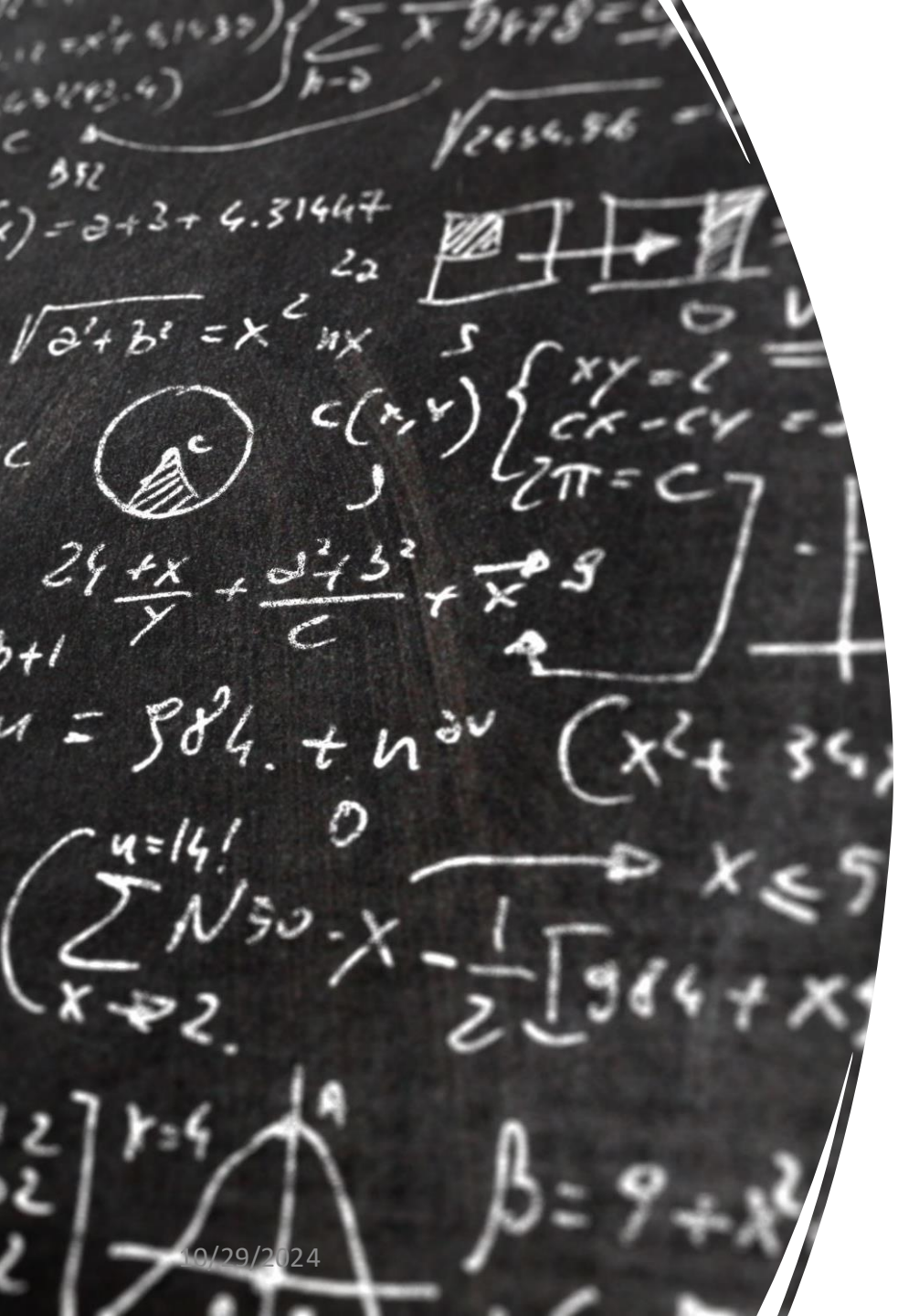


CS 113 – Computer Science I

Lecture 13 – Loops

Tuesday 10/29/2024

Announcements

- HW07
  - Due Monday night 11/04

- Office hours:
  - Adam’s Thursday 2:40-4:00pm

Agenda

- **While Loops**
- For Loops
- Arrays of Arrays

Loops

- Easy way to repeat some computation
- Two kinds of loops:
  - While
  - For
- Loops repeat block of code until the condition becomes false

While loop

While a condition is true, run a block of code

```
while(condition) {
  //run the code in this block
}
```

Example: While Loop

```
int val = 0;
int sum = 0;

int count = 0;
while (count < 6) {
    System.out.print("Enter a number: ");
    val = sc.nextInt();
    sum = sum + val;
    count = count + 1;
}
System.out.println("The sum is "+sum);
```

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Iteration	Count < 6	count	sum

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Iteration	Count < 6	count	sum
0	T	0	1
1	T	1	3
2	T	2	5
3	T	3	7

Exercise: Tracing loops

```
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

Iteration	Count < 6	count	sum

Exercise: Tracing loops

```
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

Iteration	Count < 6	count	sum
0	T	0	10
1	T	2	9
2	T	4	8
3	T	6	7
4	F		

Accumulator pattern

Idea: Repeatedly update a variable (typically in a loop)

Pattern:

1. Initialize accumulator variable
2. Loop until done
    1. Update the accumulator variable

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2

count = count + 1

count = count – 1

product = product * 2

divisor = divisor / 2

message = message + “lol!”

