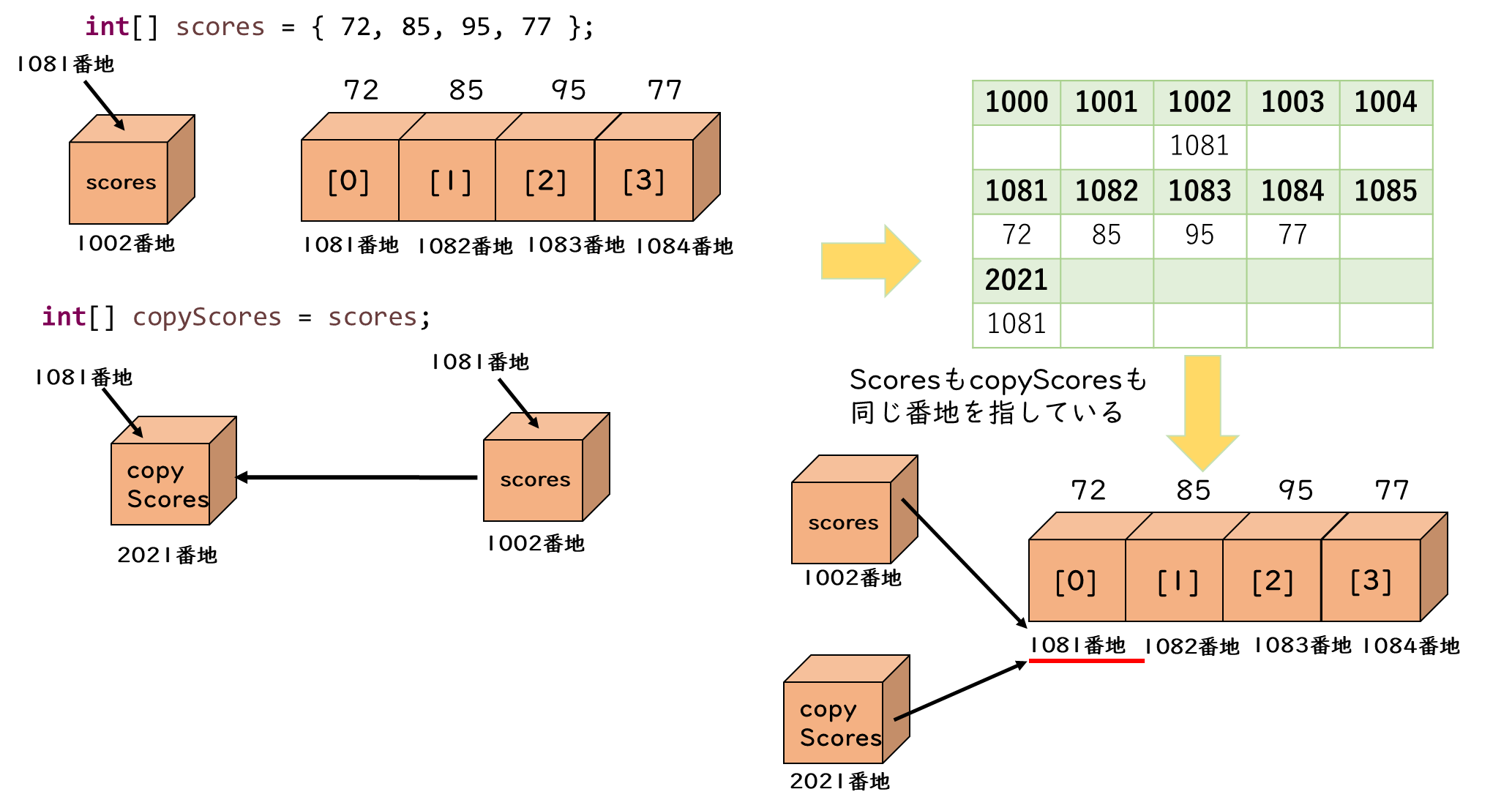
