Lesson 1: One-Dimensional (1D) Arrays

45 minutes

Overview

How can I store multiple related values without creating multiple variables?

Students have used variables to store literal values and references to objects. Software engineers often need to store multiple values of the same type in a program. In this lesson, students learn how to use one-dimensional (1D) arrays to store multiple values that can be referenced with one variable. Students practice creating arrays using the `new` keyword and with initializer lists and obtaining the size of the array.

Standards

Full Course Alignment

CSA Conceptual Framework

▶ **VAR-2** - To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure wihtout creating individual variables for each value.

Agenda

Warm Up (5 minutes)
- **Software Engineering Goals**

Activity (35 minutes)
- **1D Arrays**
- **Exploring 1D Arrays**
- **Using 1D Arrays**

Wrap Up (5 minutes)
- **Solve the Problem**
- **Assessment: Check for Understanding**
- **AP Classroom Topic Questions**

Objectives

Students will be able to:

- Declare and initialize a 1D array using the `new` keyword and an initializer list
- Determine the length of a one-dimensional (1D) array

Preparation

- Print copies of the Unit 3 Guide (one for each student)
- Print copies of the Exploring 1D Arrays handout (one for each pair of students)
- Check the **Teacher's Lounge** for verified teachers on the CSA Forum to find additional strategies or resources shared by fellow teachers

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the students

- **Exploring 1D Arrays** - Handout
- **One-Dimensional (1D) Arrays** - Video
- **U3L1 Extra Practice** - Handout
- **Unit 3 Guide** - Handout

Vocabulary

- **data structure** - a structure for organizing, processing, retrieving, and storing data

- **element** - a single value or object in a data structure
- **immutable** - an object whose state cannot be modified after it is created
- **index** - an integer value that indicates the position of a value in a data structure
- **initializer list** - a comma-separated list of values or objects for an array given inside curly braces (`{ }`)
- **one-dimensional (1D) array** - a data structure that holds multiple values of the same data type

Teaching Guide

Warm Up (5 minutes)

Software Engineering Goals

Remarks

> In the previous unit, you learned about the structure and design of classes in Java programs and continued developing your software engineering skills. We will be learning more about how we can organize and represent data in Java programs and continue to use software engineering characteristics and skills to make design decisions and develop algorithms.

Distribute: Give each student a copy of the Unit 3 Guide.

Discuss: Use the Retrieve-Pair-Share strategy to discuss the prompt, then have students note their goals on the Unit 3 Guide.

- *What software engineering characteristics do you want to work on in this unit?*
- *How will you know you grew in this area by the end of the unit?*

Discussion Goal: Students share characteristics and skills that they want to improve and identify goals.

Teaching Tip

Students can choose any characteristic or skill they want to improve regardless of whether or not it is an area for improvement. Students can choose a characteristic that is a strength that they want to further develop as an anchor for upcoming units, or they can choose a characteristic that is an area for improvement.

Activity (35 minutes)

1D Arrays (15 minutes)

Remarks

> We have used variables to store a single value and represent an object's attributes. Software engineers often need to store and process multiple values in their programs.

Do This: Review the lesson objectives.

Discuss: Use Retrieve-Pair-Share to discuss the prompt.

- *How could we store these values in a program based on what we know about Java?*

Discussion Goal: Students suggest creating multiple variables for each value.

Teaching Tip

Ask students additional probing questions to consider the problems that could arise with maintaining multiple variables. For example:

- *Is this efficient?*
- *Is this easy to work with and maintain?*
- *What would be the ideal way to do this?*

Do This: Define *data structure*.

Display: Show the video – *One-Dimensional (1D) Arrays*.

Do This: Define *one-dimensional array*.

Discuss: Use Retrieve-Pair-Share to discuss the prompt.

- *What are some physical examples that are similar to a one-dimensional array?*

Discussion Goal: Students suggest physical items that can store multiple items, such as a storage facility, school lockers, or a bike rack.

Do This: Click through the animated slides to explain how arrays are declared and initialized:

- Declare an array
- Initialize the array
- Declare and initialize in one line

Teaching Tip

Have students recall `null`, and ask students what the `scores` variable points to when it is declared but not initialized. Use the visual on the slide to remind students that declaring a reference variable but not initializing it means its value is `null`.

Do This: Click through the animated slide to demonstrate the default values that get stored in an array when it is declared and initialized using the `new` keyword.

Do This: Click through the animated slide to define *element* and *index*.

Exploring 1D Arrays (10 minutes)

Remarks

> One-dimensional arrays are useful for storing multiple elements that can be accessed with one variable. 1D arrays are objects, so we can declare and initialize them using the new keyword. There is also another way to declare and initialize a 1D array.

Group: Place students in pairs.

Distribute: Give each pair a copy of the Exploring 1D Arrays handout.

Do This: Direct students to Level 1 on Code Studio. Students work with their partners to explore the program and complete the Exploring 1D Arrays handout.

1 **Investigate: 1D Arrays**

Teaching Tip

Encourage students to make modifications beyond the prompts to explore the functionality of 1D arrays. Have students consider other questions they might want to answer and modify the code to explore these questions.

Do This: Explain the syntax and functionality of an initializer list.

Do This: Click through the animated slide to explain the syntax for accessing the size of a 1D array.

Teaching Tip

Have students identify the indices for each value in the `scores` array.

Do This: Click through the animated slide to define *immutable*.

Using 1D Arrays (10 minutes)

Remarks

> We have now seen two ways that we can declare and initialize a 1D array. We can create a 1D array that is automatically filled with the default values for the declared data type or a specific list of values.

Do This: Direct students to Level 2 on Code Studio to complete Levels 2 and 3. Students declare and initialize a 1D array using the `new` keyword on Level 2. On Level 3, students complete a choice level to declare and initialize a 1D array using an initializer list and print the length of the array.

2-3 **Using 1D Arrays**

2 3

Wrap Up (5 minutes)

Solve the Problem

Discuss: *Now that we know about 1D arrays, how can we store these values in a program?*

Discussion Goal: Students suggest declaring and initializing a 1D array using an initializer list with the values.

Do This: Click through the animated slide to demonstrate a solution.

Do This: Review the concepts covered in this lesson.

Display: Key Vocabulary

Assessment: Check for Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

4 **Check for Understanding**

AP Classroom Topic Questions

To assign questions from the AP Classroom Question Bank that align with this lesson, create a custom quiz in AP Classroom by searching the Question Bank for the Essential Knowledge statements listed at the top of this lesson plan. You can find instructions and video demonstrations to do this on **AP Central**.

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes **contact us**.