Advanced C# Collections

ARRAYS, LISTS, AND COLLECTION EQUALITY

Simon Robinson
SOFTWARE DEVELOPER

@techiesimon www.simonrobinson.com

Course Overview

Full range of Microsoft collections

- Lists, dictionaries, etc.

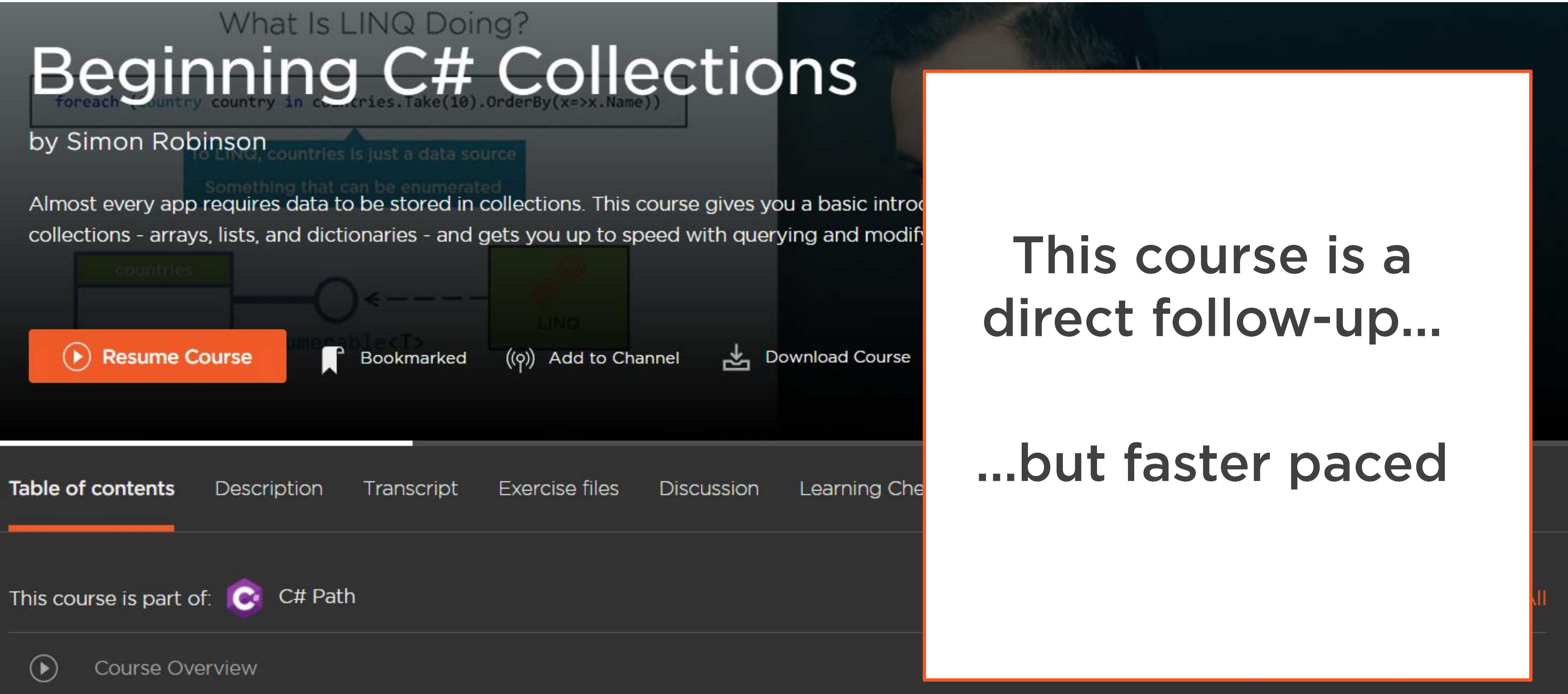
Performance

Read-only and immutable collections

Collection interfaces

Concurrency

Prerequisites


What Is LINQ Doing?

Beginning C# Collections

```
foreach (Country country in countries.Take(10).OrderBy(x=>x.Name))
```

by Simon Robinson

To LINQ, countries is just a data source

Something that can be enumerated

Almost every app requires data to be stored in collections. This course gives you a basic introd
collections - arrays, lists, and dictionaries - and gets you up to speed with querying and modify

This course is part of: C# Path

Course Overview

This course is a direct follow-up…

…but faster paced

Alternative Prerequisites

```
public List<strin
public string[] A
```

Arrays, lists, dictionaries

```
foreach (int x in
{
```

Enumerating

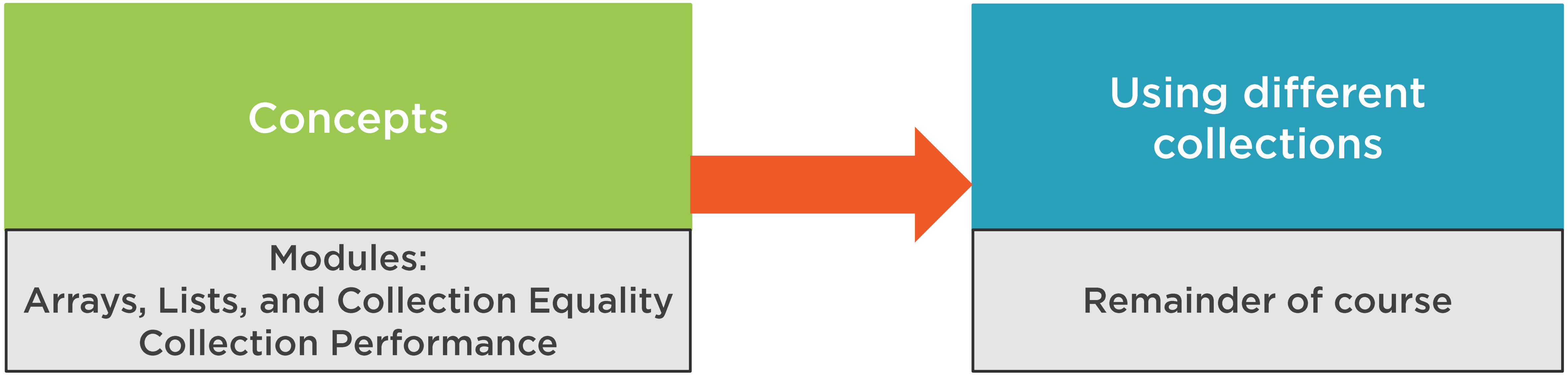
```
var people = from
```

Simple LINQ queries

```
string s = id[i];
```

Element lookup

Course Structure


Concepts
Modules:
Arrays, Lists, and Collection Equality
Collection Performance
Using different collections
Remainder of course