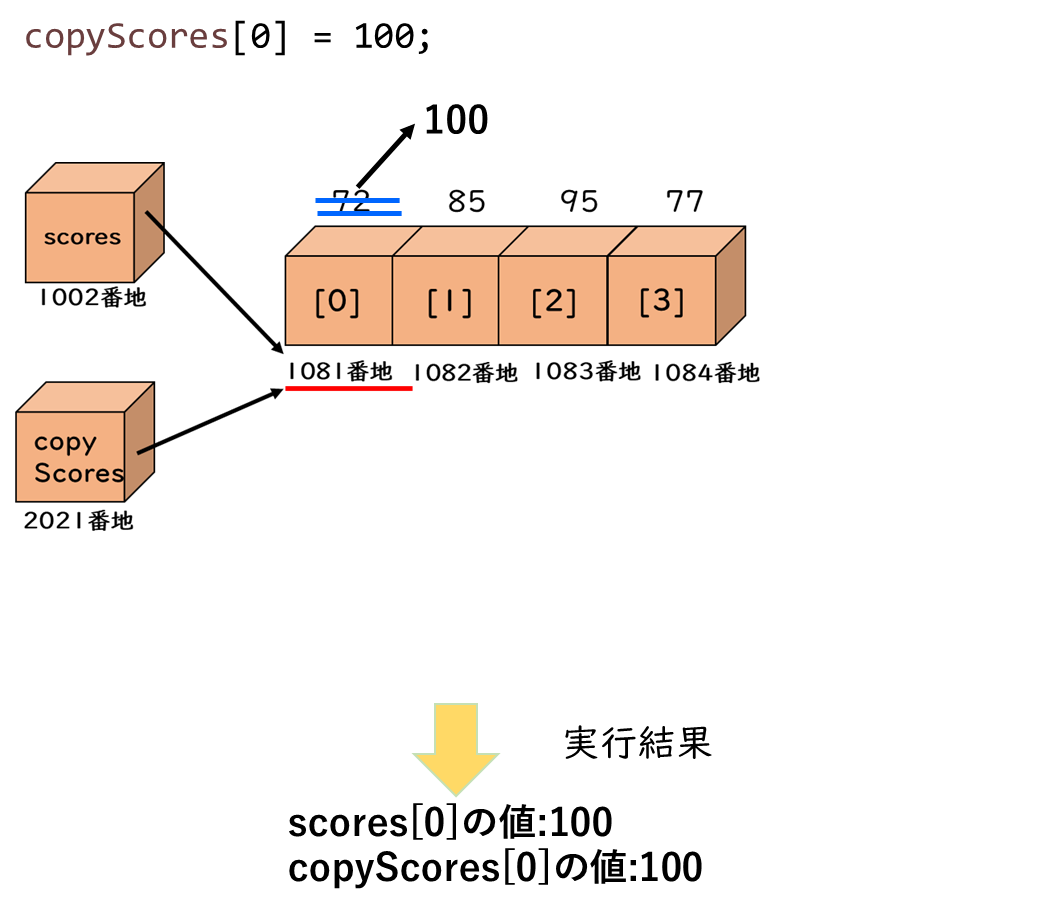
# 配列の仕組み