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	count += 1
count = count - 1	count -= 1
product = product * 2	product *= 2
divisor = divisor / 2	divisor /= 2
message = message + “ lol”	message += “ lol”

Exercise: Write a program that computes powers of 2

Write a program, LoopPow2.java, that computes powers of twos. For example,

```
$ java LoopPow2
Enter an exponent: 0
2 to the power of 0 is 1

$ java LoopPow
Enter an exponent: 1
2 to the power of 1 is 2

$ java LoopPow
Enter an exponent: 4
2 to the power of 4 is 16
```

Agenda

- While Loops
- **For Loops**
- Arrays of Arrays

Example: For Loop

```
int val = 0;
String valStr = "";
int sum = 0;

for (int count = 0; count < 6; count = count +1) {
    System.out.print("Enter a number: ");
    valStr = System.console().readLine();
    val = Integer.parseInt(valStr);
    sum = sum + val;
}
System.out.println("The sum is "+sum);
```

Example: For Loop

initialize

condition

update

```
for (int count = 0; count < 6; count = count +1) {

}
```

Exercise: Tracing loops

```
String pattern = "";
for (int i = 0; i < 3; i++) {
    pattern = pattern + "*";
}
System.out.println(pattern);
```

Iteration	i < 3	i	pattern

Exercise: Tracing loops

```
String pattern = "";
for (int i = 0; i < 3; i++) {
    pattern = pattern + "*";
}
System.out.println(pattern);
```

Iteration	i < 3	i	pattern
0	T	0	""
1	T	1	"*"
2	T	2	"**"
3	F	3	"***"

Exercise: LoopPattern.java

```
$ java LoopPattern
Enter a length: 5
*_*_*

$ java LoopPattern
Enter a length: 10
*_*_*_*_*_

$ java LoopPattern
Enter a length: 0


$ java LoopPattern
Enter a length: 1
*
```

Exercise: Nested loops

```
$ java Square
Enter a size: 5
*****
*****
*****
*****
*****

$ java Square
Enter a size: 1
*

$ java Square
Enter a size: 0
```

Iterating through an array

Write a method called printArray that takes in an array of integers and prints out the values in each array:

printArray({1,2,3,4}) -> “1 2 3 4”

While vs For loop

Use a for loop when we know the number of iterations we want

Use a while loop when we don’t know the number of iterations before hand

Agenda

- While Loops
- For Loops
- **Arrays of Arrays**

Arrays of Arrays

int[] array1 is an array of ints

String[] array2 is an array of Strings

What is int[][] array3?

An array of integer arrays

What is String[][] array4?

An array of String arrays

2D array example

What does int[][] array = new int[4][3] look like?

2D array example

What does int[][] array = new int[4][3] look like?

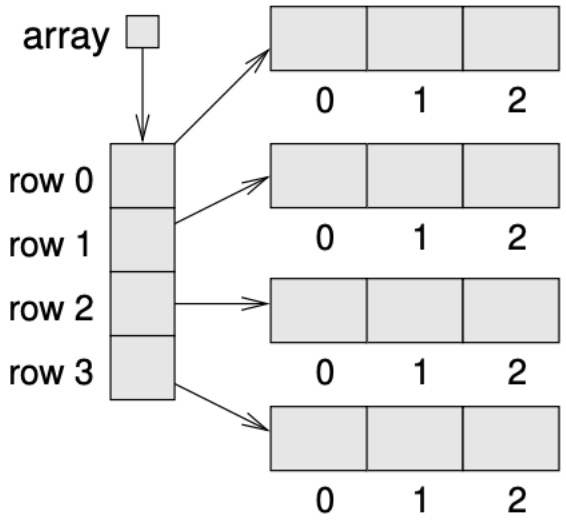


Figure 15.3: Storing rows and columns with a 2D array.