Demo

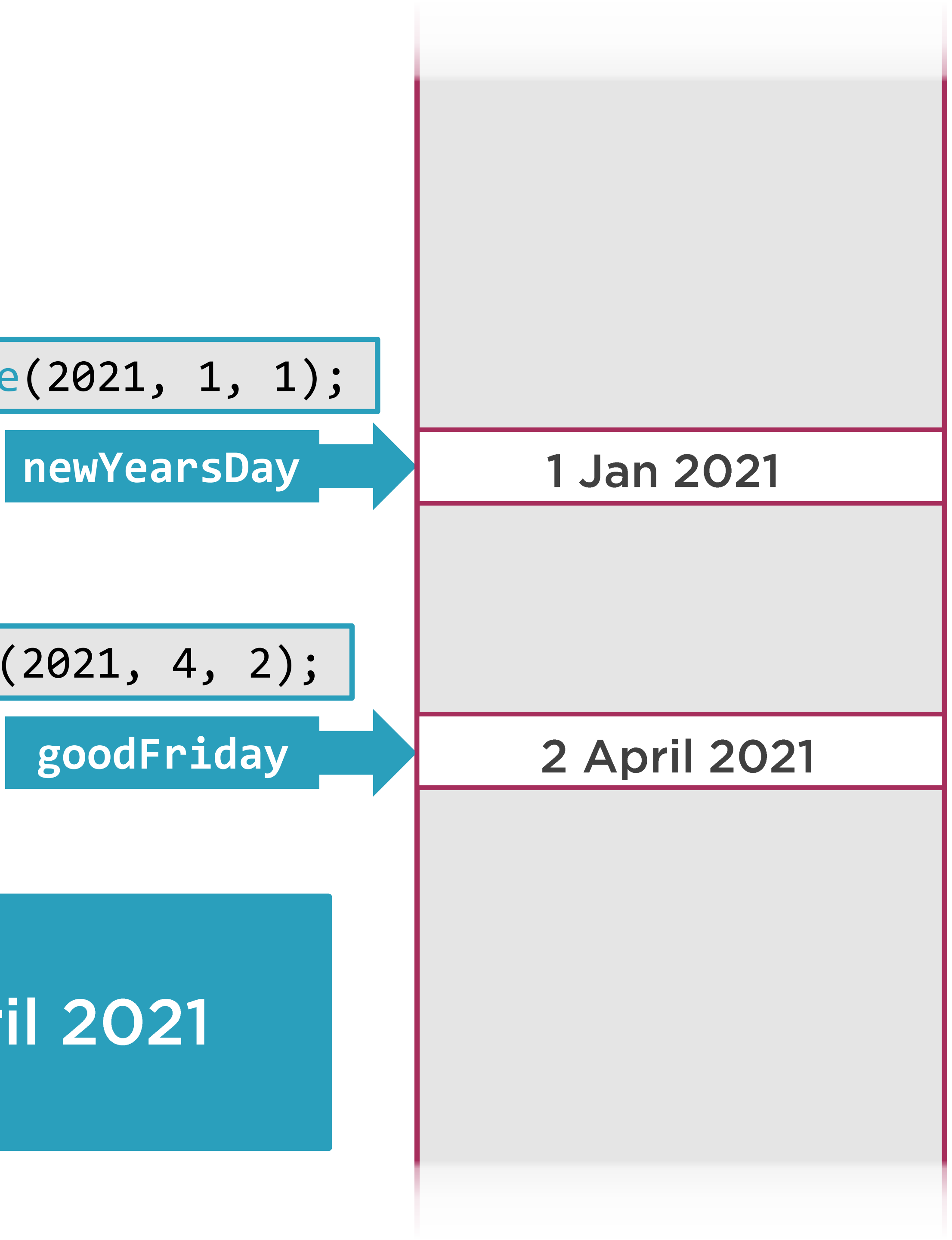
A puzzle to solve!

- Console app
- Array of dates of holidays

Value Types

```
DateTime newYearsDay = new DateTime(2021, 1, 1);
```

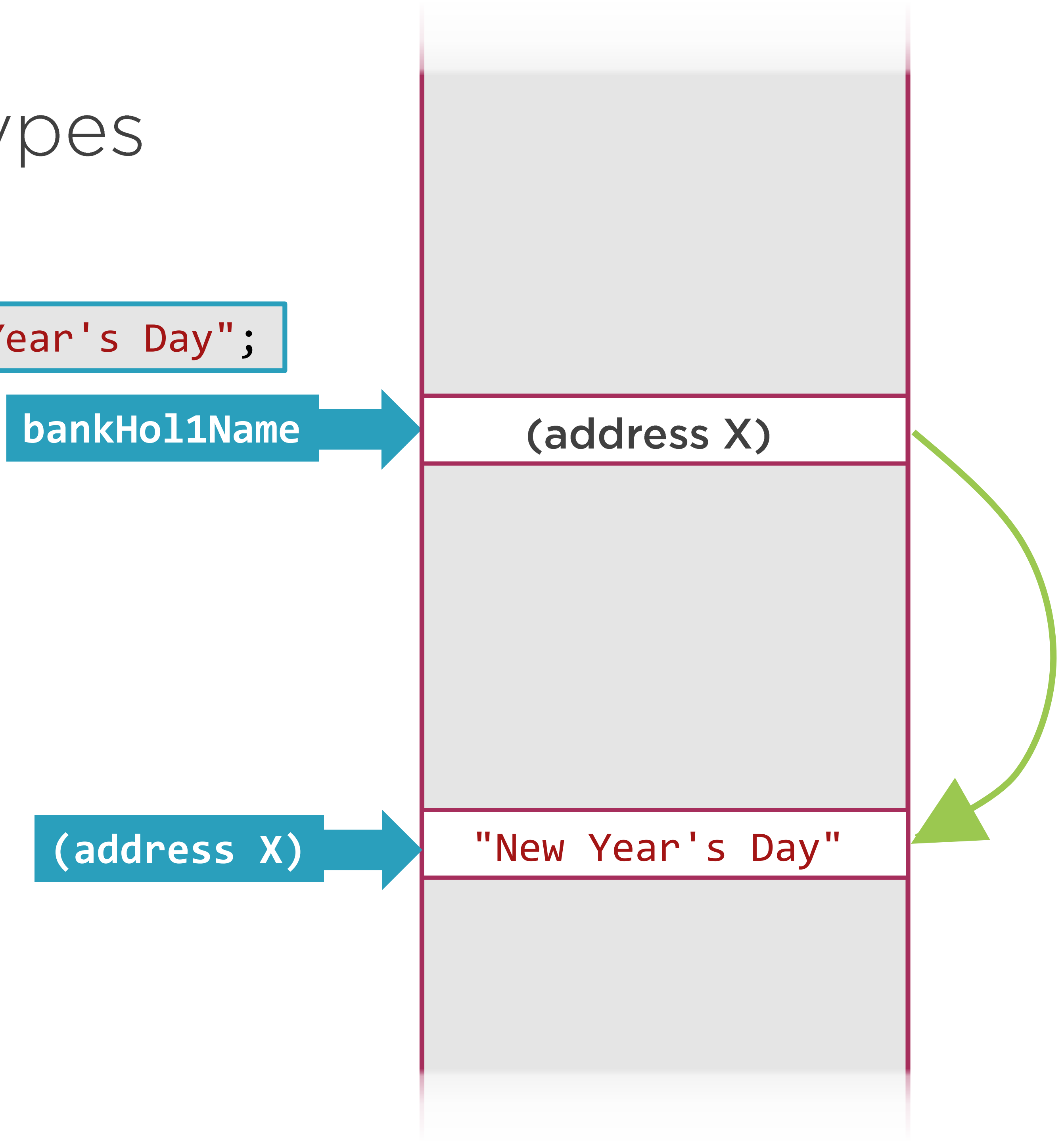
```
DateTime goodFriday = new DateTime(2021, 4, 2);
```