## 配列のコピーテキスト 自動的に生成された説明

では、本題だ！

ここでは結果がどうしてこうなったのか、先ほどの参照の考え方を踏まえて考えていこう

状況を確認していくと、「scores」も「copyScores」も同じ1081番地を参照している状態つまり2つとも同じところを見ている状態なってしまうんだ。そうするとどうなるかみんなはもう想像がつくよね。



同じところをみてるのだから、値が変更になったらscoresもcopyScoresの値も両方とも変わってします。だから実行結果が両方とも100になるんだ。

なるほど。だから同じ値が出力されたのですね。正直何か壊してしまったのではないかと焦りました。

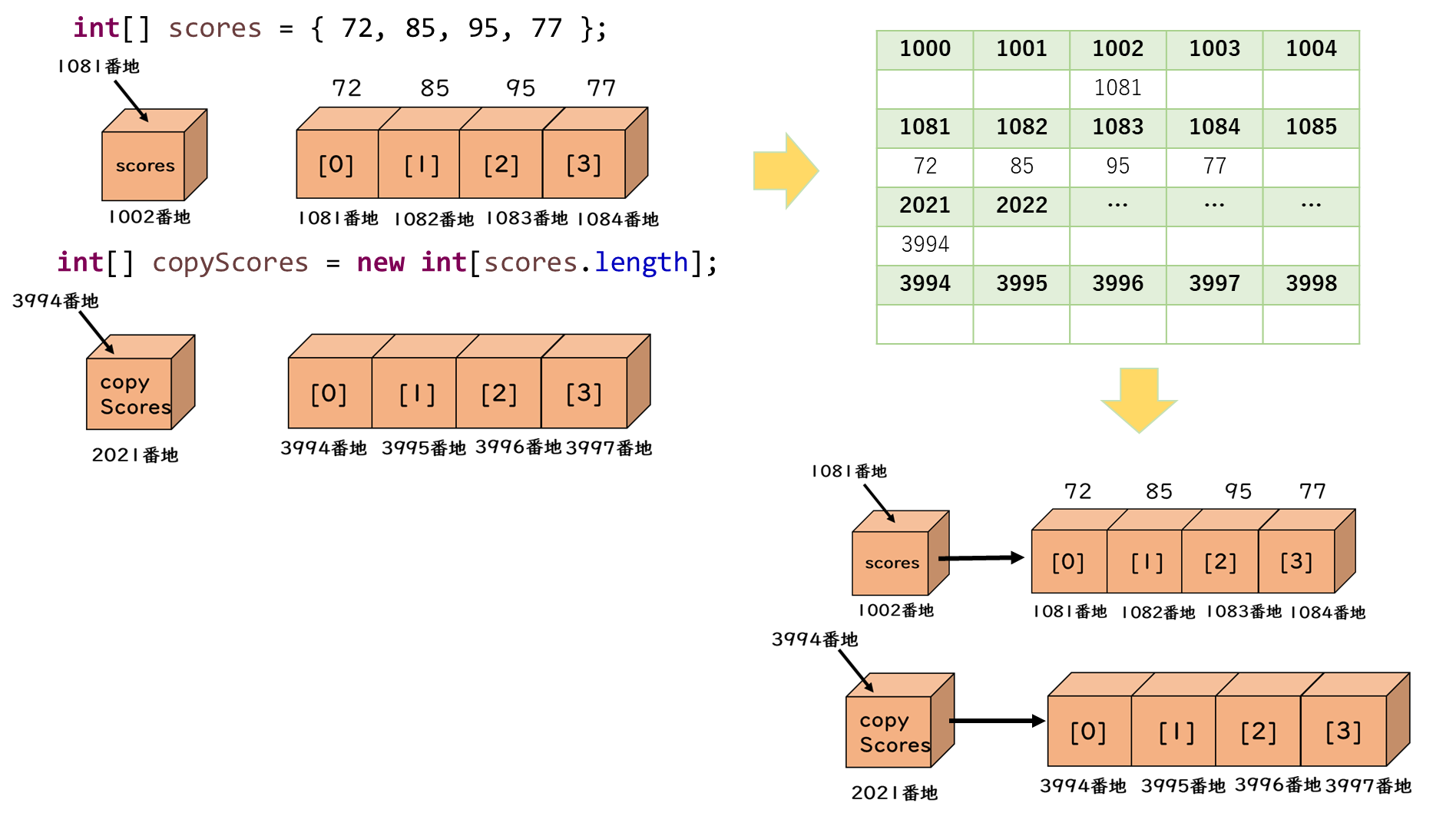
ですが、新たな疑問何ですが、本当の意味でコピーする。つまり、scoresとcopyScoresが同じ番地を見ないでコピーするにはどうすればよいのでしょうか？

