Assigning Values to Variables

This lesson teaches us about variables, the assignment operator and how the assignment operator can be used to assign values to variables.

WE'LL COVER THE FOLLOWING

Assignment and order of evaluation

The first two difficulties that most people face when learning to program involve the *assignment operation* and *the order of evaluation* in an expression.

The assignment operation

You will see lines similar to the following in almost every program in almost every programming language:

```
a = 10;
```

The meaning of this line is “make `a` hold the value 10.” Similarly, the following line means “make `b` hold the value 20”:

```
b = 20;
```

Based on that information, what can be said about the following line?

```
a = b;
```

Unfortunately, that line is not about the equality concept of mathematics that we all know. The expression above does not mean "`a` is equal to `b`!" When we apply the same logic from the earlier two lines, the expression above must mean “make `a` hold the same value as `b`.”

The well-known equals-to (=) symbol of mathematics has a completely different meaning in programming: make the left side’s value the same as the right side’s value.

Order of evaluation

In general, the operations of a program are applied step by step in the order that they appear in the program. There are exceptions to this rule, which we will see in later chapters. We may see the previous three expressions in a program in the following order:

```
a = 10;
b = 20;
a = b;
```

The meaning of those three lines altogether is this: “make `a`'s value become 10, then make `b`'s value become 20, then make `a`'s value become the same as `b`'s value.”

Accordingly, after those three operations are performed, the value of both `a` and `b` should be 20. The value of a variable is the last value that has been assigned to that variable.

Variables

Concrete, real-world concepts are represented in a program as *variables*. A value like air, temperature, and more complicated objects, such as a car engine, can be made variables in a program.

Why is a variable needed?

The main purpose of a variable is to represent a value in the program. Since every value is of a certain type, every variable is of a certain type as well. Most variables have names, but some variables are anonymous.

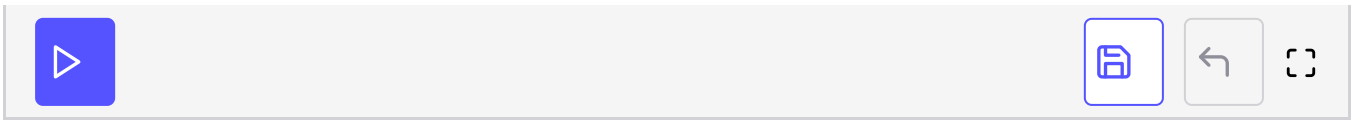
Example

As an example of a variable, we can think of the number of students at a school. Since the number of students is a whole number, `int` is a suitable type, and `studentCount` would be a sufficiently descriptive name.
According to D's syntax rules, a variable is introduced by its type, followed by its name. The introduction of a variable to the program is called its **definition**. Once a variable is defined, its name can be used to access the value that it holds.

```
import std.stdio;

void main() {
    // The definition of the variable; this definition
    // specifies that the type of studentCount is int:
    int studentCount;

    // The name of the variable represents its value:
    writeln("Now there are ", studentCount, " students.");
}
```



Variable use

The above program gives the following output:

```
There are 0 students.
```

We can see here that the value of `studentCount` is **0**. This is according to the fundamental types table from the previous lesson: the initial value of an `int` variable is **0**.

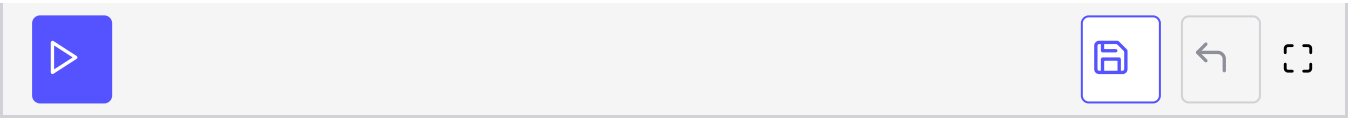
> **Note**: `studentCount` does not appear in the output as its name. In other words, the output of the program is not "*There are studentCount students.*"

The values of variables can be modified by the `=` operator. The `=` operator assigns new values to variables, and for that reason its is called the **assignment operator**:

```
import std.stdio;

void main() {
    int studentCount;
    writeln("There are ", studentCount, " students.");

    // Assigning the value 200 to the studentCount variable:
    studentCount = 200;
    writeln("There are now ", studentCount, " students.");
}
```

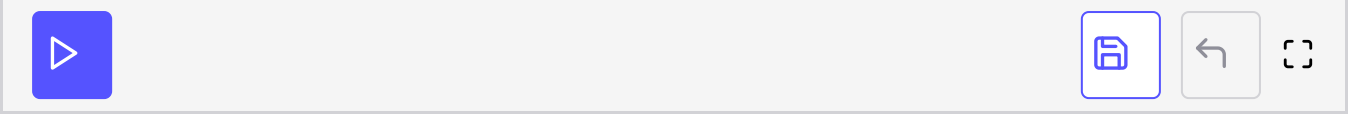


Assignment operator

When the value of a variable is known at the time of its definition, then the variable can be defined and assigned at the same time. This is an important guideline; rarely, a variable is used before being assigned a value:

```
import std.stdio;

void main() {
    // Definition and assignment at the same time:
    int studentCount = 100;
    writeln("There are ", studentCount, " students.");
}
```



Variable declaration and initialization

In the next lesson, you will learn how to use the keywords `auto` and `typeof`.