2D Array

Useful for representing a:

- Grid
- Boardgame
- Matrix
- Table
- …

Traversing through a 2D array

What type of loop should we use?

if we know the length, then a for loop

Pseudocode/algorithm:

```
for array in 2D array:
    for item in array:
```

Array Example

Given a square array, compute the sum of the diagonal

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	23	25

Array Example

Given a square 2-D array, compute the sum of the diagonal

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	23	25

Array Example

Given a 2-D array, compute the sum of the perimeter

1	2	3	4	5	2	2
6	7	8	9	10	3	6
11	12	13	14	15	1	6
16	17	18	19	20	9	8

Array Example

Given a 2-D array, compute the sum of the perimeter

1	2	3	4	5	2	2
6	7	8	9	10	3	6
11	12	13	14	15	1	6
16	17	18	19	20	9	8

Agenda

- While Loops
- For Loops
- Arrays of Arrays
- **Bank Example**

Bank example

Keep track of account balances

Use an array:

- Each index represents another account
- The value represents the account's balance

Determine how many accounts we can hold:

- Create a new array of fixed size

Bank example

Over time our bank becomes successful, lots of new clients

No more space for new customers

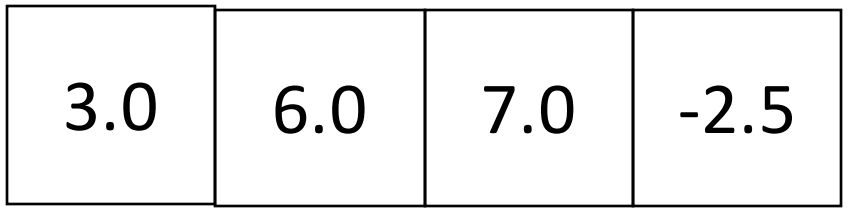
Implementation issue: running out of space in our array

Solution: build a bigger bank!

