How to insert a hash in another hash in Perl

hash

The other day I wrote on how to insert an array in another array, and one of the readers asked how would that look like for hashes.

Hashes don't have an order and `splice` does not work on hashes so how can we insert one into another?

Merge two hashes

On of the ways is merging the two hashes. In the new hash we will have all the key-value pairs of both of the original hashes. If the same key appears in both hashes, then the latter will overwrite the former, meaning that the value of the former will disappear. (See the key "Foo" in our example.)

examples/insert_hash_in_hash.pl

```
use strict;
use warnings;
use 5.010;
use Data::Dumper;

my %team_a = (
    Foo => 3,
    Bar => 7,
    Baz => 9,
);

my %team_b = (
    Moo => 10,
    Boo => 20,
    Foo => 30,
);


my %team_x = (%team_a, %team_b);

print Dumper \%team_x;
```

Running this script will result in the following output (except that the display order might be different):

```
$VAR1 = {
          'Moo' => 10,
          'Boo' => 20,
          'Foo' => 30,
          'Baz' => 9,
          'Bar' => 7
        };
```

What happens here is that putting the hashes in a list, puts them in LIST context in which case they are flattened to (key, value, key, value, ...). We put two hashes in list context which means they were converted to

The first hash turned into this list: `'Foo', 3, 'Bar', 7, 'Baz', 9` the second hash turned into this list: `'Moo', 10, 'Boo', 20, 'Foo', 30` and together they turned into this list:

`'Foo', 3, 'Bar', 7, 'Baz', 9, 'Moo', 10, 'Boo', 20, 'Foo', 30`

During the assignment to `%team_x`, perl built the hash by converting the list into key-value pairs. So it added the Foo => 3, then the Bar => 7 then Baz => 9, Moo => 10, Boo => 20 and finally Foo => 30 again. This time , it has overwritten the value of the previous Foo => 3 pair.

In the example we have assigned to a new hash called `%team_x`, but this would have worked exactly the same way if we assigned it back to either of the `%team_a` or `%team_b`.

Insert a hash reference into another hash

examples/insert_hash_ref_in_hash.pl

```
use strict;
use warnings;
use 5.010;
use Data::Dumper;

my %team_a = (
    Foo => 3,
    Bar => 7,
    Baz => 9,
);

```

```
    my %team_b = (
        Moo => 10,
        Boo => 20,
        Foo => 30,
    );


    $team_b{other} = \%team_a;

print Dumper \%team_b;
```

In this example we start with the same two hashes, but instead of merging them together we insert a reference to first hash into the second hash as the **value** of a new key.

`$team_b{other} = \%team_a;`

The back-slash in-front of the hash provides us with a reference to the hash. We assign this to be the value of a new key in the other hash. I pick the name 'other' which probably required great imagination.

Running the script this is the output we get:

```
$VAR1 = {
          'Boo' => 20,
          'Foo' => 30,
          'Moo' => 10,
          'other' => {
                       'Foo' => 3,
                       'Bar' => 7,
                       'Baz' => 9
                     }
        };
```

The order of the pairs is random, but the important part is that the hash now has a new key called 'other', and the value of the key is basically another hash. In this representation we don't see it, but just as in the case of the arrays, here too the internal hash is exactly the same as the `%team_a`. It is not a copy!

In the next example we change the content of the internal hash of `%team_b`.

examples/insert_hash_ref_in_hash_change.pl

```
use strict;
```

```
    use warnings;
    use 5.010;
    use Data::Dumper;

    my %team_a = (
        Foo => 3,
        Bar => 7,
        Baz => 9,
);

    my %team_b = (
        Moo => 10,
        Boo => 20,
        Foo => 30,
    );


    $team_b{other} = \%team_a;

say Dumper \%team_b;
    say Dumper \%team_a;
    say '-------------';

$team_b{other}{Bar} = 700;

    say Dumper \%team_b;
    say Dumper \%team_a;
```

The output shows the two hashes before and after the assignment separated by a line. You can see that the value of 'Bar' as updated in both hashes.

```
$VAR1 = {
          'Boo' => 20,
          'Moo' => 10,
          'Foo' => 30,
          'other' => {
                       'Baz' => 9,
                       'Foo' => 3,
                       'Bar' => 7
                     }
        };

$VAR1 = {
          'Baz' => 9,
```

```
          'Foo' => 3,
          'Bar' => 7
        };

-------------
$VAR1 = {
          'Boo' => 20,
          'Moo' => 10,
          'Foo' => 30,
          'other' => {
                       'Baz' => 9,
                       'Foo' => 3,
                       'Bar' => 700
                     }
        };

$VAR1 = {
          'Baz' => 9,
          'Foo' => 3,
          'Bar' => 700
        };
```