確かにその考え方は大事だね！では、ソースコードともに一緒に見ていこう！

先にどういうソースを書くのかを確認していくよ！下のソースと実行結果を見てみよう！

本当だ！正しくコピーできています。でもどうしてかわからないです。

確かにこの部分の考えは結構難しいから一緒に一つずつ見ていこう！



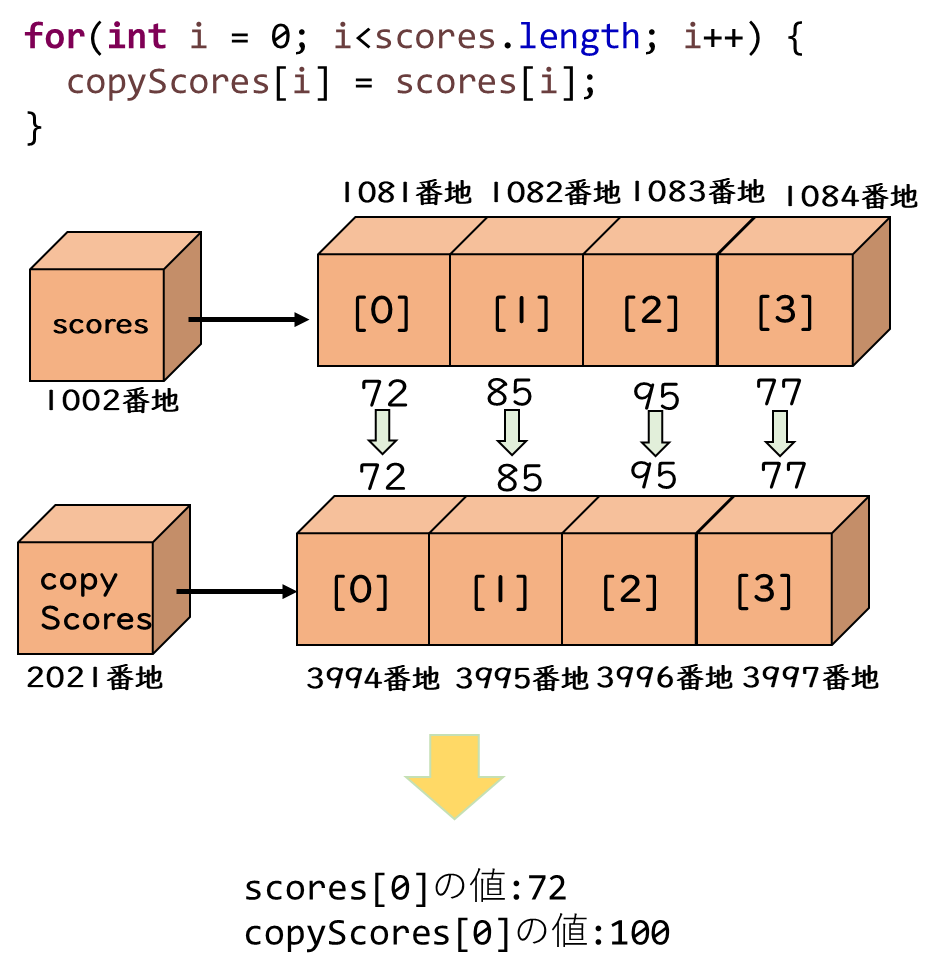
なるほど、この「scores.length」の部分は、コピーする要素の数を指定しているんですね。

今回、scoresの内容をコピーしたいから、scoresの要素の数を取得するために、「scores.length」を書くという書き方を初めて知りました。こういう使い方もあるのですね！

ただ、要素の数をコピーする。つまり入れ物は用意できましたけど、このままだと入れ物を用意しただけで、値がコピーされていないと思うのですが、どうやって値をコピーするのですか？

きちんと、理解が進んでいるね。今回きちんと値をコピーするのにつかわれるのがfor文なんだ。

この部分も一緒に見ていこう！



なるほど、コピー元（scores）のインデックス番号を指定して値を取得した後,コピー先の（copyScpres）のインデックス番号を指定して値を代入しているのでね！

