Type Casting

https://csci-1301.github.io/about#authors

May 24, 2021 (06:20:46 PM)

Contents

1 Numerical Datatypes

For this part, it is recommended to have the datatypes cheatsheet, which is available in the Week 4 module of the lecture notes. Note that it contains numerous references at its end. You are encouraged to open those links, if you have not already, to have a look at the official documentation, which should not scare you.

1.1 Literals and Variables

This part should be first carried out without using VS.

Assume we have the following statements:

```
int a = 21, b = 4;
float f = 2.5000000f;
double d = -1.3;
decimal m = 2.5m;
```

Answer the following:

- How many variables are declared?
- What are their datatypes?
- What are their values?
- What are their names?

1.2 Operations

- Consider the following expressions. For each of them, tell if they are legal and if so, give the result and its corresponding datatype. The first two are given as examples:

Operation	Legal?	Result	Datatype
a + d	Yes	19.7	double
m + f	No	—	—
a / b	—	—	—
b * f	—	—	—
d + f	—	—	—
d + b	—	—	—
a + m	—	—	—
f / m	—	—	—
d * m	—	—	—

You can check your answers using VS: create a new project, copy the variable declarations and assignments, and write your own statements to perform the calculations in the `Main` method. For instance, if you want to check that the result of `a + d` is of type `double`, write something like:

```
double tempVariable1 = a + d;
Console.WriteLine($"The value of d+f is {tempVariable1}");
int tempVariable2 = a + d; // This line should give you an error.
```

1.3 Cast Operator

Create a new project, and then do the following.

1. Add in your program the following:

   ```
   float floatVar = 4.3f;
   int intVar = floatVar; // This statement will give you an error
   ```

   You will get an error that reads

   ```
   Cannot implicitly convert type 'float' to 'int'. An explicit conversion exists (are you missing a c
   ```

   Can you explain it?

2. VS is suggesting that we use a "cast" to "force" C# to store the value of the variable `floatVar` into the variable `intVar`. To do so, replace the previous statement with the following:

   ```
   int intVar = (int)floatVar; // This statement will compile
   ```

3. Using a `Console.WriteLine` statement, observe the value stored in `intVar`. Can you tell if the value stored in `floatVar` was rounded or truncated before being stored in the variable `intVar`? Conduct further experiments if needed to answer this question.