Building a bigger bank

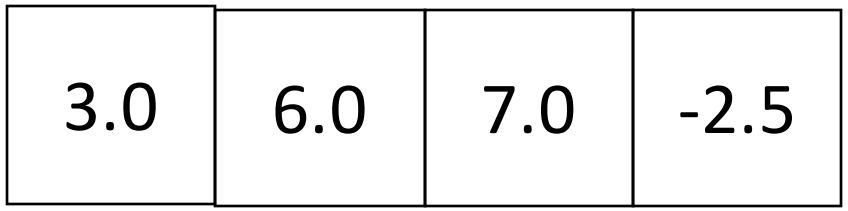
Copying arrays

Old bank

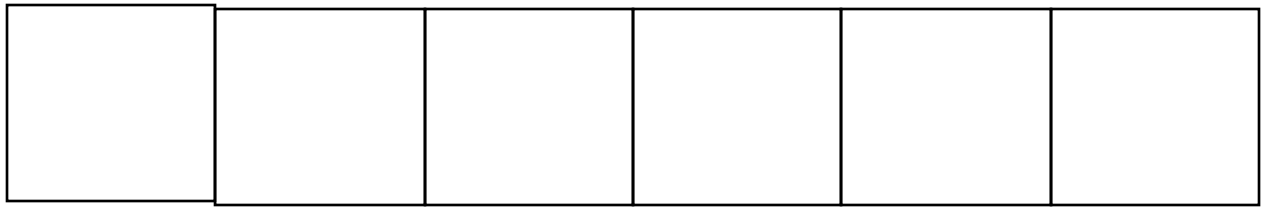


Copying arrays – build the new bank/array

Old bank

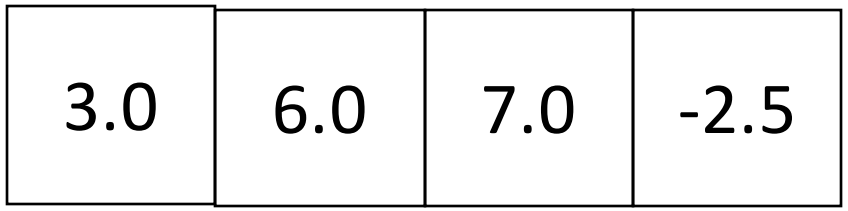


new bank