1 Jan 2021 != 2 April 2021

Reference Types

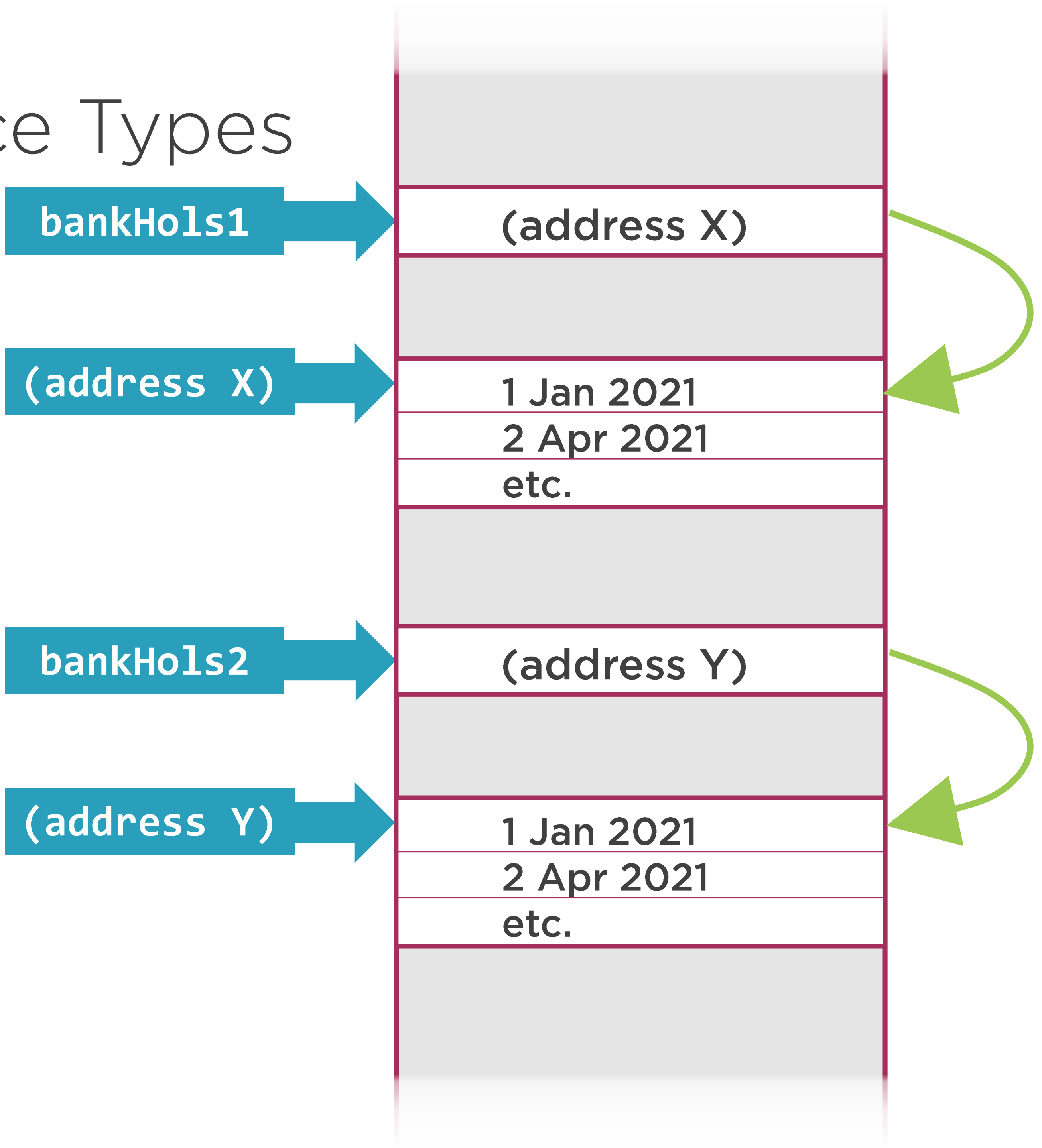
```
string bankHol1Name = "New Year's Day";
```



