6.1 Array/vector concept (general)

A typical variable stores one data item, like the number 59 or the character 'a'. Instead, sometimes a *list* of data items should be stored. Ex: A program recording points scored in each quarter of a basketball game needs a list of 4 numbers. Requiring a programmer to declare 4 variables is annoying; 200 variables would be ridiculous. An ***array*** is a special variable having one name, but storing a list of data items, with each item being directly accessible. Some languages use a construct similar to an array called a ***vector***. Each item in an array is known as an ***element***.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.1.1: Sometimes a variable should store a list, or array, of data items.

Animation content:

Static Figure: Variable numPlayers stores the number 12 and the variable pointsPerQuarter is an array that stores 4 items.

Step 1: A variable usually stores just one data item.

Step 2: Some variables should store a list of data items, like variable pointsPerQuarter that stores 4 items. Elements 22, 19, 12, and 28 are stored in pointsPerQuarter. Element 22 is stored at location number 0, element 19 is stored at location number 1, element 12 is stored at location number 2, and element 28 is stored at location number 3.

Step 3: Each element is accessible, like the element numbered 3. The answer to the question "How many points in 4th quarter?" is 28 because element 28 is the fourth item in pointsPerQuarter and is located at location number 3.

Animation captions:

1. A variable usually stores just one data item.
2. Some variables should store a list of data items, like variable pointsPerQuarter that stores 4 items.
3. Each element is accessible, like the element numbered 3.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

You might think of a normal variable as a truck, and an array variable as a train. A truck has just one car for carrying "data", but a train has many cars, each of which can carry data.

Figure 6.1.1: A normal variable is like a truck, whereas an array variable is like a train.

Source: Truck (Ian Britton / freefoto.com), Train (Ian Britton / freefoto.com)

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

In an array, each element's location number is called the ***index***, myArray[2] has index 2. An array's key feature is that the index enables direct access to any element, as in myArray[2]; different languages may use different syntax, like myArray(3) or myVector.at(3). In many languages, indices start with 0 rather than 1, so an array with 4 elements has indices 0, 1, 2, and 3.

PARTICIPATION ACTIVITY 6.1.2: Update the array's data values.

Start

Update myItems with the given code.

myItems

0	14
1	27
2	68
3	45
4	74
5	35
6	37

1	2	3	4	5	6

Check Next

PARTICIPATION ACTIVITY 6.1.3: Array basics.

Array peoplePerDay has 365 elements, one for each day of the year. Valid accesses are peoplePerDay[0], [1], ..., [364].

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1) Which assigns element 0 with the value 250?

- peoplePerDay[250] = 0
- peoplePerDay[0] = 250
- peoplePerDay = 250

2) Which assigns element 1 with the value 99?

- peoplePerDay[1] = 99
- peoplePerDay[99] = 1

3) What is the value of peoplePerDay[8]?

```
peoplePerDay[9] = 5;
peoplePerDay[8] =
peoplePerDay[9] - 3;
```

- 8
- 5
- 2

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

4) Assume N is initially 1. What is the value of peoplePerDay[2]?

```
peoplePerDay[N] = 15;
N = N + 1;
peoplePerDay[N] = peoplePerDay[N
- 1] * 3;
```

- 15
- 2
- 45

PARTICIPATION ACTIVITY 6.1.4: Arrays with element numbering starting with 0.

Array scoresList has 10 elements with indices 0 to 9, accessed as scoresList[0] to scoresList[9].

1) Assign the first element in scoresList with 77.

Check Show answer

2) Assign the second element in scoresList with 77.

Check Show answer

3) Assign the last element with 77.

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

4) If that array instead has 100 elements, what is the last element's index?

Check Show answer

5) If the array's last index was 499, how many elements does the array have?

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

6.2 Arrays

Array declarations and accessing elements

A programmer commonly needs to maintain a list of items, just as people often maintain lists of items like a grocery list or a course roster. An ***array*** is an ordered list of items of a given data type. Each item in an array is called an ***element***.

Construct 6.2.1: Array declaration.

```
dataType
arrayName[numElements];
```

This statement declares an array having the specified number of elements in memory, each element of the specified data type. The desired number of elements are specified in [ ] symbols.

Terminology note: [ ] are ***brackets***. { } are ***braces***. In an array access, the number in brackets is called the ***index*** of the corresponding element. The first array element is at index 0.

PARTICIPATION ACTIVITY 6.2.1: An array declaration creates multiple variables in memory, each accessible using [ ].

```
#include <iostream>
using namespace std;

int main() {
   int itemCounts[3];

   itemCounts[0] = 122;
   itemCounts[1] = 119;
   itemCounts[2] = 117;

   cout << itemCounts[1];

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation content:

Static figure:
Begin C++ code:

```
#include <iostream>
using namespace std;

int main() {
  int itemCounts[3];

  itemCounts[0] = 122;
  itemCounts[1] = 119;
  itemCounts[2] = 117;

  cout << itemCounts[1];

  return 0;
}
```

End C++ code.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

There is a memory box with four segmented rows contained inside, all of which are empty. The rows are numbered from top to bottom: 96, 97, 98, and 99. There is another box which displays the output, and is also empty.

Step1: An array named itemCounts is declared. The array consists of 3 elements, each of data type int. Rows 97, 98, and 99 are all labeled "itemCounts". Row 97 is labeled "itemCounts[0]". Row 98 is labeled "itemCounts[1]". Row 99 is labeled "itemCounts[2]".

Step 2: An element is accessed with brackets. The number in brackets is called the index of the corresponding element. The line of code, "itemCounts[0] = 122;", is highlighted and 122 is stored into row 97 of the memory box. The line of code, "itemCounts[1] = 119;", is highlighted and 119 is stored into row 98 of the memory box. The line of code, "itemCounts[2] = 117;", is highlighted and 117 is stored into row 99 of the memory box. The line of code "cout << itemCounts[1];" is highlighted, row 98 of the memory box is accessed, and the stored integer (119) is output. Thus, the output display contains the text, "119."

Animation captions:

1. An array named itemCounts is declared. The array consists of 3 elements, each of data type int.
2. An element is accessed with brackets. The number in brackets is called the index of the corresponding element.

PARTICIPATION ACTIVITY 6.2.2: Select the index shown.

Start

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Check Next

PARTICIPATION ACTIVITY 6.2.3: Array basics.

Given:

```
int yearsArr[4];

yearsArr[0] = 1999;
yearsArr[1] = 2012;
yearsArr[2] = 2025;
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1) How many elements in memory does the array declaration create?

- 0
- 1
- 3
- 4

2) What value is stored in yearsArr[1]?

- 1
- 1999
- 2012

3) With what value does `currYear = yearsArr[2]` assign currYear?

- 2
- 2025
- Invalid index

4) With what value does `currYear = yearsArr[3]` assign currYear?

- 3
- 2025
- Invalid index
- Unknown

5) Recall that the array declaration was `int yearsArr[4]`. Is `currYear = yearsArr[4]` a valid assignment?

- Yes, it accesses the fourth element.
- No, yearsArr[4] does not exist.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

6) What is the proper way to access the *first* element in array yearsArr?

- yearsArr[1]
- yearsArr[0]

7) What are the contents of the array if the above code is followed by the statement: yearsArr[0] = yearsArr[2]?

- 1999, 2012, 1999, ?
- 2012, 2012, 2025, ?
- 2025, 2012, 2025, ?

8) What is the index of the *last* element for the following array: `int pricesArr[100];`

- 99
- 100
- 101

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Using an expression for an array index

A powerful aspect of arrays is that the index is an expression. Ex: userNums[i] uses the value held in the int variable `i` as the index. As such, an array is useful to easily lookup the Nth item in a list.

An array's index must be an integer type. The array index cannot be a floating-point type, even if the value is 0.0, 1.0, etc.

The program below allows a user to print the age of the Nth oldest known person to have ever lived. The program quickly accesses the Nth oldest person's age using `oldestPeople[nthPerson - 1]`. Note that the index is `nthPerson - 1` rather than just `nthPerson` because an array's indices start at 0, so the 1st age is at index 0, the 2nd at index 1, etc.