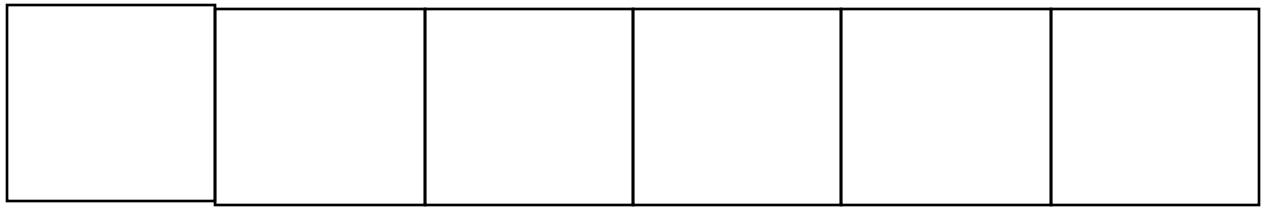


Copying arrays – copy over values/customers

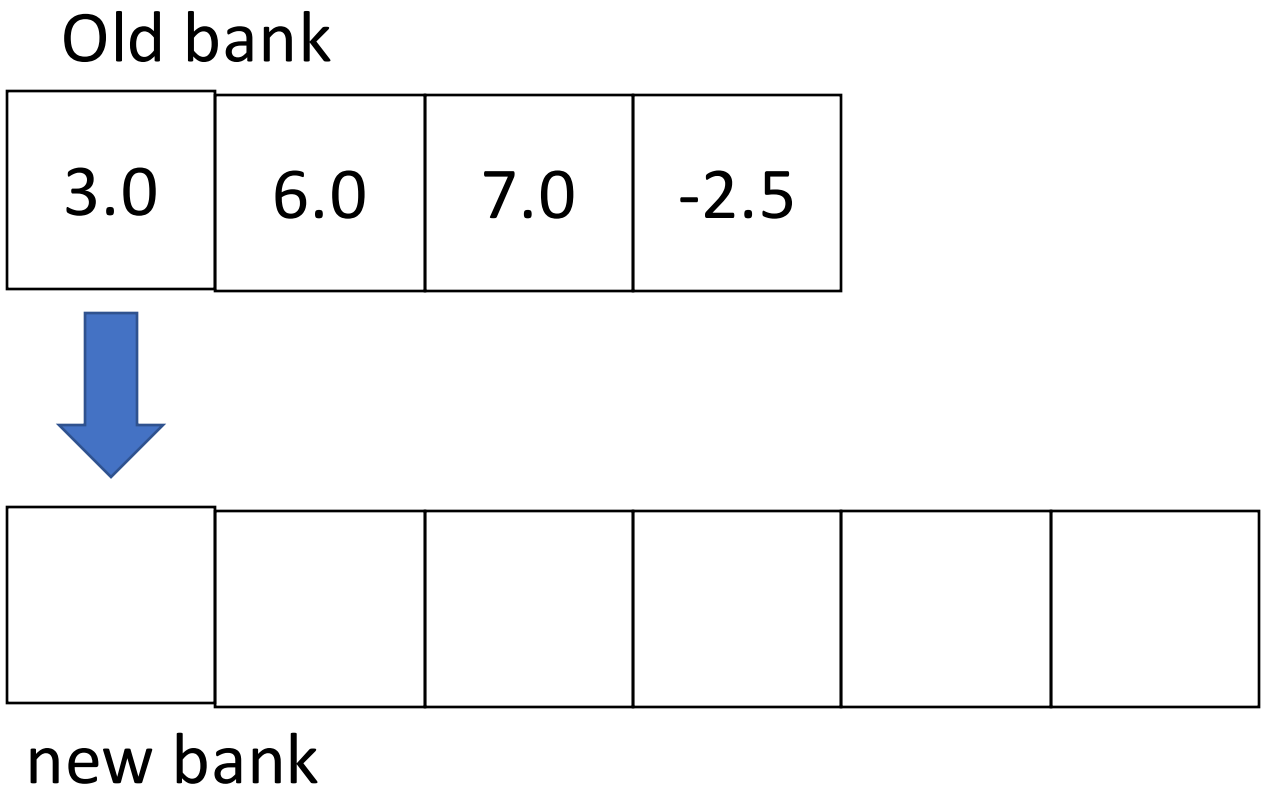
Old bank



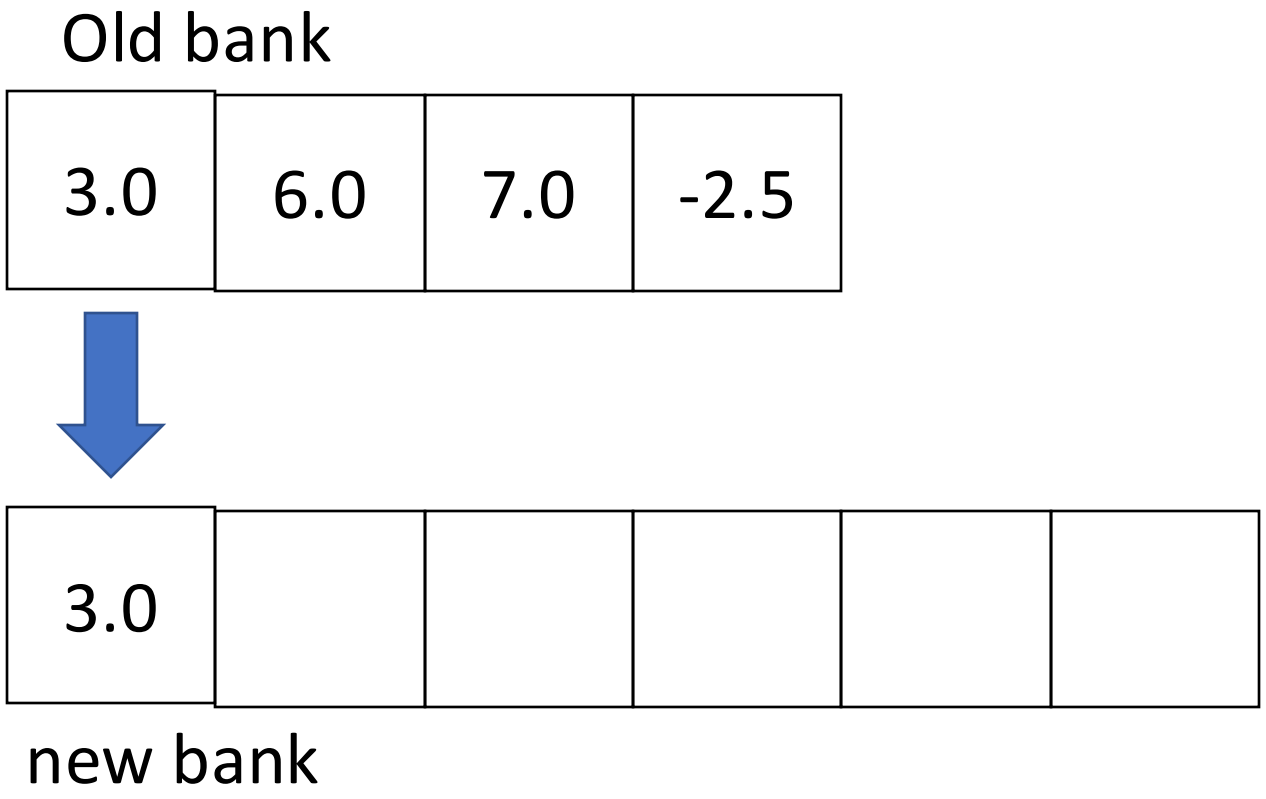
new bank



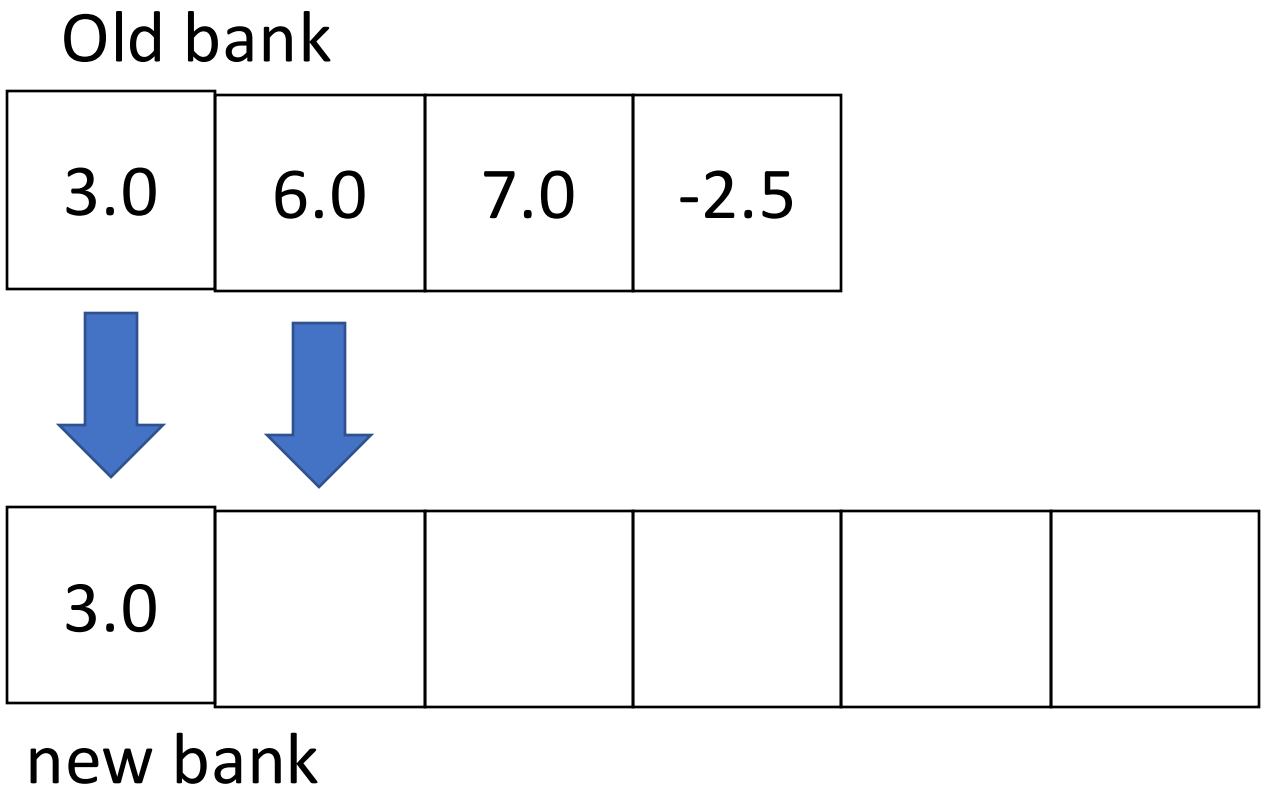
Copying arrays – copy over values/customers