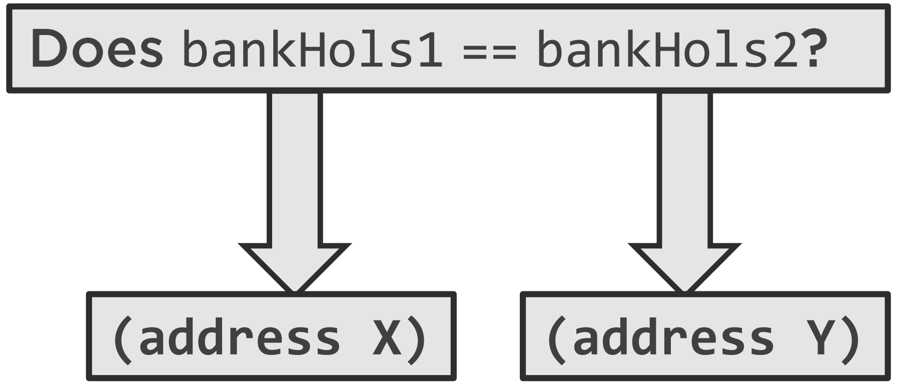
Arrays Are Reference Types

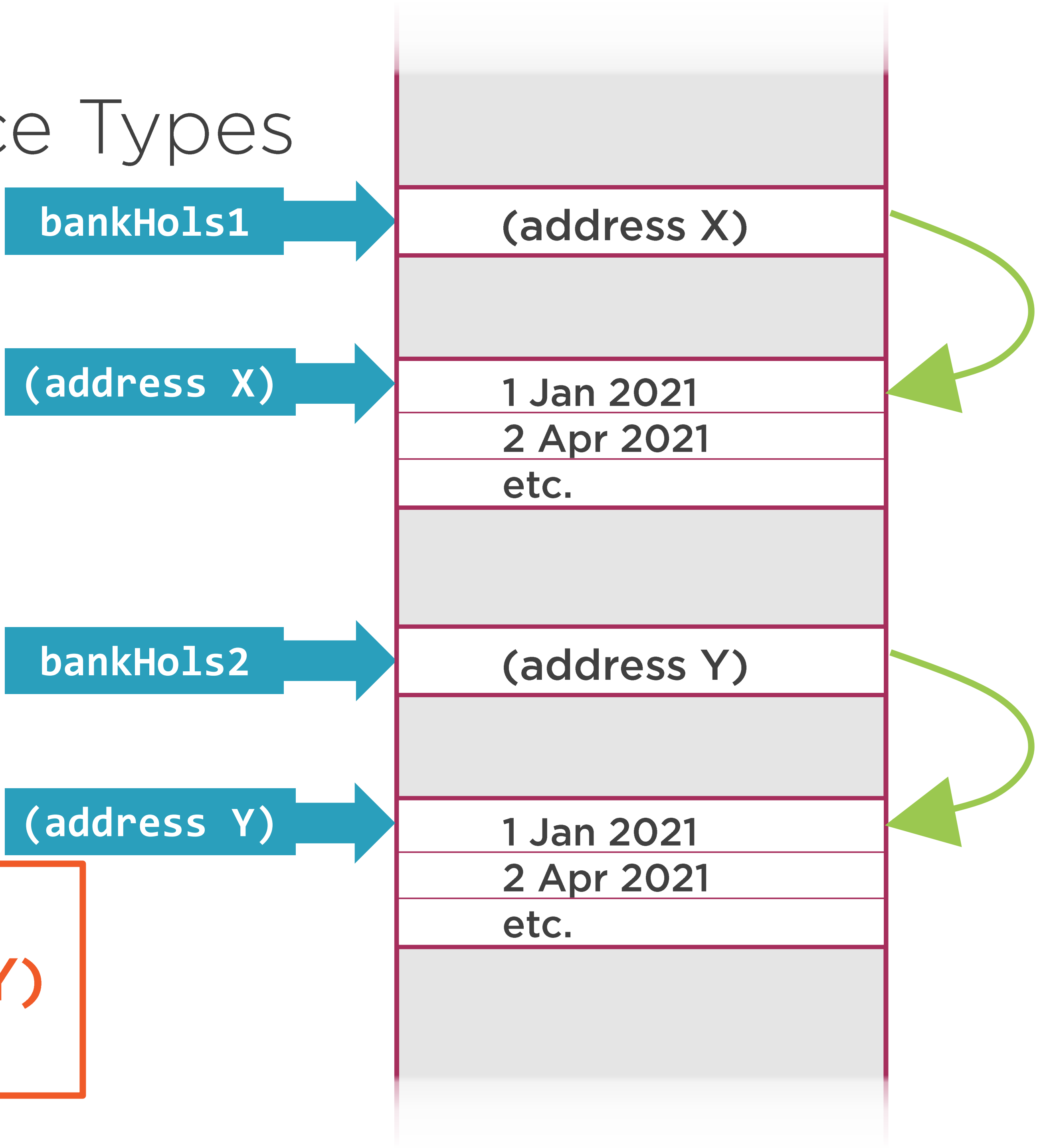
```
DateTime[] bankHols1 =
{
    new DateTime(2021, 1, 1),
    new DateTime(2021, 4, 2),
    new DateTime(2021, 4, 5),
    // etc.
```

```
DateTime[] bankHols2 =
{
    new DateTime(2021, 1, 1),
    new DateTime(2021, 4, 2),
    new DateTime(2021, 4, 5),
    // etc.
```



Arrays Are Reference Types





No!
(address X) and (address Y) are different!

Reference equality:

Same memory location
NOT equal values

Array Equality

```
DateTime[] bankHols1 =
{
    new DateTime(2021, 1, 1),
    new DateTime(2021, 4, 2),
    new DateTime(2021, 4, 5),
    // etc.
```

```
DateTime[] bankHols2 =
{
    new DateTime(2021, 1, 1),
    new DateTime(2021, 4, 2),
    new DateTime(2021, 4, 5),
    // etc.
```

To a human, these are equal (same values)

To C#, they are not equal (different memory locations)

Array equality:
Are they the same array/same instance?

String Equality

```
string bankHol1Name = "New Year's Day";
string bankHol2Name = "New Year's Day";
bool areEqual = (bankHol1Name == bankHol2Name);
```

