Get a random value from an array

The kingdom of stupidity

Purpose

In order to get you thinking about problem solving, we’ll show you an algorithm we found more than one university.

The code wasn’t written by students. To our great surprise, it was handed out by the programming teachers.

We hope to make you think about how to solve problems, by thinking a little harder about the problem at hand.

At least harder than the original authors...

We’ll use PHP for the example

The original code was written in Java, but PHP works to show the problem too.

Since you know a little PHP, we’ve translated the code for you.

The problem

Given an array with the following values:

```
$numbers = array(1,3,6,11,20,22,55,57,88,99);
```

Write code which finds a random value from the array.

How would you solve this?

Can you describe your solution, before writing it?

Hint: You can generate a random number between MIN and MAX like this:

```
$randomInt = random_int($MIN,$MAX);
```

Solution using a loop - What’s problematic?

This is the actual algorithm used in the code we found (ported to PHP):

```php
<?php
$numbers = array(1,3,6,11,20,22,55,57,88,99);
$wantedNumber=random_int(1,99);
print "Looking for $wantedNumber\n";
$index = array_search($wantedNumber, $numbers);

while (! $index) {
    print "$wantedNumber not found\n";
    $wantedNumber=random_int(1,99);
    print "Trying with $wantedNumber instead...";
    $index = array_search($wantedNumber, $numbers);
}
print "Found $wantedNumber at: $index\n";
?>
```

The problem

The problem was to find one randomly selected number from the numbers in the array.

The numbers in the array were 10 numbers distributed between 1 and 99 (almost 100 possible values).

The algorithm generates a random number between 1 and 99.

What is the probability for us to directly find a number which is a member of the array?

What could go wrong?

Since the probability to find one of the numbers in the array is roughly 10%, the loop will in average loop 10 times before the random function produces a valid number.

Every number it generate is used in search_array which means some iterations (granted, in Java, the code used binary search, which is rather efficient, but still, searching for a number in an array requires iterations and CPU).

What if we have a crappy random_int method? There’s no telling how many iterations the loop will use…

What could we do instead?

We could think about the problem in a different way.

The task is to randomly select a value from an array.

How do we get a value from an array? We index the array:

```
$value = $numbers[$someIndex];
```

What are the legal indices in any array?

0..size-1

So, we should generate a random index between 0 and size-1!

Alternative algorithm, solving the problem

```
$wantedNumber = $numbers[random_int(0, count($numbers) - 1)];
```

The above solution follows the problem description:

“Get a random value from the array numbers”

“Get a value from the array” translates to `$numbers[someIndex];`

So “Get a random value from the array” must translate to:

```
$numbers[random_int(0, count($numbers) - 1)];
```

Since the index must be between 0 and count($numbers) -1.

Which code is easier to read, debug, understand?

Contestant 1:

```
$wantedNumber = random_int(1,99);
$index = array_search($wantedNumber, $numbers);
while (! $index) {
    $wantedNumber=random_int(1,99);
    $index = array_search($wantedNumber, $numbers);
}
```

Contestant 2:

```
$wantedNumber = $numbers[random_int(0, count($numbers) - 1)];
```