Figure 6.2.1: Array's nth element can be directly accessed using [n-1]: Oldest people program.

```
#include <iostream>
using namespace std;

int main() {
   int oldestPeople[5];
   int nthPerson;       // User input, Nth oldest
person

   oldestPeople[0] = 122; // Died 1997 in France
   oldestPeople[1] = 119; // Died 1999 in U.S.
   oldestPeople[2] = 117; // Died 1993 in U.S.
   oldestPeople[3] = 117; // Died 1998 in Canada
   oldestPeople[4] = 116; // Died 2006 in Ecuador

   cout << "Enter N (1..5): ";
   cin  >> nthPerson;

   if ((nthPerson >= 1) && (nthPerson <= 5)) {
      cout << "The " << nthPerson << "th oldest
person lived ";
      cout << oldestPeople[nthPerson - 1] << "
years." << endl;
   }

   return 0;
}
```

```
Enter N (1..5): 1
The 1th oldest person lived
122 years.

...

Enter N (1..5): 4
The 4th oldest person lived
117 years.

...

Enter N (1..5): 9

...

Enter N (1..5): 0

...

Enter N (1..5): 5
The 5th oldest person lived
116 years.
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.2.4: Nth oldest person program.

1) In the program above, what is the purpose of this if statement:

```
if ((nthPerson >= 1) &&
(nthPerson <= 5)) {
   ...
}
```

- To avoid overflow because nthPerson's data type can only store values from 1 to 5.
- To ensure only valid array elements are accessed because the array oldestPeople only has 5 elements.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.2.5: Array declaration and accesses.

1) Declare an array named myVals that stores 10 items of type int.

Check Show answer

2) Assign variable x with the value stored at index 8 of array myVals.

Check Show answer

3) Assign the second element of array myVals with the value 555.

Check Show answer

4) Assign myVals array element at the index held in currIndex with the value 777.

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

5) Assign tempVal with the myVals array element at the index one after the value held in variable i.

Check Show answer

Loops and arrays

A key advantage of arrays becomes evident when used in conjunction with loops. The program below uses a loop to allow a user to enter 8 integer values, storing those values in an array, and then printing those 8 values.

Figure 6.2.2: Arrays combined with loops are powerful together: User-entered numbers.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8;  // Number of
elements in array
   int userVals[NUM_ELEMENTS];  // User numbers
   int i;                       // Loop index

   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   cout << "You entered: ";
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " ";
   }
   cout << endl;

   return 0;
}
```

```
Enter 8 integer values...
Value: 5
Value: 99
Value: -1
Value: -44
Value: 8
Value: 555555
Value: 0
Value: 2
You entered: 5 99 -1 -44 8
555555 0 2
```

PARTICIPATION ACTIVITY 6.2.6: Array with loops.

Refer to the program above.

1) How many times does each for loop iterate?

- 1
- 8
- Unknown

2) Which one line of code can be changed to allow the user to enter 100 elements?

- `const int NUM_ELEMENTS = 8;`
- `for (i = 0; i < NUM_ELEMENTS; ++i) {`

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Array initialization

An array's elements are not automatically initialized during the variable declaration and should be initialized before being read. A programmer may initialize an array's elements in an array variable declaration by specifying the initial values in braces {}

separated by commas. Ex: `int myArray[3] = {5, 7, 11};` initializes element at index 0 with 5, element at index 1 with 7, and element at index 2 with 11. For larger arrays, a loop may be used for initialization.

Like other variables, the keyword ***const*** may be prepended to an array variable declaration to prevent changes to the array. Thus, `const int YEARS[3] = {1865, 1920, 1964};` followed by `YEARS[0] = 2000;` yields a compiler error.

PARTICIPATION ACTIVITY 6.2.7: Array initialization.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1) Declare an array of ints named myVals with 4 elements each initialized to 10. The array declaration and initialization should be done in a single statement.

Check Show answer

2) Given: int maxScores[4] = {20, 20, 100, 50};. What is maxScores[3]?

Check Show answer

CHALLENGE ACTIVITY 6.2.1: Enter the output for the array.

602398.4584902.qx3zqy7

Start

Type the program's output

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 3;
   int userVals[NUM_ELEMENTS];
   int i;

   userVals[0] = 1;
   userVals[1] = 4;
   userVals[2] = 9;

   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << endl;
   }

   return 0;
}
```

```
1
4
9
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1 2 3 4 5

Check Next

CHALLENGE ACTIVITY 6.2.2: Printing array elements.

Write three statements to print the first three elements of array runTimes. Follow each statement with a newline. Ex: If runTimes = {800, 775, 790, 805, 808}, print:

```
800
775
790
```

Note: These activities will test the code with different test values. This activity will perform two tests, both with a 5-element array (int runTimes[5]). See "How to Use zyBooks".

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Also note: If the submitted code tries to access an invalid array element, such as runTime[9] for a 5-element array, the test may generate strange results. Or the test may crash and report "Program end never reached", in which case the system doesn't print the test case that caused the reported message.

Learn how our autograder works

602398.4584902.qx3zqy7

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 5;
   int runTimes[NUM_ELEMENTS];
   int i;

   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cin >> runTimes[i];
   }

   /* Your solution goes here  */

   return 0;
```

Run

CHALLENGE ACTIVITY 6.2.3: Printing array elements with a for loop. Full screen

602398.4584902.qx3zqy7

Organize the lines of code to create two for loops that print all elements in courseGrades, following each element with a spa (including the last). Print forwards, then backwards. End each loop with a newline.

Ex: If the input is `7 9 11 10`, then the output is:

```
7 9 11 10

10 11 9 7
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

How to use this tool

Unused

```
for (i = NUM_VALS - 1; i >= 0; --i) {
cout << endl;
cout << courseGrades[i] << " ";
for (i = 0; i < NUM_VALS; ++i) {
```

main.cpp Load default templ

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_VALS = 4;
   int courseGrades[NUM_VALS];
```

```
}
}
cout << endl;
cout << courseGrades[i] << " ";
```

```
   int i;

   for (i = 0; i < NUM_VALS; ++i) {
      cin >> courseGrades[i];
   }

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Check

CHALLENGE ACTIVITY | 6.2.4: Arrays.

602398.4584902.qx3zqy7

Start

Declare an array of integers named userRating of size 6.

Ex: If the input is `67 93 91 33 44 31`, then the output is:

`Ratings: 67 93 91 33 44 31`

```
#include <iostream>
using namespace std;

int main() {
   int rating1;
   int rating2;
   int rating3;
   int rating4;
   int rating5;
   int rating6;
   int i;

   /* Variable declaration goes here */

   cin >> rating1;
```

1 2 3 4

Check Next level

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

6.3 Array/vector iteration drill

The following activities can help one become comfortable with iterating through arrays or vectors, before learning to code such iteration.

PARTICIPATION ACTIVITY 6.3.1: Find the maximum value in the array.

Click "Store value" if a new maximum value is seen.

Start

Array Max

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Next value Store value

PARTICIPATION ACTIVITY 6.3.2: Negative value counting in array.

Click "Increment" if a negative value is seen.

Start

Array Counter

Next value Increment

PARTICIPATION ACTIVITY 6.3.3: Array sorting largest value.

Move the largest value to the right-most position. Click "Swap values" if the larger of the two current values is on the left.

Start

Array

Next value Swap values

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

6.4 Iterating through arrays

Iterating through an array using loops

Iterating through arrays using loops is commonplace and is an important programming skill to master.

Because array indices are numbered 0 to N - 1 rather than 1 to N, programmers commonly use this for loop structure:

Figure 6.4.1: Common for loop structure for iterating through an array.

```
// Iterating through myArray
for (i = 0; i < numElements; ++i)
{
   // Loop body accessing
myArray[i]
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Note that index variable i is initialized to 0, and the loop expression is i < N rather than i <= N. If N were 5, the loop's iterations would set i to 0, 1, 2, 3, and 4, for a total of 5 iterations. The benefit of the loop structure is that each array element is accessed as myArray[i] rather than the more complex myArray[i - 1].

PARTICIPATION ACTIVITY 6.4.1: Iterating through an array.

1) Complete the code to print all items for the given array, using the above common loop structure.

```
int daysList[365];
...

for (i = 0;
                    ; ++i)
{
   cout << daysList[i] <<
endl;
}
```

Check Show answer

2) Given that this loop iterates over all items of the array, how many items are in the array?

```
for (i = 0; i < 99; ++i) {
   cout << someArray[i] <<
endl;
}
```

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Determining a quantity about an array's items

Iterating through an array for various purposes is an important programming skill to master. The program below computes the sum of an array's element values. For computing the sum, the program initializes a variable sum to 0, then simply adds the current iteration's array element value to that sum.

Figure 6.4.2: Iterating through an array example: Program that computes the sum of an array's elements.

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8; // Number of elements
   int userVals[NUM_ELEMENTS]; // User numbers
   int i;                      // Loop index
   int sumVal;                 // For computing sum

   // Prompt user to populate array
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;

   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Determine sum
   sumVal = 0;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      sumVal = sumVal + userVals[i];
   }

   cout << "Sum: " << sumVal << endl;

   return 0;
}
```

```
Enter 8 integer
values...
Value: 3
Value: 5
Value: 234
Value: 346
Value: 234
Value: 73
Value: 26
Value: -1
Sum: 920

...

Enter 8 integer
values...
Value: 3
Value: 5
Value: 234
Value: 346
Value: 234
Value: 73
Value: 26
Value: 1
Sum: 922
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Finding the maximum value in an array

Programs commonly iterate through arrays to determine some quantity about the array's items. The program below determines the maximum value in a user-entered list. If the user enters numbers 7, -9, 55, 44, 20, -400, 0, 2, then the program will output "max: 55". The program uses the variable maxVal to store the largest value seen "so far" as the program iterates through the array. During each iteration, if the array's current element value is larger than the max seen so far, the program assigns maxVal with the array element.

Before entering the loop, maxVal must be initialized to some value because max will be compared with each array element's value. A logical error would be to initialize maxVal to 0, because 0 is not in fact the largest value seen so far, and would result in incorrect output (of 0) if the user entered all negative numbers. Instead, the program peeks at an array element (using the first element, though any element could have been used) and initializes maxVal to that element's value.

Figure 6.4.3: Iterating through an array example: Program that finds the max item.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8;  // Number of elements
   int userVals[NUM_ELEMENTS];  // Array of user numbers
   int i;                       // Loop index
   int maxVal;                  // Computed max

   // Prompt user to populate array
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;

   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Determine largest (max) number
   maxVal = userVals[0];        // Largest so far

   for (i = 0; i < NUM_ELEMENTS; ++i) {
      if (userVals[i] > maxVal) {
         maxVal = userVals[i];
      }
   }
   cout << "Max: " << maxVal << endl;

   return 0;
}
```

```
Enter 8 integer
values...
Value: 3
Value: 5
Value: 23
Value: -1
Value: 456
Value: 1
Value: 6
Value: 83
Max: 456

...

Enter 8 integer
values...
Value: -5
Value: -10
Value: -44
Value: -2
Value: -27
Value: -9
Value: -27
Value: -9
Max: -2
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.4.2: Array iteration.

Given an integer array myVals of size N_SIZE (i.e. int myVals[N_SIZE]), complete the code to achieve the stated goal.

1) Determine the minimum number in the array, using the same initialization as the maximum number example above.