This does test whether values are equal

But only because Microsoft overrode default reference behaviour for strings

Demo

What if you **want to test collections for value equality?**

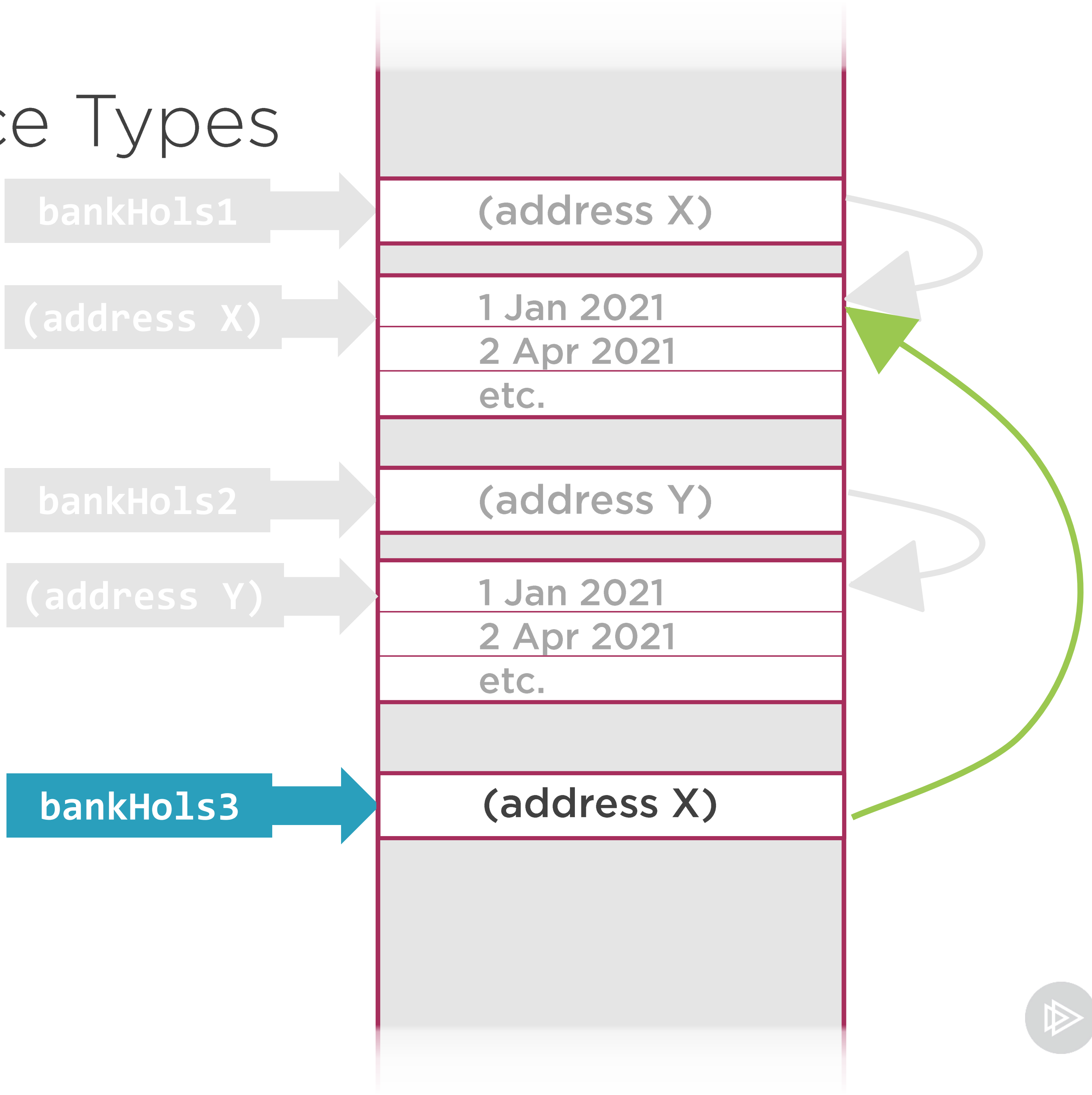
`SequenceEqual()` **extension method**

Arrays Are Reference Types

```
DateTime[] bankHols1 =
{
    new DateTime(2021, 1, 1),
// etc.
```

```
DateTime[] bankHols2 =
{
    new DateTime(2021, 1, 1),
// etc.
```

```
DateTime[] bankHols3
    = bankHols1
```

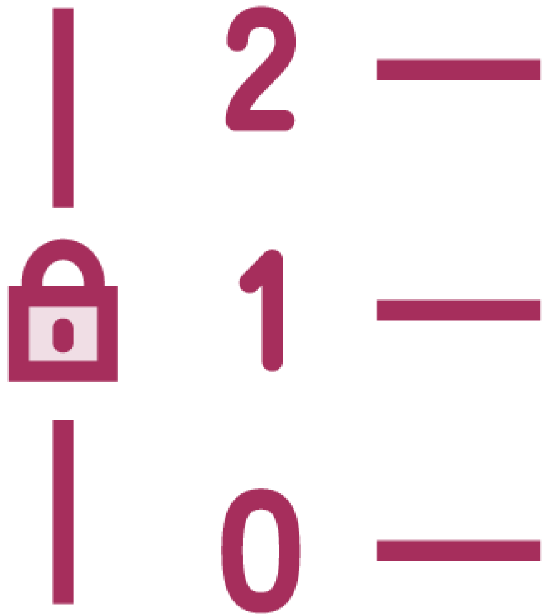


Structure and Purpose of Arrays

Demo

Size of an array is fixed

Arrays



How often do you need this?

Size is fixed

- Can replace elements
- But not add or remove them

Can only look up by index

Collection Lookup Scenarios

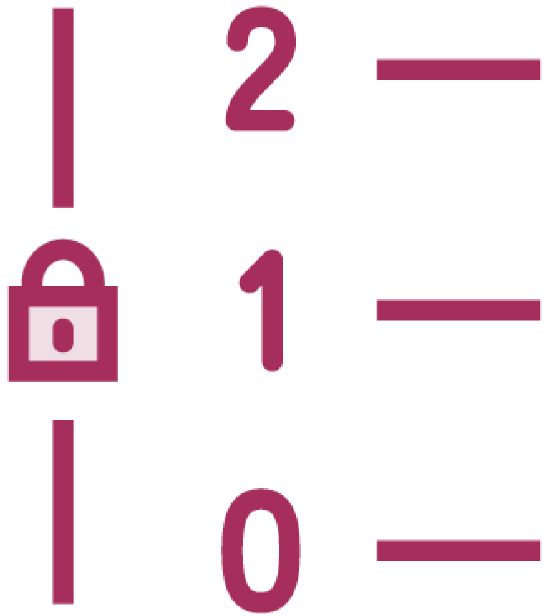
Person using social security number

Employee using name

Product using product ID

These all require keys not indices!

Arrays



Size is fixed

- Can replace elements
- But not add or remove them

Can only look up by index

Why did MS choose this?