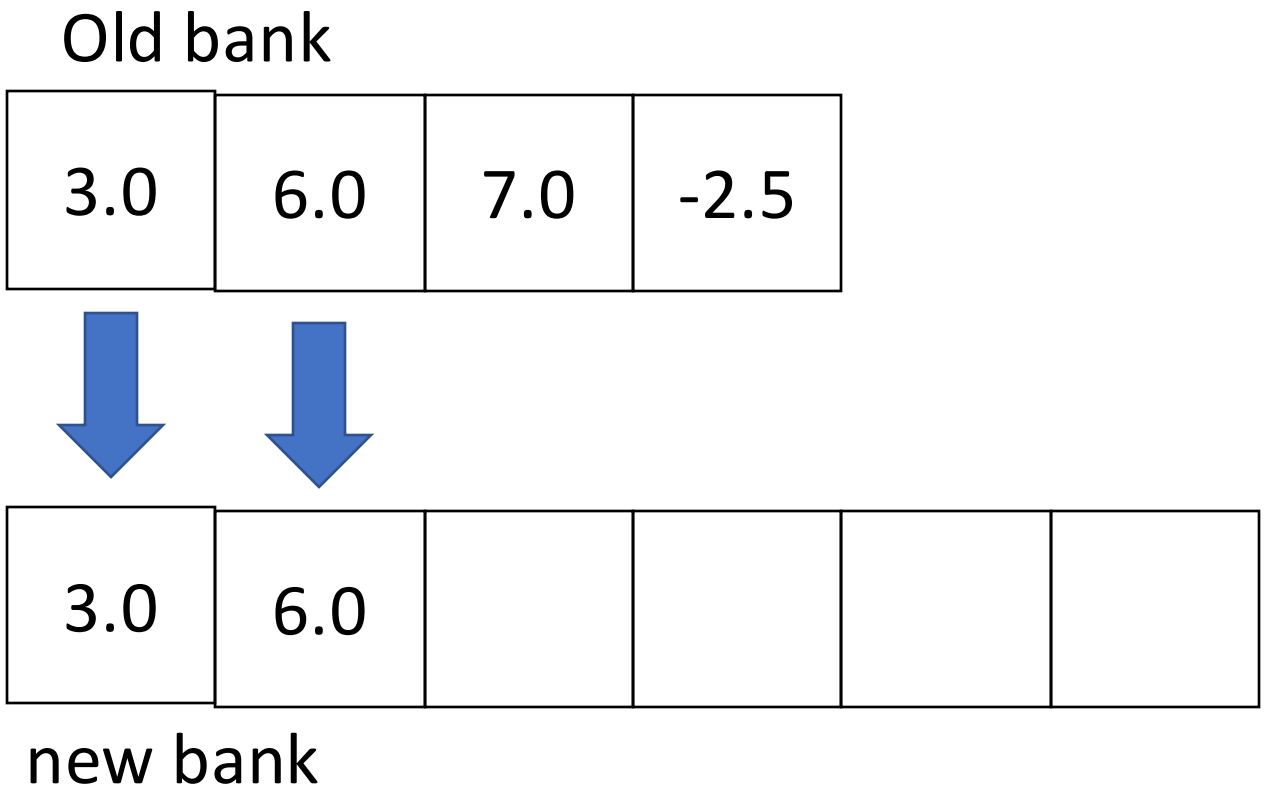
Copying arrays – copy over values/customers



