

Tool	Usage	Download Location
Visual Studio Code	Lightweight program editor.	<a href="#">Download</a>

Tool	Usage	Download Location
GCC (Linux)	C / C++ Compiler.	<a href="#">Download</a>
MingGW (Windows)	C / C++ Compiler.	<a href="#">Download</a>
XCode toolkit (MacOS)	IDE with multi language support.	<a href="#">Download</a>
Catch2	C++ Test framework	<a href="#">Download</a>

## Sensai Says

***"What you learn is not what you read or listened to, but rather what you attempted at..."***

### Progressive learning

If you feel the exercise a little bit difficult to solve do not get disheartened. The whole idea behind these programming exercises is not to solve them but rather attempt them.

Try to attempt them in as many ways as possible, you will learn new techniques that will be very helpful to you in the long run especially in the work field.

We will present you with **solution mail/document** also. The solution will show you various ways to solve a problem and why a technique is better than the last one.

We encourage you to make notes from the solution provided and try to apply what you have learned in future exercise.

### What I will gain from these exercises ?

1. A better and faster way to solve exercise.
2. Reusable components like containers, algorithms, etc. that you can apply to the problem at hand rather than designing your own.
3. Confidence and attitude to solve a problem in new ways, instead of trying monotonous techniques.
4. Writing the robust and quality software using [test driven development](#).

## I need Help

Still, have some questions related to this exercise. If you still have questions feel free to connect. We will be happy to answer your questions.