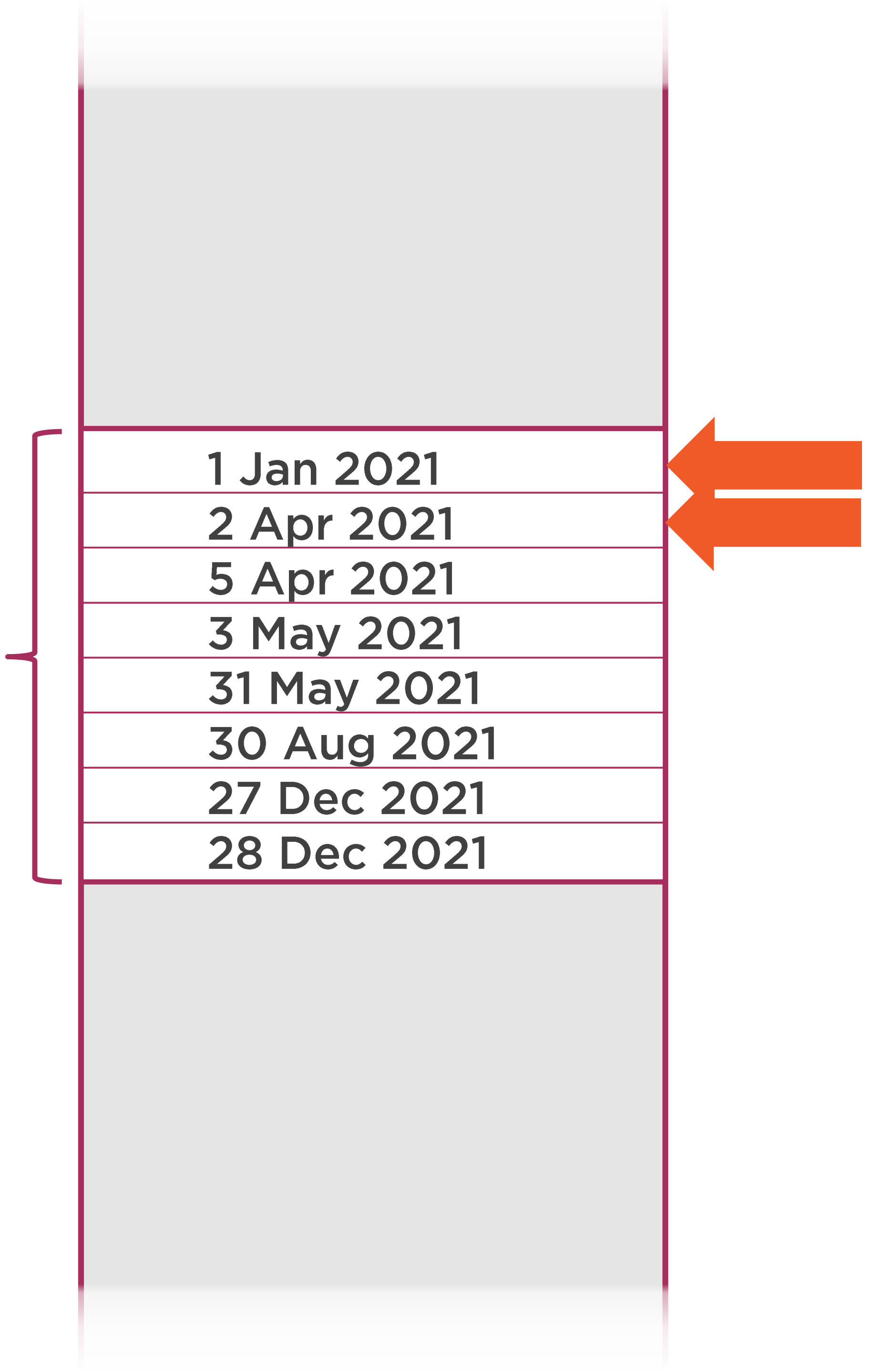
The answer will help us understand performance too

Arrays under the Hood

```
DateTime[] bankHols1 =
{
    new DateTime(2021, 1, 1),
// etc.
```