```
minVal = [          ] ;
for (i = 0; i < N_SIZE; ++i) {
   if (myVals[i] < minVal) {
      minVal = myVals[i];
   }
}
```

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

2) Count how many negative numbers exist in the array.

```
cntNeg = 0;
for (i = 0; i < N_SIZE; ++i) {
   if ( [          ] ) {
      ++cntNeg;
   }
}
```

Check Show answer

3) Count how many odd numbers exist in the array.

```
cntOdd = 0;
for (i = 0; i < N_SIZE; ++i) {
   if ( (myVals[i] % 2) == 1 )
{
         [          ] ;
   }
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Check Show answer

zyDE 6.4.1: Print the sum and average of an array's elements.

Modify the program to print the average (mean) as well as the sum. Hint: You don't actually have to change the loop, but rather change what you print.

Load default template...

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS
   int userVals[NUM_ELEMEN
   int i;
   int sumVal;

   // Prompt user to popu
   cout << "Enter " << NUM

   for (i = 0; i < NUM_ELE
      cout << "Value: " <<
      cin >> userVals[i];
```

3 5 234 346 234 73 26 -1

Run

zyDE 6.4.2: Print selected elements of an array.

Modify the program to instead just print each number that is greater than 21.

Load default template...

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS
   int userVals[NUM_ELEMEN
   int i;
   int sumVal;
```

3 5 234 346 234 73 26 -1

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Run

```
10    // Prompt user to popu
11    cout << "Enter " << NUM
12
13    for (i = 0; i < NUM_ELE
14       cout << "Value: " <<
15       cin >> userVals[i];
```

Common error: Accessing out of range array element

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

A common error is to try to access an array with an index that is out of the array's index range. Ex: Trying to access highScores[8] when highScores's valid indices are 0-7. Care should be taken whenever a user enters a number that is then used as an array index, and when using a loop index as an array index also, to ensure the index is within the array's valid index range. Checking whether an array index is in range is very important. Trying to access an array with an out-of-range index is not only a very common error, but is also one of the hardest errors to debug. The following animation shows what happens when a program writes to an out-of-range index using an array.

PARTICIPATION ACTIVITY 6.4.3: Writing to an out-of-range index using an array.

```
int userWeights[3];
int userAge;

userAge = 44;

userWeights[0] = 122;
userWeights[1] = 119;
userWeights[2] = 117;
userWeights[3] = 199; // (Problematic)

// Print userAge
```

Animation content:

Static figure:
Begin code.
int userWeights[3];
int userAge;

userAge = 44;

userWeights[0] = 122;
userWeights[1] = 119;
userWeights[2] = 117;
userWeights[3] = 199; *// (Problematic)*

// Print userAge
End code.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

There is a memory box with five segmented rows contained inside, all of which are empty. The rows are numbered from top to bottom: 96, 97, 98, 99, and 100. Rows 97, 98, and 99 are all labeled "userWeights". Row 97 is labeled "userWeights[0]". Row 98 is labeled "userWeights[1]". Row 99 is labeled "userWeights[2]". Row 100 is labeled "userAge". There is another box which displays the output, and is also empty.

Step 1: Variable userAge is allocated a location in memory immediately after the array userWeights. userAge is assigned with 44. The line of code, "userAge = 44;", is highlighted and 44 is stored into row 100 of the memory box.

Step 2: Each element of array userWeights is assigned a value. The line of code, "userWeights[0] = 122;", is highlighted and 122 is stored into row 97 of the memory box. The line of code, "userWeights[1] = 119;", is highlighted and 119 is stored into row 98 of the memory box. The line of code, "userWeights[2] = 117;", is highlighted and 117 is stored into row 99 of the memory box.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Step 3: Index 3 is out of userWeights's index range, and results in overwriting userAge's value. The line of code, "userWeights[3] = 199; // (Problematic)", is highlighted and 199 is stored into row 100 of the memory box, replacing the previously stored value, 44. The line of code "// Print userAge" is highlighted, row 100 of the memory box is accessed, and the stored integer (199) is output. Thus, the output display contains the text, "199."

Animation captions:

1. Variable userAge is allocated a location in memory immediately after the array userWeights. userAge is assigned with 44.
2. Each element of array userWeights is assigned a value.
3. 3 is out of userWeights's index range, and results in overwriting userAge's value.

A write to an array with an out-of-range index may simply write to a memory location of a different variable X residing next to the array in memory. Later, when the program tries to read X, the program encounters incorrect data. For example, a program may write X with the number 44, but when reading X later in the program X may be 2533, with X never (intentionally) written by any program statement in between.

PARTICIPATION ACTIVITY 6.4.4: Iterating through an array.

Given the following code:

```
const int NUM_ELEMENTS = 5;
int myArray[NUM_ELEMENTS];
int i;
```

1) The normal for loop structure iterates as long as: `i <= NUM_ELEMENTS`

- True
- False

2) To compute the sum of elements, a reasonable statement preceding the for loop is: `sumVal = 0;`

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

3) To find the maximum element value, a reasonable statement preceding the for loop is: `maxVal = 0;`

- True

False

CHALLENGE ACTIVITY 6.4.1: Enter the output for the array elements.

602398.4584902.qx3zqy7

Start

Type the program's output

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 3;
   int userVals[NUM_ELEMENTS];
   int i;

   userVals[0] = 2;
   userVals[1] = 6;
   userVals[2] = 7;

   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << endl;
   }

   return 0;
}
```

```
2
6
7
```

1 2 3 4 5 6

Check Next

CHALLENGE ACTIVITY 6.4.2: Finding values in arrays. Full screen

602398.4584902.qx3zqy7

Integer matchValue is read from input. Then, array userValues is assigned with NUM_VALS integers read from input. Assign numMatches with the number of elements in userValues that equal matchValue.

Ex: If the input is `2 2 1 2 2`, then the output is:

`matchValue: 2, numMatches: 3`

How to use this tool

Unused

```
if (userValues[i] == matchValue) {
for (i = 0; i < NUM_VALS; ++i) {
numMatches = 0;
}
}
numMatches = numMatches + 1;
```

main.cpp Load default templ

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_VALS = 4;
   int userValues[NUM_VALS];
   int i;
   int matchValue;
   int numMatches;

   cin >> matchValue;
   for (i = 0; i < NUM_VALS; ++i) {
      cin >> userValues[i];
   }
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
cout << "matchValue: " << matchValue << ", numMa
```

Check

CHALLENGE ACTIVITY 6.4.3: Populating an array with a for loop.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Write a for loop to populate array userGuesses with NUM_GUESSES integers. Read integers using cin. Ex: If NUM_GUESSES is 3 and user enters 9 5 2, then userGuesses is {9, 5, 2}.

Learn how our autograder works

602398.4584902.qx3zqy7

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_GUESSES = 3;
   int userGuesses[NUM_GUESSES];
   int i;

   /* Your solution goes here  */

   for (i = 0; i < NUM_GUESSES; ++i) {
     cout << userGuesses[i] << " ";
   }

   return 0;
```

Run

CHALLENGE ACTIVITY 6.4.4: Array iteration: Sum of excess.

Full screen

602398.4584902.qx3zqy7

Array testGrades contains NUM_VALS test grades read from input. Organize the lines of code to create a for loop that sets sumExtra to the total extra credit received. Full credit is 100, so any grade over 100 has extra credit.

Ex: If the input is `101 83 107 90`, then the output is:

```
sumExtra: 8
```

because 1 + 0 + 7 + 0 is 8.

Note: This activity includes distractors. Not all lines of code on the left will be used in the final solution.

How to use this tool

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Unused

```
}
}
sumExtra = 0;
sumExtra = sumExtra + (testGrades[i] - 100);
for (i = 0; i < NUM_VALS; ++i) {
```

main.cpp Load default templ

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_VALS = 4;
   int testGrades[NUM_VALS];
   int i;
   int sumExtra;
```

```
for (i = 0; i <= NUM_VALS; ++i) {
if (testGrades[i] < 100) {
if (testGrades[i] > 100) {
```

```
   for (i = 0; i < NUM_VALS; ++i) {
      cin >> testGrades[i];
   }

   cout << "sumExtra: " << sumExtra << endl;

   return 0;
}
```

Check

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

CHALLENGE ACTIVITY 6.4.5: Printing array elements separated by commas.

Write a for loop to print all NUM_VALS elements of array hourlyTemp. Separate elements with a comma and space. Ex: If hourlyTemp = {90, 92, 94, 95}, print:

```
90, 92, 94, 95
```

Your code's output should end with the last element, without a subsequent comma, space, or newline.