その通り！

でこのコピーには実は名前があるんだ

最初に紹介したコピー方法の場合、

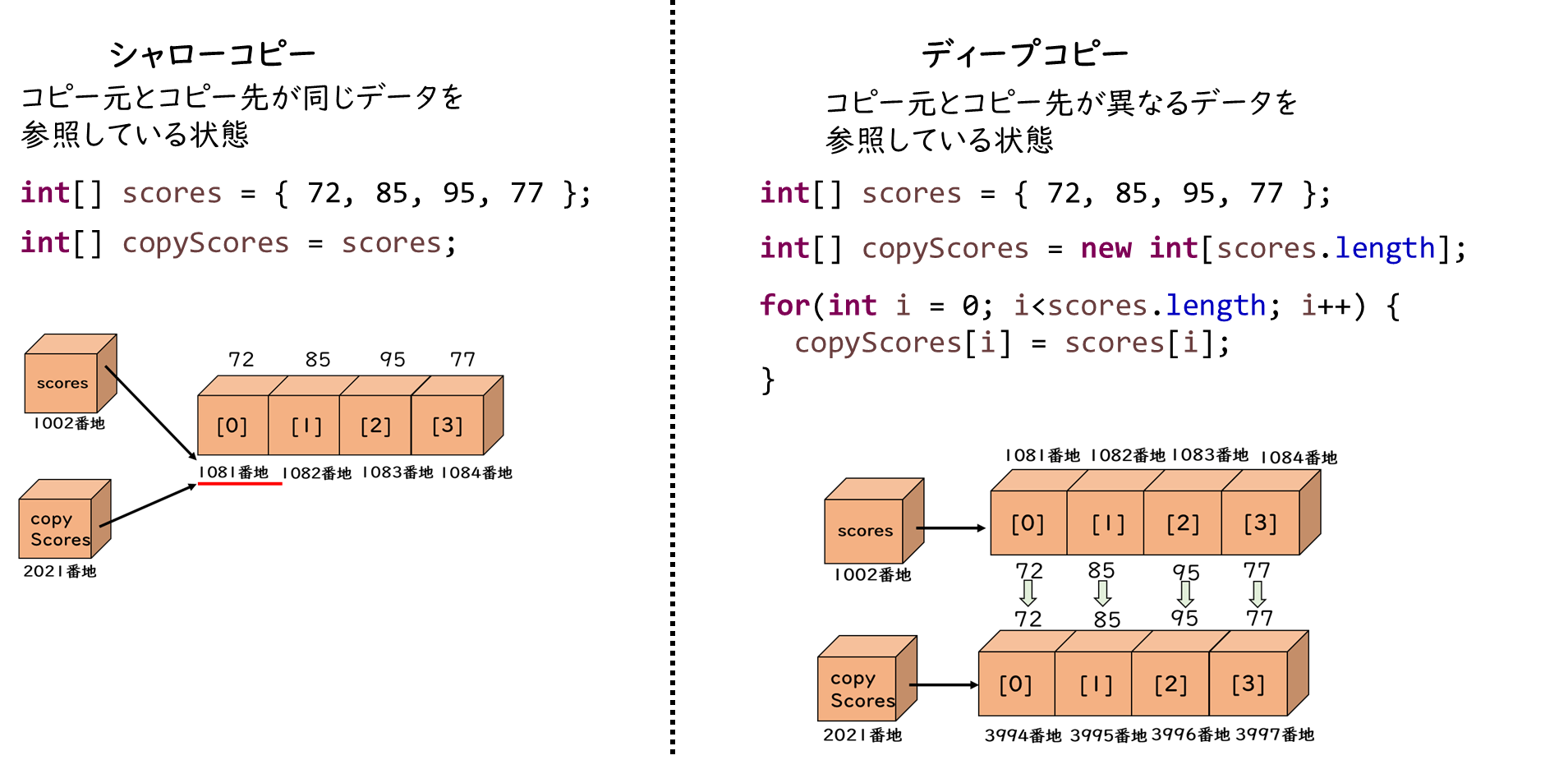
コピー先がコピー元と同じ番地を参照するようにコピーする方法を「シャローコピー」

コピー先とコピー元が違う番地を参照していてなおかつ値もコピーできている状態を「ディープコピー」

という言い方をするからぜひ知っておこう！

ごちゃごちゃしてしまうので申し訳ないのですが、図でまとめてもらってもいいですか？

もちろん！じゃあ図でまとめて頭の中を整理整頓しよう！



他に先ほど習った拡張for文でもディープコピーをすることができるから参考までだけどこんな書き方もあるんだなということを知っておく良いかな！

テキスト

自動的に生成された説明