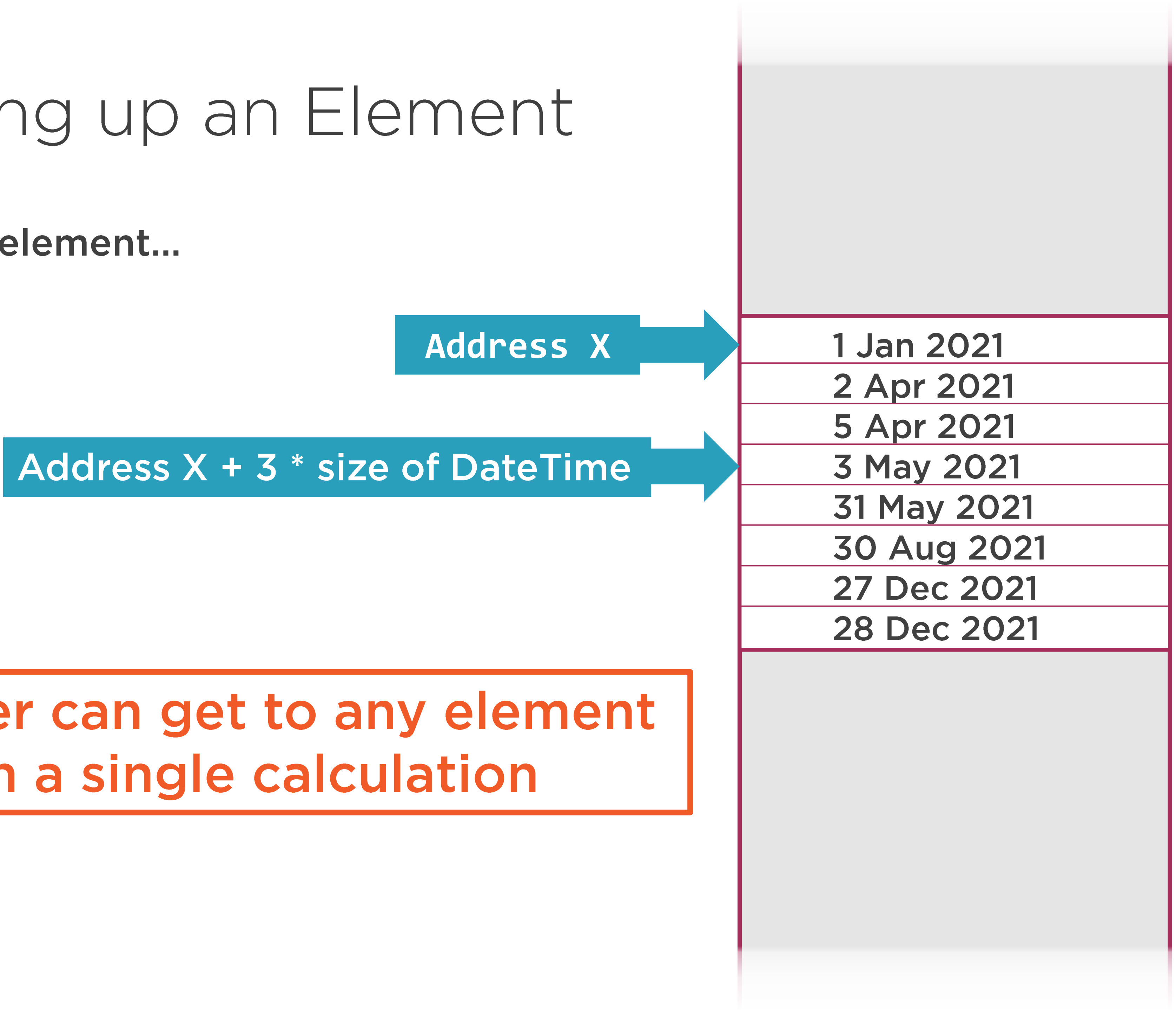
Single block of memory

Items stored sequentially



Looking up an Element

To get 4th element…



Computer can get to any element with a single calculation

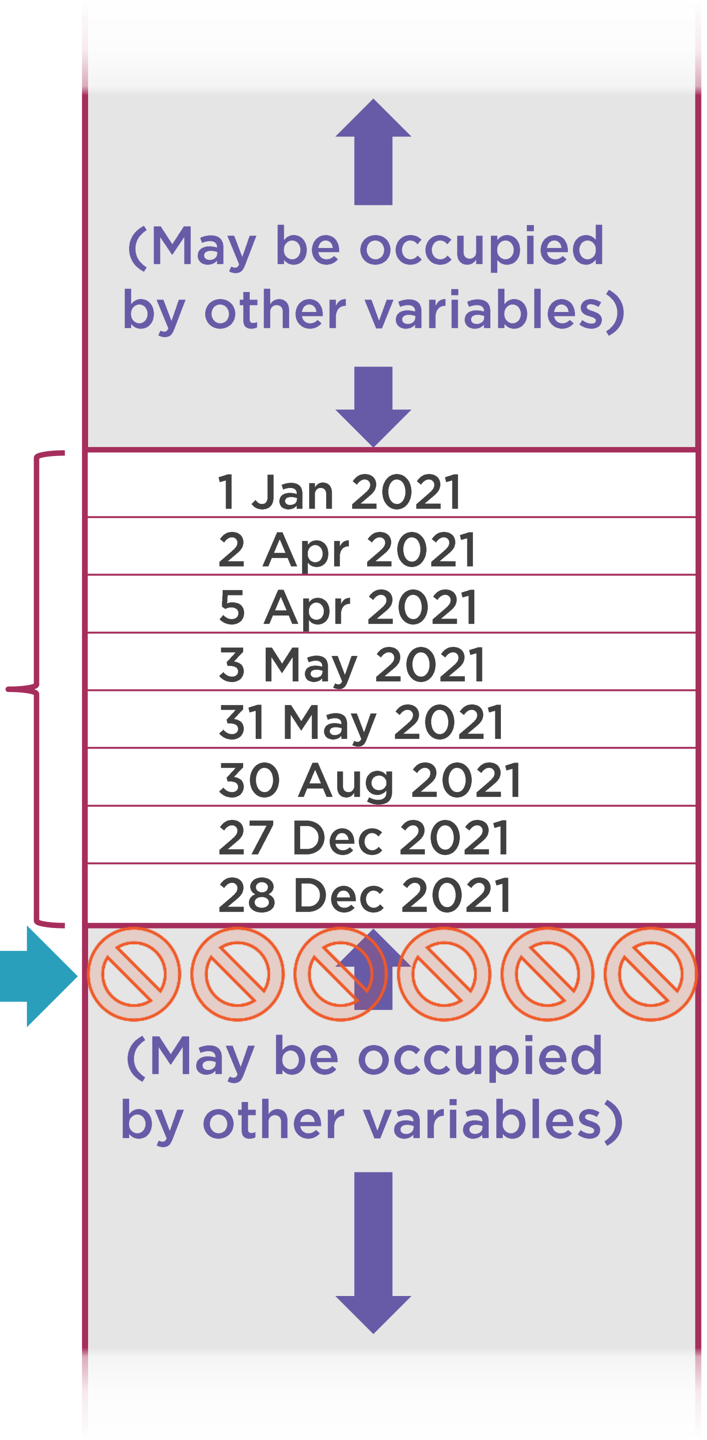
Adding an Element

```
DateTime[] bankHols1 =
{
    new DateTime(2021, 1, 1),
// etc.
```

The array owns this block of memory

New element must go here

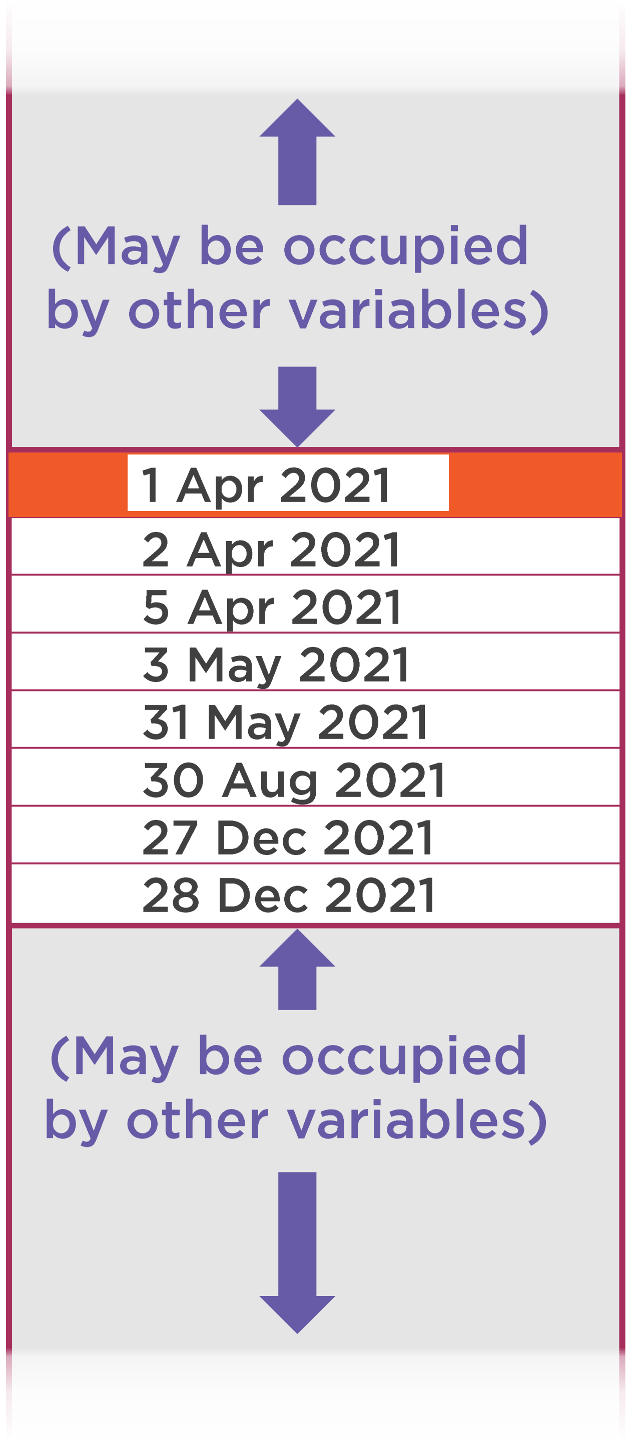
```
// This won't work for an array!
bankHols1.Add(new DateTime(2022, 1, 1));
```