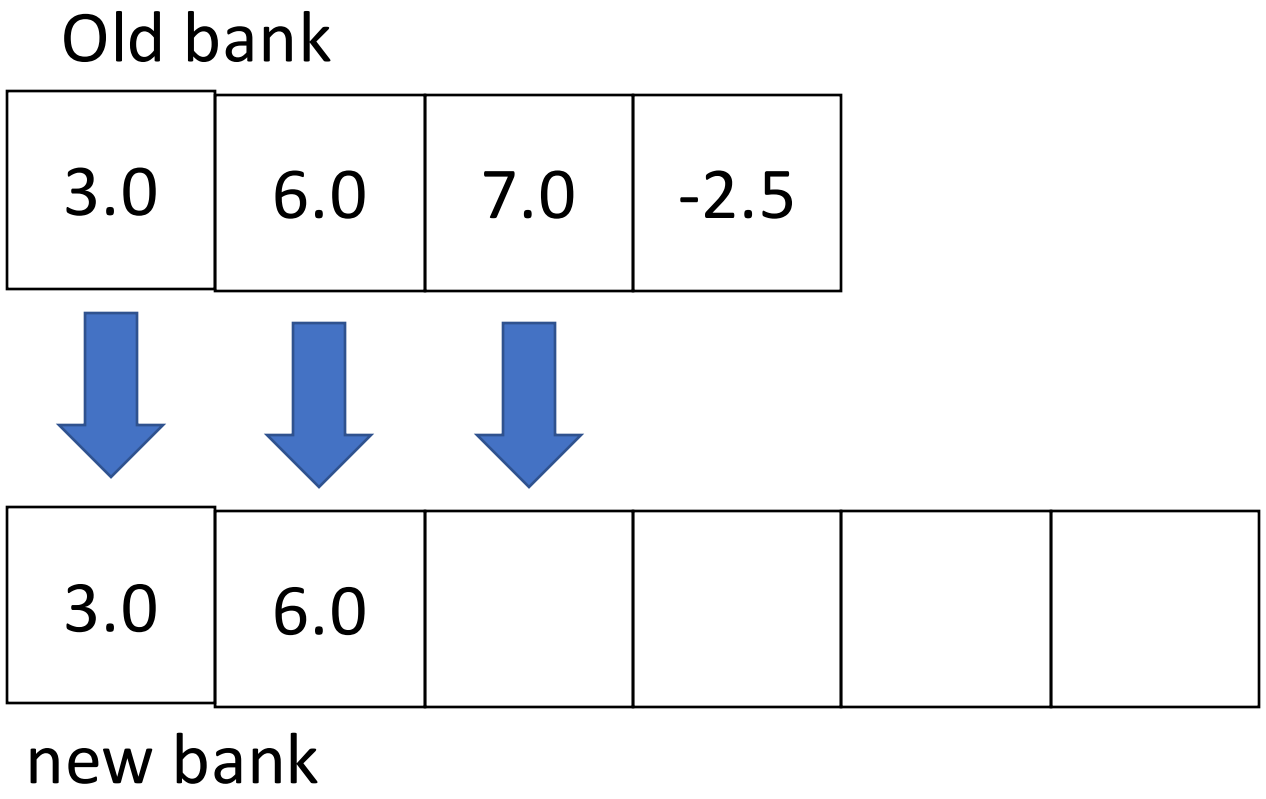
Copying arrays – copy over values/customers



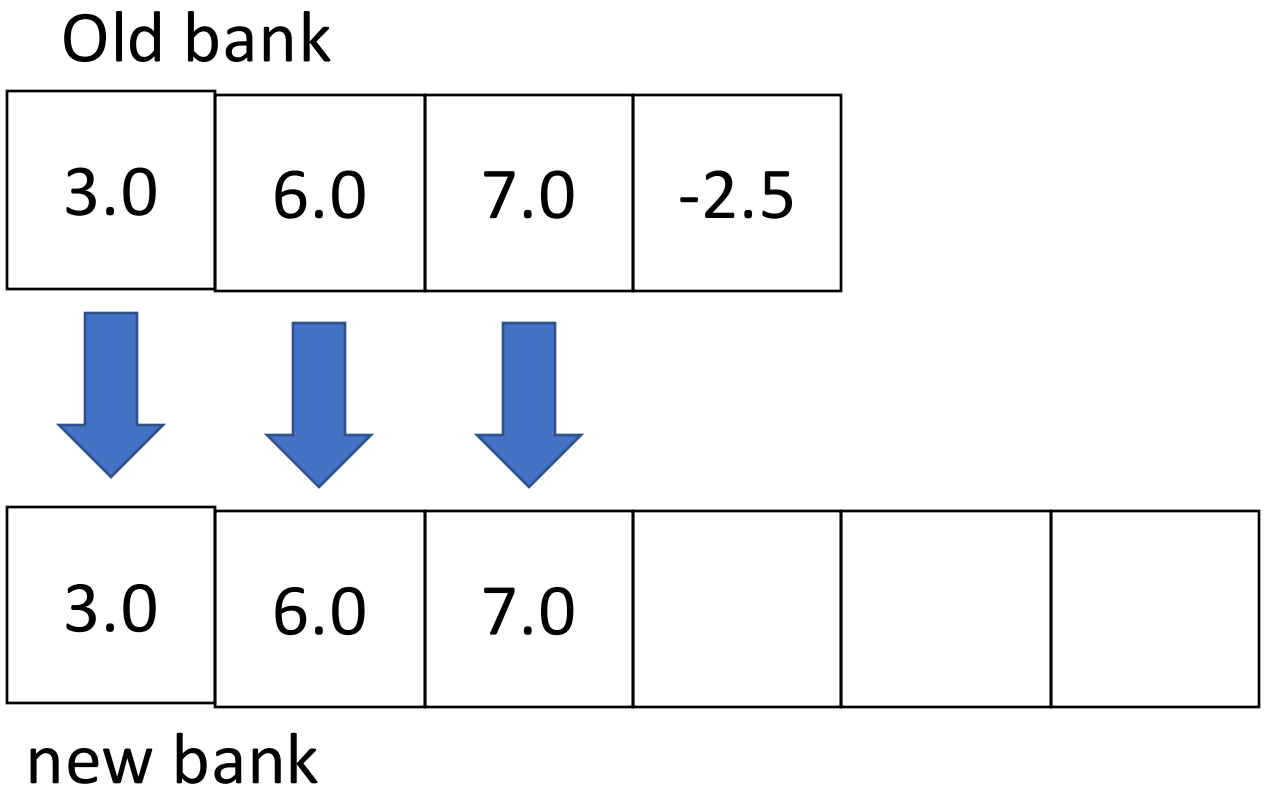
Copying arrays – copy over values/customers