Learn how our autograder works

602398.4584902.qx3zqy7

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_VALS = 4;
   int hourlyTemp[NUM_VALS];
   int i;

   for (i = 0; i < NUM_VALS; ++i) {
      cin >> hourlyTemp[i];
   }

   /* Your solution goes here  */

   cout << endl;
```

Run

6.5 Multiple arrays

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Programmers commonly use multiple same-sized arrays to store related lists. The program below maintains a list of letter weights in ounces, and another list indicating the corresponding postage cost for first class mail (usps.com).

The `if (userLetterWeight <= letterWeights[i])` statement compares the user-entered letter weight with the current element in the letterWeights array. If the entered weight is less than or equal to the current element in the letterWeights array, the program prints the element in postageCosts at the same index.

The loop's expression `(i < NUM_ELEMENTS) && (!foundWeight)` depends on the value of the variable foundWeight. This expression prevents the loop from iterating through the entire array once the correct letter weight has been found. Omitting the check for found from the loop expression would result in an incorrect output; the program would incorrectly print the postage cost for all letter weights greater than the user's letter weight.

Figure 6.5.1: Multiple array example: Letter postage cost program.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main () {
   const int NUM_ELEMENTS = 14;
   // Weights in ounces
   double letterWeights[NUM_ELEMENTS] = {1.0, 2.0, 3.0, 3.5, 4.0, 5.0, 6.0,
                                         7.0, 8.0, 9.0, 10.0, 11.0, 12.0,
13.0};
   // Costs in cents (usps.com 2017)
   int postageCosts[NUM_ELEMENTS] = {49, 70, 91, 112, 161, 182, 203,
                                     224, 245, 266, 287, 308, 329, 350};
   double userLetterWeight;
   bool foundWeight;
   int i;

   // Prompt user to enter letter weight
   cout << "Enter letter weight (in ounces): ";
   cin >> userLetterWeight;

   // Postage costs is based on smallest letter weight greater than
   // or equal to mailing letter weight
   foundWeight = false;

   for (i = 0; (i < NUM_ELEMENTS) && (!foundWeight); ++i) {
      if (userLetterWeight <= letterWeights[i]) {
         foundWeight = true;
         cout << "Postage for USPS first class mail is ";
         cout << postageCosts[i] << " cents" << endl;
      }
   }

   if( !foundWeight ) {
      cout << "Letter is too heavy for USPS first class mail." << endl;
   }

   return 0;
}
```

```
Enter letter weight (in ounces): 3
Postage for USPS first class mail is 91 cents

...

Enter letter weight (in ounces): 9.5
Postage for USPS first class mail is 287 cents

...

Enter letter weight (in ounces): 15
Letter is too heavy for USPS first class mail.
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.5.1: Multiple arrays in the above postage cost program.

1) letterWeights[0] is 1, meaning the first element of letterWeights and

postageCosts correspond to a weight of 1 ounce.

- True
- False

2) postageCosts[2] represents the cost for a weight of 2 ounces.

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

3) The program fails to provide a cost for a weight of 7.5.

- True
- False

zyDE 6.5.1: Postage calculation.

Improve the program by also outputting "The next higher weight is ___ with a cost of ___ cents".

Load default template...

```
#include <iostream>
using namespace std;

int main () {
   const int NUM_ELEMENTS
   // Weights in ounces
   double letterWeights[NU

   // Costs in cents (usps
   int postageCosts[NUM_EL

   double userLetterWeight
   bool foundWeight;
   int i;

```

3

Run

PARTICIPATION ACTIVITY 6.5.2: Multiple arrays.

1) Using two separate statements, declare two related integer arrays named seatPosition and testScore (in that order) each with 130 elements.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Check Show answer

2) How many total elements are stored within the two arrays int familyAges[50] and double familyHeights[50]?

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

CHALLENGE ACTIVITY 6.5.1: Multiple arrays.

602398.4584902.qx3zqy7

Start

Subtract each element in origList with the corresponding value in offsetAmount. Print each difference followed by a comma (no spaces).

Ex: If origList = {4, 5, 10, 12} and offsetAmount = {2, 4, 7, 3}, print:

```
2,1,3,9,
```

```
#include <iostream>
#include <string.h>
using namespace std;

int main() {
   const int NUM_VALS = 4;
   int origList[NUM_VALS];
   int offsetAmount[NUM_VALS];
   int i;

   cin >> origList[0];
   cin >> origList[1];
   cin >> origList[2];
   cin >> origList[3];

   cin >> offsetAmount[0];
   cin >> offsetAmount[1];
   cin >> offsetAmount[2];
   cin >> offsetAmount[3];

   /* Your code goes here */

   cout << endl;

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1 2

Check Next

CHALLENGE ACTIVITY 6.5.2: Multiple arrays.

602398.4584902.qx3zqy7

Start

Integer array totalNumbersList contains NUM_VALS elements read from input. Assign each element in thirdNumbersList wit corresponding element in totalNumbersList divided by 3.

Ex: If the input is `81 18 27 15 42`, then the output is:

```
27 6 9 5 14
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_VALS = 5;
   int totalNumbersList[NUM_VALS];
   int thirdNumbersList[NUM_VALS];
   int i;

   for (i = 0; i < NUM_VALS; ++i) {
      cin >> totalNumbersList[i];
   }

   /* Your code goes here */

```

1 2 3

Check Next level

6.6 Using a loop to modify or copy an array

Modifying array elements

A program may need to modify elements while iterating through an array. The program below uses a loop to convert any negative array element value to 0.

Figure 6.6.1: Modifying an array during iteration example: Converting negatives to 0 program.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8;  // Number of elements
   int userVals[NUM_ELEMENTS];  // User values
   int i;                       // Loop index

   // Prompt user to input values
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Convert negatives to 0
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      if (userVals[i] < 0) {
         userVals[i] = 0;
      }
   }

   // Print numbers
   cout << "New numbers: ";
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " ";
   }

   return 0;
}
```

```
Enter 8 integer
values...
Value: 5
Value: 67
Value: -5
Value: -4
Value: 5
Value: 6
Value: 6
Value: 4
New numbers: 5 67 0 0 5
6 6 4
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.6.1: Modifying an array in a loop.

What is the resulting array contents, assuming each question starts with an array of size 4 having contents -55, -1, 0, 9?

1)
```
for (i = 0; i < 4; ++i) {
   itemsList[i] = i;
}
```

- -54, 0, 1, 10
- 0, 1, 2, 3
- 1, 2, 3, 4

2)
```
for (i = 0; i < 4; ++i) {
   if (itemsList[i] < 0) {
      itemsList[i] =
itemsList[i] * -1;
   }
}
```

- -55, -1, 0, -9
- 55, 1, 0, -9
- 55, 1, 0, 9

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

3)
```
for (i = 0; i < 4; ++i) {
   itemsList[i] =
itemsList[i+1];
}
```

- -1, 0, 9, 0
- 0, -55, -1, 0
- Out-of-range access

4)
```
for (i = 0; i < 3; ++i) {
   itemsList[i] =
itemsList[i+1];
}
```

- -1, 0, 9, 9
- Out-of-range access
- -1, 0, 9, 0

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

5)
```
for (i = 0; i < 3; ++i) {
   itemsList[i+1] =
itemsList[i];
}
```

- -55, -55, -55, -55
- -55, -55, -1, 0
- Out-of-range access

zyDE 6.6.1: Modifying an array during iteration example: Doubling element values.

Complete the following program to double each number in the array.

Load default template...

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS
   int userVals[NUM_ELEMEN
   int i;

   // Prompt user to inpu
   cout << "Enter " << NUM
   for (i = 0; i < NUM_ELE
      cout << "Value: " <<
      cin >> userVals[i];
   }

```

5 67 -5 -4 5 6 6 4

Run

Copying an array

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Copying an array is a common task. Given a second array of the same size, a loop can copy each element one-by-one. Modifications to either array do not affect the other.

Figure 6.6.2: Array copying: Converting negatives to 0 program.

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8;    // Number of elements
   int userVals[NUM_ELEMENTS];    // User numbers
   int copiedVals[NUM_ELEMENTS]; // Copied/modified user
numbers
   int i;                         // Loop index

   // Prompt user for input values
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Copy userVals to copiedVals array
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      copiedVals[i] =  userVals[i];
   }

   // Convert negatives to 0
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      if (copiedVals[i] < 0) {
         copiedVals[i] = 0;
      }
   }

   // Print numbers
   cout << endl << "Original and new values: " << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " became " << copiedVals[i]
<< endl;
   }

   return 0;
}
```

```
Enter 8 integer
values...
Value: 12
Value: -5
Value: 34
Value: 75
Value: -14
Value: 33
Value: 12
Value: -104

Original and new
values:
12 became 12
-5 became 0
34 became 34
75 became 75
-14 became 0
33 became 33
12 became 12
-104 became 0
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.6.2: Array copying.

Given array firstList with size 4 and element values, 33, 44, 55, 66, and array secondList with size 4 and elements values 0, 0, 0, 0.

1) firstList = secondList copies 0s into each firstList element.

- True
- False

2) This loop copies firstList to secondList, so that secondList becomes 33, 44, 55, 66:

```
for (i = 0; i < 4; ++i) {
   secondList[i] = firstList[i];
}
```

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

3) After a for loop copies firstList to secondList, the assignment secondList[0] = 99 will modify both arrays.

- True
- False

4) Given thirdList with size 5 and elements 22, 21, 20, 19, 18, the following causes firstList's values to be 22, 21, 20, 19, 18:

```
for (i = 0; i < 5; ++i) {
   firstList[i] = thirdList[i];
}
```

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

CHALLENGE ACTIVITY 6.6.1: Using a loop to modify or copy an array.

602398.4584902.qx3zqy7

Start

Array numbersData has NUM_VALS integers read from input. For each element in numbersData that is divisible by 3:

- Output the element, followed by ": corrected to a number not divisible by 3" and a newline.
- Assign the element with the element's current value minus 1.

▶ **Click here for example**

Note: (x % 3 == 0) returns true if x is divisible by 3.

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_VALS = 3;
   int numbersData[NUM_VALS];
   int i;

   printf("Original numbers: ");
   for (i = 0; i < NUM_VALS; ++i) {
      cin >> numbersData[i];
      cout << numbersData[i] << " ";
   }
   cout << endl;

```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1 2 3

Check Next level

6.7 Swapping two variables (general)

Sometimes a program must swap values among two variables. ***Swapping*** two variables x and y means to assign y's value to x, and x's value to y. If x is 33 and y is 55, then after swapping x is 55 and y is 33.