Replacing an Element

This is easy….

```
bankHols1[0] = new DateTime(2021, 4, 1);
```



Arrays vs. Dictionaries

Arrays

Can’t add elements because of how arrays are stored

Simple

Quick to look up/enumerate

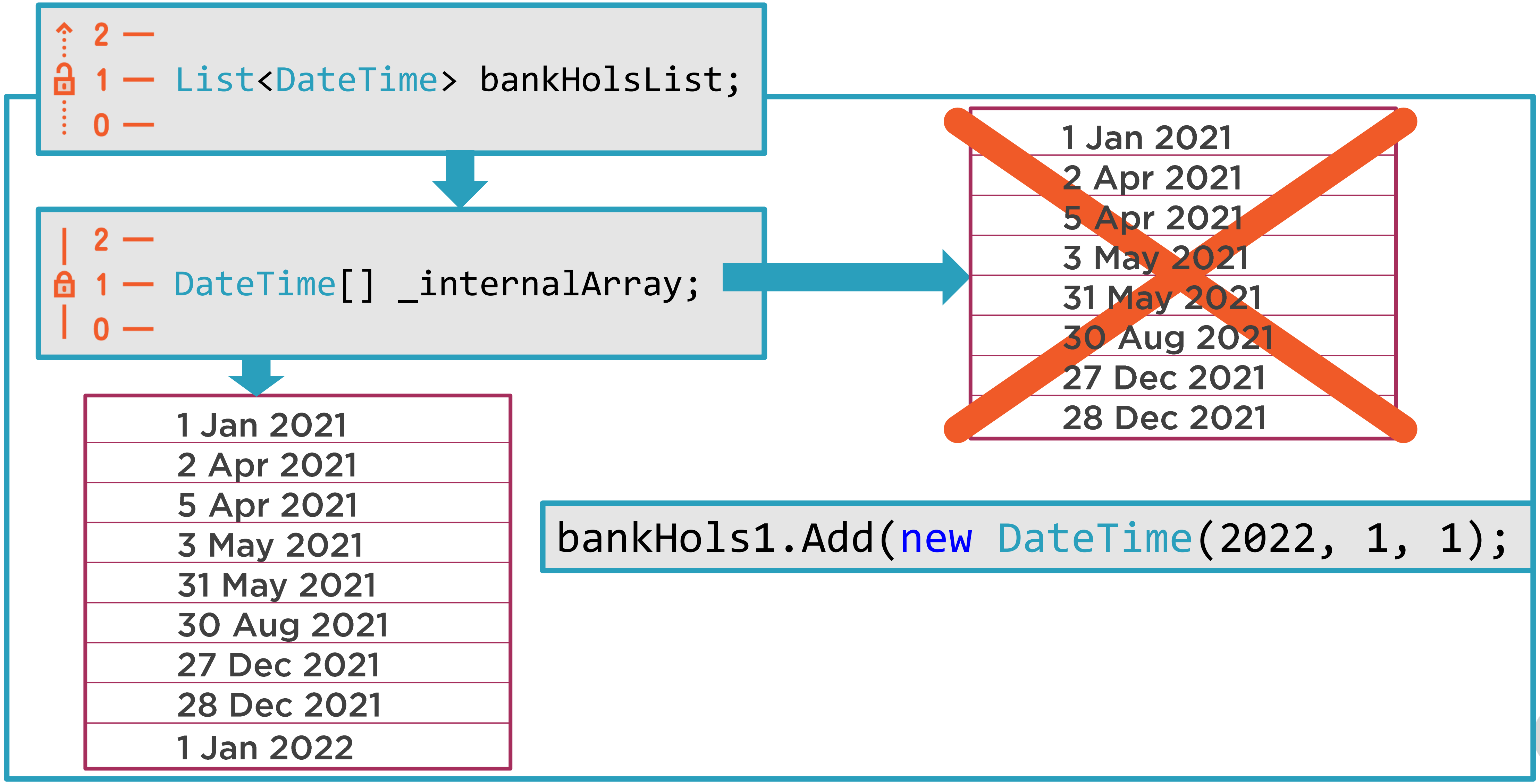
Dictionaries

More useful because of keyed access

Requires complex data structures

(Most collections use arrays under the hood)

List<T> Contains an Array



```
List<DateTime> bankHolsList;
```

```
DateTime[] _internalArray;
```

1 Jan 2021
2 Apr 2021
5 Apr 2021
3 May 2021
31 May 2021
30 Aug 2021
27 Dec 2021
28 Dec 2021
1 Jan 2022

~~1 Jan 2021~~
~~2 Apr 2021~~
~~5 Apr 2021~~
~~3 May 2021~~
~~31 May 2021~~
~~30 Aug 2021~~
~~27 Dec 2021~~
~~28 Dec 2021~~

```
bankHols1.Add(new DateTime(2022, 1, 1);
```

List<T>

Adding items is slow

But it is possible

We need to look at collection performance

Summary

Equality comparisons check for the same collection

Assignment copies references, not entire collections

`SequenceEqual()` checks for same values

Arrays: Single block of memory

Lists: Encapsulate arrays