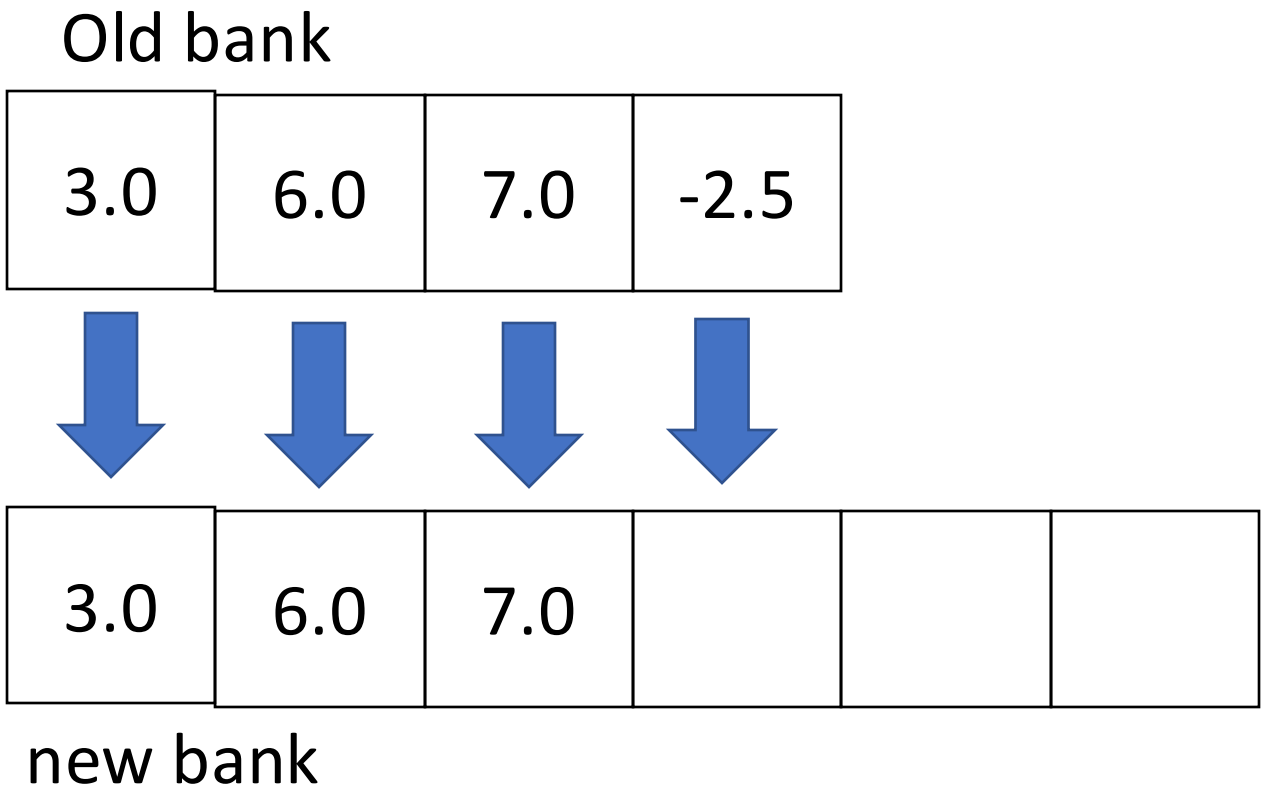
Copying arrays – copy over values/customers



Copying arrays – copy over values/customers



Copying arrays – copy over values/customers



Copying arrays – copy over values/customers

Algorithm

When we run out of space in an array

- Create a new array (that's a bit bigger)
- Copy over all elements from the older array to the new array

How many steps do we take in this algorithm?

- Creating a new array – 1 step
- Copying *n* elements from the old array to the new array – *n* steps

How big should the new array be?

Previous size plus 1

- Pro: not making too much space
- Con: might have to create new arrays a lot of times

As big as possible

- Pro: rarely have to create a new array
- Con: wasted space

Typical solution – previous size x 2