A common method for swapping uses a temporary variable. A ***temporary variable*** is a variable used briefly to store a value. To understand the intuition of such temporary storage, consider a person holding a book in one hand and a phone in the other, wishing to swap the items. The person can temporarily place the phone on a table, move the book to the other hand, then pick up the phone.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.7.1: Swap idea: Use a temporary location.

Animation content:

Static figure: A hand holding a book, a table, and another hand holding a phone are displayed. A set of three instructions are displayed as well. The first instruction is to put phone on table. The second instruction is to move book. The third instruction is to pick up phone.

Step 1: A swap between two hands requires a third, temporary place. The first instruction to put phone on table is executed by temporarily placing the phone on the table. The hand that was holding the phone is now empty. The second instruction to move book is executed by moving the book to the empty hand. The hand that was holding the phone is now holding the book and the hand that was holding the book is now empty. The third instruction pick up phone is executed by picking up the phone with the empty hand.

Animation captions:

1. A swap between two hands requires a third, temporary place.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Similarly, swapping two variables can use a third variable to temporarily hold one value while the other value is copied over.

PARTICIPATION ACTIVITY 6.7.2: Swapping two variables using a third temporary variable.

```
int X = 33;
int Y = 55;
int tempVal = 0;

tempVal = X;
X = Y;
Y = tempVal;

// Print X and Y
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation content:

Static figure: Variables X, Y, and tempVal are displayed, along with their respective original and updated values. X's original value is 33 and updated value is 55. Y's original value is 55 and updated value is 33. tempVal's original value is 0 and updated value is 33. Both X and Y's updated values are output on a monitor.

Step 1: 55 is assigned to x. 33 is lost. The swap fails because no temporary variable was used.
Begin C++ code:
int X = 33;
int Y = 55;
int tempVal = 0;

X = Y;
Y = X;

// Print X and Y
End C++ code.
X is initialized with 33, Y is initialized with 55, and tempVal is initialized with 0. X is assigned with Y's value (55). Y is assigned with X's updated value (55). The text "X: 55, Y:55" is output on a monitor.

Step 2: The swap succeeds because x's value is saved before being assigned 55, and then 33 is assigned to y.
Begin C++ code:
int X = 33;
int Y = 55;
int tempVal = 0;

tempVal = X;
X = Y;
Y = tempVal;

// Print X and Y
End C++ code.
X is initialized with 33, Y is initialized with 55, and tempVal is initialized with 0. tempVal is assigned with X's value (33). X is assigned with Y's value (55). Y is assigned with tempVal's updated value (33). The text "X: 55, Y:33" is output on a monitor.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation captions:

1. 55 is assigned to x. 33 is lost. The swap fails because no temporary variable was used.

2. The swap succeeds because x's value is saved before being assigned 55, and then 33 is assigned to y.

PARTICIPATION ACTIVITY 6.7.3: Swap.

To begin, x is 22 and y is 99. What are x and y after the given code?

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1)
```
x = y;
y = x;
```

- x is 99 and y is 22.
- x is 22 and y is 99.
- x is 99 and y is 99.

2)
```
x = y;
y = x;
x = y;
```

- x is 99 and y is 22.
- x is 99 and y is 99.
- x is 22 and y is 22.

3)
```
tempVal = x;
x = y;
y = x;
```

- x is 99 and y is 22.
- x is 99 and y is 99.

4)
```
tempVal = x;
x = y;
y = tempVal;
```

- x is 99 and y is 22.
- x is 99 and y is 99.

If you have studied arrays or vectors (or other kinds of lists), you know that most swaps are actually performed between two list elements. For example, reversing a list with N elements can be achieved by swapping element 1 and N, element 2 and N-1, element 3 and N-2, etc. (stopping at the middle of the list).

PARTICIPATION ACTIVITY 6.7.4: Reversing a list using swaps.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation content:

Static figure: A list of integer elements are displayed. Integers 33, 55, 22, 77, and 11 are displayed, respectively.

Step 1: Swap outermost elements. The outermost elements (33 and 11) are swapped. The list is now 11, 55, 22, 77, and 33.

Step 2: Swap next outermost elements, repeat until reach at middle. The next outermost elements (55 and 77) are swapped. The list is now 11, 77, 22, 55, and 33. There are no next outermost elements left, since the middle has been reached (22).

Animation captions:

1. Swap outermost elements.
2. Swap next outermost elements. Repeat until the middle is reached.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.7.5: Reversing a list using swaps.

1) Using the above approach, how many swaps are needed to reverse this list:
999 888 777 666 555 444 333 222

Check Show answer

6.8 Debugging example: Reversing an array

A common array modification is to reverse an array's elements. One way to accomplish this goal is to perform a series of swaps. For example, starting with an array of numbers 10 20 30 40 50 60 70 80, we could first swap the first item with the last item, yielding 80 20 30 40 50 60 70 10. We could next swap the second item with the second-to-last item, yielding 80 70 30 40 50 60 20 10. The next swap would yield 80 70 60 40 50 30 20 10, and the last would yield 80 70 60 50 40 30 20 10.

With this basic idea of how to reverse an array, we can attempt to write a program to carry out such reversal. Below we develop such a program but we make common mistakes along the way, to aid learning from examples of what not to do.

A first attempt to write a program that reverses an array appears below:

Figure 6.8.1: First program attempt to reverse array: Invalid access out of array bounds.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8; // Number of elements
   int userVals[NUM_ELEMENTS]; // User numbers
   int i;                      // Loop index

   // Prompt user to input values
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Reverse array's elements
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      userVals[i] = userVals[NUM_ELEMENTS - i]; //
Swap
   }

   // Print numbers
   cout << endl << "New values: ";
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " ";
   }

   return 0;
}
```

```
Enter 8 integer values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values: 0 80 70 60 50
60 70 80
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Something went wrong: The program did not reverse the array, and the first element was set to 0. Let's try to find the code that caused the problem.

The first and third for loops are fairly standard, so let's initially focus attention on the middle for loop that does the reversing. The swap statement inside that loop is `userVals[i] = userVals[NUM_ELEMENTS - i]`. When i is 0, the statement will execute `userVals[0] = userVals[8];`. However, userVals has size 8 and thus valid indices are 0..7. userVals[8] does not exist. The program should actually swap elements 0 and 7, then 1 and 6, etc. Thus, let's change the right-side index to `NUM_ELEMENTS - 1 - i`. The revised program is shown below.

Figure 6.8.2: Next program attempt to reverse an array: Doesn't reverse properly; we forgot to swap.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8; // Number of elements
   int userVals[NUM_ELEMENTS]; // User numbers
   int i;                      // Loop index

   // Prompt user to input values
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Reverse array's elements
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      userVals[i] = userVals[NUM_ELEMENTS - 1 - i];
// Swap
   }

   // Print numbers
   cout << endl << "New values: ";
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " ";
   }

   return 0;
}
```

```
Enter 8 integer values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values: 80 70 60 50 50
60 70 80
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

The last four elements are still wrong. To determine what went wrong, we can manually (i.e., on paper) trace the loop's execution.

- i is 0: userVals[0] = userVals[7]. Array now: 80 20 30 40 50 60 70 80.
- i is 1: userVals[1] = userVals[6]. Array now: 80 70 30 40 50 60 70 80.
- i is 2: userVals[2] = userVals[5]. Array now: 80 70 60 40 50 60 70 80.
- i is 3: userVals[3] = userVals[4]. Array now: 80 70 60 50 50 60 70 80.
- i is 4: userVals[4] = userVals[3]. Array now: 80 70 60 50 50 60 70 80. *Uh-oh, where did 40 go?*

We failed to actually swap the array elements, instead the code just copies values in one direction. We need to add code to properly swap. We add a variable tempVal to temporarily hold `userVals[i]` so we don't lose that element's value.

Figure 6.8.3: Program with proper swap: However, the program's output shows the array doesn't change.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8; // Number of elements
   int userVals[NUM_ELEMENTS]; // User numbers
   int i;                      // Loop index
   int tempVal;                // Temp variable for
swapping

   // Prompt user to input values
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Reverse array's elements
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      tempVal = userVals[i];
// Temp for swap
      userVals[i] = userVals[NUM_ELEMENTS - 1 - i];
// First part of swap
      userVals[NUM_ELEMENTS - 1 - i] = tempVal;
// Second complete
   }

   // Print numbers
   cout << endl << "New values: ";
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " ";
   }

   return 0;
}
```

```
Enter 8 integer
values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values: 10 20 30 40
50 60 70 80
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

The new values are not reversed. Again, let's manually trace the loop iterations.

- i is 0: userVals[0] = userVals[7]. Array now: 80 20 30 40 50 60 70 10.
- i is 1: userVals[1] = userVals[6]. Array now: 80 70 30 40 50 60 20 10.
- i is 2: userVals[2] = userVals[5]. Array now: 80 70 60 40 50 30 20 10.
- i is 3: userVals[3] = userVals[4]. Array now: 80 70 60 50 40 30 20 10. *Looks reversed.*
- i is 4: userVals[4] = userVals[3]. Array now: 80 70 60 40 50 30 20 10. *Why are we still swapping?*

Tracing makes clear that the for loop should not iterate over the entire array. The reversal is completed halfway through the iterations. The solution is to set the loop expression to `i < (NUM_ELEMENTS / 2)`.

Figure 6.8.4: Program with correct loop bound: Running the program yields the correct output.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ELEMENTS = 8; // Number of elements
   int userVals[NUM_ELEMENTS]; // User numbers
   int i;                      // Loop index
   int tempVal;                // Temp variable for
swapping

   // Prompt user to input values
   cout << "Enter " << NUM_ELEMENTS << " integer
values..." << endl;
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << "Value: ";
      cin >> userVals[i];
   }

   // Reverse array's elements
   for (i = 0; i < (NUM_ELEMENTS / 2); ++i) {
      tempVal = userVals[i];
// Temp for swap
      userVals[i] = userVals[NUM_ELEMENTS - 1 - i];
// First part of swap
      userVals[NUM_ELEMENTS - 1 - i] = tempVal;
// Second complete
   }

   // Print numbers
   cout << endl << "New values: ";
   for (i = 0; i < NUM_ELEMENTS; ++i) {
      cout << userVals[i] << " ";
   }

   return 0;
}
```

```
Enter 8 integer
values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50
Value: 60
Value: 70
Value: 80

New values: 80 70 60 50
40 30 20 10
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

We should ensure the program works if the number of elements is odd rather than even. Suppose the array has 5 elements (0-4) with values 10 20 30 40 50. `NUM_ELEMENTS / 2` would be 5 / 2 = 2, meaning the loop expression would be i < 2. The iteration when i is 0 would swap elements 0 and 4 (5-1-0), yielding 50 20 30 40 10. The iteration for i = 1 would swap elements 1 and 3, yielding 50 40 30 20 10. The loop would then not execute again because i is 2. So the results are correct for an odd number of elements, because the middle element will just not move.

The mistakes made above are each very common when dealing with loops and arrays, especially for beginning programmers. An incorrect (in this case out-of-range) index, an incorrect swap, and an incorrect loop expression. The lesson is that loops and arrays require attention to detail, greatly aided by manually executing the loop to determine what is happening on each iteration. Ideally, a programmer will take more care when writing the original program, but the above mistakes are quite common.

PARTICIPATION ACTIVITY 6.8.1: Array reversal example.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Questions refer to the problematic example in this section.

1) The first problem was trying to access a non-existent element.

- True
- False

2) The second problem was failing to properly swap, using just this

statement:

```
userVals[i] =
userVals[NUM_ELEMENTS - 1 - i];
// Swap
```

- True
- False

3) The third problem was that the loop did not iterate over all the elements, but rather stopped one short.

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

4) The programmer probably should have been more careful in creating the first version of the program.

- True
- False

6.9 Arrays vs. vectors

C++ supports two kinds of ordered list types.

- *Arrays:* declared as `int myList[10]`, accessed as `myList[i]`.
- *Vectors:* declared as `vector<int> myList(10)`, accessed as `myList.at(i)`.

Arrays have a simpler syntax than vectors, but vectors are safer to use. *Thus, using vectors rather than arrays is good practice.*

Vectors are safer because the access v.at(i) is checked during execution to ensure the index is within the vector's valid range. An array access a[i] involves no such check. *Such checking is important; trying to access an array with an out-of-range index is a very common error, and one of the hardest errors to debug.*

PARTICIPATION ACTIVITY 6.9.1: Writing to an out-of-range index using an array.

```
int userWeights[3];
int userAge;

userAge = 44;

userWeights[0] = 122;
userWeights[1] = 119;
userWeights[2] = 117;
userWeights[3] = 199; // (Problematic)

// Print userAge
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation content:

Static figure:
Begin C++ code:

```
int userWeights[3];
int userAge;

userAge = 44;

userWeights[0] = 122;
userWeights[1] = 119;
userWeights[2] = 117;
userWeights[3] = 199; // (Problematic)

// Print userAge
```

End C++ code.
The elements of userWeights and userAge are shown as their representations in memory.
userWeights[0] is in position 97 in memory, and is set to 122.
userWeights[1] is in position 98 in memory, and is set to 119.
userWeights[2] is in position 99 in memory, and is set to 117.
userAge] is in position 100 in memory, and is set to 199.
199 is printed to the screen.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Step 1: Variable userAge is allocated a location in memory immediately after the array userWeights. userAge is assigned with 44.
The line of code reading "userAge = 44" is highlighted, and userAge is set to the value of 44 at position 100 in memory.
Step 2: Each element of array userWeights is assigned a value.
The line of code reading "userWeights[0] = 122;" is highlighted, and position 97 in memory is set to 122. The code reading "userWeights[1] = 119;" is highlighted, and position 98 in memory is set to 119. The code reading "userWeights[2] = 117;" is highlighted, and position 99 in memory is set to 117.
Step 3: 3 is out of userWeights's index range, and results in overwriting userAge's value.
The code reading "userWeights[3] = 199; // (Problematic)" is highlighted, and position 100 in memory is set to 199, overwriting userAge's original value of 44. userAge is printed to be 119.

Animation captions:

1. Variable userAge is allocated a location in memory immediately after the array userWeights. userAge is assigned with 44.
2. Each element of array userWeights is assigned a value.
3. 3 is out of userWeights's index range, and results in overwriting userAge's value.

As shown above, assigning with an out-of-range index can mysteriously change some other variable's value. Debugging such an error can be a nightmare.

Vectors have more advantages, like resizing during runtime, easy insertion of items at the front or rear, determining vector size, etc., discussed later. Arrays have minor benefits that don't really outweigh drawbacks. Like choosing to not wear seatbelts, choosing to not use vectors may be quite risky.

C++ allows vectors to be accessed using brackets [], but brackets involve no range checking, so a good practice is to use .at() to access vector elements.

PARTICIPATION ACTIVITY 6.9.2: Arrays and vectors.

Given:

```
int arrayList[5];
vector<int> vectorList(5);
```

1) arrayList[6] = 777 will yield a compiler error.

- True
- False

2) vectorList[6] = 777 will yield a compiler error.

- True
- False

3) arrayList[6] = 777 will execute without an error message.

- True
- False

4) vectorList.at(6) = 777 will execute without an error message.

- True
- False

5) vectorList[6] = 777 will execute without an error message.

- True
- False

6) `while ( i < arrayList.size() )` loops while i is less than the array's size.

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

6.10 Two-dimensional arrays

An array can be declared with two dimensions. `int myArray[R][C]` represents a table of int variables with R rows and C columns, so R*C elements total. For example, `int myArray[2][3]` creates a table with 2 rows and 3 columns, for 6 int variables total. Example accesses are `myArray[0][0] = 33;` or `num = myArray[1][2];`.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.10.1: Two-dimensional array.

```
// Define array with size [2][3]

// Write to some elements
myArray[0][0] = 55;
```

Animation content:

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Begin static figure:
Begin C++ code:
// Define array with size [2][3]

// Write to some elements
myArray[0][0] = 55;
myArray[1][1] = 77;
myArray[1][2] = 99;
End C++ code.
A 2 by 3 matrix represents the two-dimensional myArray. Position [0][0] is 55. Position[1][1] is 77. Position [1][2] is 99. All other positions in the matrix are empty. In memory, row 0 is represented by memory positions 90 through 92. Position 90 is set to 55. Row 1 is represented by memory positions 93 through 95. Position 94 is set to 77, and position 95 is set to 99.

Step 1: Conceptually, a two-dimensional array is a table with rows and columns.
A 2 by 3 matrix is shown to represent the two-dimensional myArray. Position [0][0] is 55. Position[1][1] is 77. Position [1][2] is 99. All other positions in the matrix are empty.
Step 2: A two-dimensional array is implemented in a one-dimensional memory by placing each row following the previous row.
In memory, row 0 is represented by memory positions 90 through 92. Position 90 represents myArray[0][0], position 91 represents myArray[0][1], and position 92 represents myArray[0][2]. Row 1 is represented by memory positions 93 through 95. Position 93 represents myArray[1][0], position 94 represents myArray[1][1], and position 95 represents myArray[1][2].
The line of code reading "myArray[0][0] = 55;" is highlighted, and position 90 in memory is set to 55.
The line of code reading "myArray[1][1] = 77;" is highlighted, and position 94 in memory is set to 77.
The line of code reading "myArray[1][2] = 99;" is highlighted, and position 95 in memory is set to 99.

Animation captions:

1. Conceptually, a two-dimensional array is a table with rows and columns.
2. A two-dimensional array is implemented in a one-dimensional memory by placing each row following the previous row.

©zyBooks 09/29/24 11:10 2292451
TXSTATECS1428Vargas-GarzonFall2024

Conceptually, a two-dimensional array is a table with rows and columns. The compiler maps two-dimensional array elements to one-dimensional memory, each row following the previous row, known as ***row-major order***.

Figure 6.10.1: Using a two-dimensional array: A driving distance between cities example.

```
#include <iostream>
using namespace std;

/* Direct driving distances between cities, in miles
*/
/* 0: Boston  1: Chicago  2: Los Angeles */

int main() {
   int cityA;                  // Starting city
   int cityB;                  // Destination city
   int drivingDistances[3][3]; // Driving distances

   // Initialize distances array
   drivingDistances[0][0] = 0;
   drivingDistances[0][1] = 960;  // Boston-Chicago
   drivingDistances[0][2] = 2960; // Boston-Los
Angeles
   drivingDistances[1][0] = 960;  // Chicago-Boston
   drivingDistances[1][1] = 0;
   drivingDistances[1][2] = 2011; // Chicago-Los
Angeles
   drivingDistances[2][0] = 2960; // Los Angeles-
Boston
   drivingDistances[2][1] = 2011; // Los Angeles-
Chicago
   drivingDistances[2][2] = 0;

   cout << "0: Boston  1: Chicago  2: Los Angeles"
<< endl;

   cout << "Enter city pair (Ex: 1 2) -- ";
   cin >> cityA;
   cin >> cityB;

   if ((cityA >= 0) && (cityA <= 2) && (cityB >= 0)
&& (cityB <= 2)) {
      cout << "Distance: " <<
drivingDistances[cityA][cityB];
      cout << " miles." << endl;
   }

   return 0;
}
```

```
0: Boston  1: Chicago  2:
Los Angeles
Enter city pair (Ex: 1 2)
-- 1 2
Distance: 2011 miles.

...

0: Boston  1: Chicago  2:
Los Angeles
Enter city pair (Ex: 1 2)
-- 2 0
Distance: 2960 miles.

...

0: Boston  1: Chicago  2:
Los Angeles
Enter city pair (Ex: 1 2)
-- 1 1
Distance: 0 miles.
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

A programmer can initialize a two-dimensional array's elements during declaration using nested braces, as below. Multiple lines make the rows and columns more visible.

Construct 6.10.1: Initializing a two-dimensional array during declaration.

```
// Initializing a 2D array
int numVals[2][3] = { {22, 44, 66}, {97, 98, 99}
};

// Use multiple lines to make rows more visible
int numVals[2][3] = {
   {22, 44, 66}, // Row 0
   {97, 98, 99}  // Row 1
};
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Arrays of three or more dimensions can also be declared, as in `int myArray[2][3][5]`, which declares a total of 2*3*5 or 30 elements. Note the rapid growth in size -- an array declared as `int myArray[100][100][5][3]` would have

100*100*5*3 or 150,000 elements. A programmer should make sure not to unnecessarily occupy available memory with a large array.

PARTICIPATION ACTIVITY 6.10.2: Two-dimensional arrays.

1) Declare a two dimensional array of integers named dataVals with 4 rows and 7 columns.

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

2) How many total elements are in an array with 4 rows and 7 columns?

Check Show answer

3) How many elements are in the array declared as: char streetNames[20][50];

Check Show answer

4) Write a statement that assigns 99 into the fifth row, third column of array numVals. Note: the first row/column is at index 0, not 1.

Check Show answer

CHALLENGE ACTIVITY 6.10.1: Find 2D array max and min.

Write a loop to iterate from 0 to the last row of milesTracker. Within each row, use a nested loop to iterate from 0 to the last column of milesTracker.

Within the nested loop, if an element is greater than maxMiles, assign maxMiles with the value of the element. Similarly, if an element is less than minMiles, assign minMiles with the value of the element.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Ex: If the input is:

```
-10 20 30 40
```

the output is:

```
Min miles: -10
Max miles: 40
```

Learn how our autograder works

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

602398.4584902.qx3zqy7

```
#include <iostream>
using namespace std;

int main() {
   const int NUM_ROWS = 2;
   const int NUM_COLS = 2;
   int milesTracker[NUM_ROWS][NUM_COLS];
   int i;
   int j;
   int maxMiles = 0;
   int minMiles = 0;
   int value;

   for (i = 0; i < NUM_ROWS; i++){
      for (j = 0; j < NUM_COLS; j++){
```

Run

6.11 Functions with array parameters

Functions commonly have array parameters. The following program uses a function to calculate the average value of the elements within an array of test scores.

Figure 6.11.1: Array parameters in an average test score calculation program.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

double CalculateAverage(double scoreVals[], int
numVals) {
   int index;
   double scoreSum = 0.0;

   for (index = 0; index < numVals; ++index) {
      scoreSum = scoreSum + scoreVals[index];
   }

   return scoreSum / numVals;
}

int main() {
   const int NUM_SCORES = 4;     // Array size
   double testScores[NUM_SCORES];  // User test scores
   int index;
   double averageScore;

   // Prompt user to enter test scores
   cout << "Enter " << NUM_SCORES << " test scores:" <<
endl;
   for (index = 0; index < NUM_SCORES; ++index) {
      cout << "Test score: ";
      cin >> testScores[index];
   }
   cout << endl;

   // Call function to calculate average
   averageScore = CalculateAverage(testScores,
NUM_SCORES);
   cout << "Average test score: "
        << averageScore << endl;

   return 0;
}
```

```
Enter 4 test scores:
Test score: 90.5
Test score: 92.0
Test score: 87.5
Test score: 97.0

Average test score:
91.75
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

The parameter definition uses [] to indicate an array parameter. The function call's argument does not use []. The compiler automatically passes array arguments by passing a pointer to the array, rather than making a copy like for other parameter types, in part to avoid the inefficiency of copying large arrays.

Functions with array parameters, other than C string parameters (discussed elsewhere), need a second parameter indicating the number of elements in the array. Ex: numVals indicates the number of elements within the scoresVals array.

PARTICIPATION ACTIVITY 6.11.1: Functions with array parameters.

1) Passing an array creates a copy of that array within the function.

- True
- False

2) An array is automatically passed by pointer.

- True
- False

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.11.2: Functions with arrays.

1) Complete the FindMin function definition to have an array parameter of type double named arrayVals and second parameter of type int numVals indicating the number of array elements.

```
double FindMin(
                                    ) {
   // ...
}
```

Check Show answer

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

2) Complete the PrintVals function to print each array element on a separate line.

```
void PrintVals(int
arrayVals[], int numElements)
{
   int i ;

   for(i = 0; i <
                    ; ++i) {
      cout << arrayVals[i] <<
endl;
   }
}
```

Check Show answer

3) sellingPrices is an array of type double with 10 elements. Complete the statement to find the lowest price by calling the GetLowestPrice function defined below.

```
GetLowestPrice(double itemPrices[],
int numItems);

lowestPrice = GetLowestPrice(
                    );
```

Check Show answer

Because an array is passed to a function by passing a pointer to the array, a function can modify the elements of an array argument.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.11.3: Functions can modify elements of an array argument.

```
void AdjustScores(double scoreVals[], int numVals,
                  double scoreAdj) {
   int i;

   for (i = 0; i < numVals; ++i) {
```

96	92.0	testScores
97	97.5	
98	99.0	main
99		

```
      scoreVals[i] = scoreVals[i] + scoreAdj;
   }
}

int main() {
   const int NUM_SCORES = 4;        // Array size
   double testScores[NUM_SCORES];   // Test scores

   // ...

   AdjustScores(testScores, NUM_SCORES, 2.0);

   // ...

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation content:

Begin C++ code:

```
void AdjustScores(double scoreVals[], int numVals,
            double scoreAdj) {
  int i;

  for (i = 0; i < numVals; ++i) {
    scoreVals[i] = scoreVals[i] + scoreAdj;
  }
}

int main() {
  const int NUM_SCORES = 4;      // Array size
  double testScores[NUM_SCORES];  // Test scores

  // ...

  AdjustScores(testScores, NUM_SCORES, 2.0);

  // ...

  return 0;
}
```

End C++ code.

Before the function call, the array testScores has values 90.0, 95.5, 97.0, and 92.5. After the function call, the array testScores has values 92.0, 97.5, 99.0, and 94.5.

Step 1: Arrays are automatically passed by pointer. So the address of testScores is passed to the scoreVals parameter. The line of code AdjustScores(testScores, NUM_SCORES, 2.0); is highlighted. The line of code void AdjustScores(double scoreVals[], int numVals, double scoreAdj) { is highlighted. NUM_SCORES is passed to the numVals parameter. 2.0 is passed to the scoreAdj parameter.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Step 2: A function can modify the elements of an array argument. Because scoreVals is a pointer to testScores, the function modifies testScores' elements. The line of code int i; is highlighted. 0 is assigned to i. The lines of code for (i = 0; i < numVals; ++i) {scoreVals[i] = scoreVals[i] + scoreAdj; are highlighted. The first iteration of the loop completes, and the element of testScores at index i is increased by 2.0.

Step 3: Function adds 2.0 to all elements of the array argument. The for loop is highlighted: for (i = 0; i < numVals; ++i) {scoreVals[i] = scoreVals[i] + scoreAdj;}. The for loop iterates to completion, each time increasing the element of testSCores at index i by 2.0.

Animation captions:

1. Arrays are automatically passed by pointer. So the address of testScores is passed to the scoreVals parameter.
2. A function can modify the elements of an array argument. Because scoreVals is a pointer to testScores, the function modifies testScores' elements.
3. Function adds 2.0 to all elements of the array argument.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

The keyword ***const*** can be prepended to a function's array parameter to prevent the function from modifying the parameter. In the following program, the AdjustScores function modifies the array so defines a normal array parameter. The PrintScores and CalculateAverage functions do *not* modify the array so define a const array parameter.

Figure 6.11.2: Normal and const array parameters in test score adjustment and averaging program.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
#include <iostream>
using namespace std;

void AdjustScores(double scoreVals[], int numVals,
                  double scoreAdj) {
   int i;

   for (i = 0; i < numVals; ++i) {
      scoreVals[i] = scoreVals[i] + scoreAdj;
   }
}

void PrintScores(const double scoreVals[], int
numVals) {
   int i;

   for (i = 0; i < numVals; ++i) {
      cout << " " << scoreVals[i];
   }
   cout << endl;
}

double CalculateAverage(const double scoreVals[],
int numVals) {
   int i;
   double scoreSum = 0;

   for (i = 0; i < numVals; ++i) {
      scoreSum = scoreSum + scoreVals[i];
   }

   return scoreSum / numVals;
}

int main() {
   const int NUM_SCORES = 4;     // Array size
   double testScores[NUM_SCORES];  // User test
scores
   int i;
   double averageScore;

   // Prompt user to enter test scores
   cout << "Enter " << NUM_SCORES << " test
scores:" << endl;
   for (i = 0; i < NUM_SCORES; ++i) {
      cout << "Test score: ";
      cin >> testScores[i];
   }
   cout << endl;

   // Print original scores
   cout << "Original test scores: ";
   PrintScores(testScores, NUM_SCORES);

   AdjustScores(testScores, NUM_SCORES, 2.0);

   cout << "Adjusted test scores: ";
   PrintScores(testScores, NUM_SCORES);

   // Call function to calculate average
   averageScore = CalculateAverage(testScores,
NUM_SCORES);
   cout << "Average adjusted test score: "
        << averageScore << endl;

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
Enter 4 test scores:
Test score: 90.0
Test score: 95.5
Test score: 97.0
Test score: 92.5

Original test scores:  90
95.5 97 92.5
Adjusted test scores:  92
97.5 99 94.5
Average adjusted test
score: 95.75
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.11.4: Const array function parameters.

Can the following functions modify the array parameter?

1) `void PrintArray(double userNums[], int numElements)`

- Yes
- No

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

2) `int GetHighScore(int userScores[], const int numHighScores)`

- Yes
- No

3) `int FindIndex(const int searchVals[], int numVal, int key)`

- Yes
- No

CHALLENGE ACTIVITY 6.11.1: Enter the output of the function with array parameters.

602398.4584902.qx3zqy7

Start

Type the program's output

```
#include <iostream>
using namespace std;

void PrintScoreVals(const int scoreVals[], int numVals) {
   int i;

   for (i = 0; i < numVals; ++i) {
      cout << scoreVals[i] << endl;
   }
}

int main() {
   const int NUM_SCORES = 3;
   int quizScores[NUM_SCORES];
   int i;

   for (i = 0; i < NUM_SCORES; ++i) {
      cin >> quizScores[i];
   }

   PrintScoreVals(quizScores, NUM_SCORES);

   return 0;
}
```

Input

```
1
0
8
```

Output

```
1
0
8
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

1 2 3 4

Check Next

CHALLENGE ACTIVITY 6.11.2: Modify an array parameter. Full screen

602398.4584902.qx3zqy7

Organize the lines of code to define a function SwapArrayEnds() that swaps the first and last elements of the function's array parameter.

Ex: If the input is `10 20 30 40`, then the output is:

```
40 20 30 10
```

Note: This activity includes distractors. Not all lines of code on the left will be used in the final solution.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

How to use this tool

Unused

```
inputArray[sizeOfArray - 1] = tempValue;
void SwapArrayEnds(int inputArray[], const int sizeOfArra
void SwapArrayEnds(const int inputArray[], int sizeOfArra
void SwapArrayEnds(int inputArray[], int sizeOfArray) {
inputArray[0] = inputArray[sizeOfArray - 1];
int tempValue;
}
tempValue = inputArray[0];
```

main.cpp Load default templ

```
#include <iostream>
using namespace std;

int main() {
   const int SORT_ARR_SIZE = 4;
   int sortArray[SORT_ARR_SIZE];
   int i;
   int userNum;

   for (i = 0; i < SORT_ARR_SIZE; ++i) {
      cin >> userNum;
      sortArray[i] = userNum;
   }

   SwapArrayEnds(sortArray, SORT_ARR_SIZE);

   for (i = 0; i < SORT_ARR_SIZE; ++i) {
      cout << sortArray[i] << " ";
```

Check

CHALLENGE ACTIVITY 6.11.3: Functions with array parameters.

602398.4584902.qx3zqy7

Start

Six integers are read from input and stored into the array arrToModify. Write a function ChangeFirstValue() that takes an inte array parameter and replaces the first element with 0.

Ex: If the input is `120 130 35 85 145 25`, then the output is:

```
Initial array: 120 130 35 85 145 25
Final array: 0 130 35 85 145 25
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
1  #include <iostream>
2  using namespace std;
3
4  /* Your code goes here */
5
6  void PrintArr(const int arr[], int numVals) {
7     int i;
8
9     for (i = 0; i < numVals; ++i) {
10       cout << arr[i] << " ";
```

```
    }

    cout << endl;
}
```

1 2 3 4

Check Next level

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

6.12 Functions with array parameters: Common errors

For arrays other than C strings, a common error is defining a function with an array parameter without a second parameter indicating the number of array elements. Without a parameter indicating the number of array elements, the function will only work for one specific array size. A programmer may call the function with a larger or smaller array, which may lead to incorrect results.

PARTICIPATION ACTIVITY 6.12.1: Common error: Functions with arrays without a parameter indicating the number of array elements.

```
float GetAvgScore(float scoreVals[]) {
   int i;
   float scoreSum = 0;

   for (i = 0; i < 5; ++i) {
      scoreSum = scoreSum + scoreVals[i];
   }

   return scoreSum / 5.0;
}
```

scoreSum = 0.0 + 90.0 + 95.5 + 97.0 + 80.0 + 85.0
= 447.5

`hwAvg = GetAvgScore(hwScores);` 83.2 ✗ Average of all 7 scores in hwScores should be 81.214

`examAvg = GetAvgScore(examScores);` 89.5 ✗ Average incorrectly includes values outside the array bounds

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation content:

Static figure:
Begin C++ code:
float GetAvgScore(float scoreVals[]) {
  int i;
  float scoreSum = 0;

```
  for (i = 0; i < 5; ++i) {
    scoreSum = scoreSum + scoreVals[i];
  }

  return scoreSum / 5.0;
}
```

End C++ code.

The first call to GetAvgScore passes an array hwScores of size 7 with values 80.0, 85.0, 87.0, 80.0, 84.0, 74.5, and 78.0. to parameter scoreVals. The function returns the average of the first five scores which is 83.2. The average of all 7 scores in hwScores should be 81.214. The second call to GetAvgScore passes an array examScores of size 3 with values 90.0, 95.5, and 97.0 to parameter scoreVals. The function calculates the average using the three values in examScores and the values in the next two address spaces which contains the first two values of hwScores, which comes out to 89.5. The average incorrectly includes values outside of array bounds.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Animation captions:

1. Passing the 7 element array to the GetAvgScore will incorrectly calculate the average of the first 5 elements, not all 7 elements. GetAvgScore does not use a second parameter specifying the number of array elements.
2. For an array with less than 5 elements, the function will access memory outside the array bounds.

PARTICIPATION ACTIVITY | 6.12.2: Function with array missing parameter indicating number of array elements.

1) The GetMinScore() function has a common error of not including a parameter to indicate the number of array elements. Given the arrays preGames, seasonGames, and playoffGames, which function call does not return the minimum value?

```
int GetMinScore(int scoreVals[])
{
   int i;
   int minScore;

   minScore = scoreVals[0];
   for (i = 0; i < 5; ++i) {
      if (scoreVals[i] <
minScore) {
         minScore = scoreVals[i];
      }
   }

   return minScore;
}

int preGames[]     = { 80, 45,
86, 85, 64, 99, 98 };
int seasonGames[]  = { 92, 90,
84, 82, 98 };
int playoffGames[] = { 85, 83,
75, 90, 70, 68, 69 };
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

- GetMinScore(preGames);
- GetMinScore(seasonGames);
- GetMinScore(playoffGames);

2) Which function definition correctly supports arrays of any size?

```
int GetMaxScoreA(int
scoreVals[]) {
   int i;
   int maxScore;

   maxScore = scoreVals[0];
   for (i = 0; i <
sizeof(scoreVals); ++i) {
      if (scoreVals[i] >
maxScore) {
         maxScore =
scoreVals[i];
      }
   }

   return maxScore;
}
```

```
int GetMaxScoreB(int
scoreVals[], int numScores) {
   int i;
   int maxScore;

   maxScore = scoreVals[0];
   for (i = 0; i < numScores;
++i) {
      if (scoreVals[i] >
maxScore) {
         maxScore =
scoreVals[i];
      }
   }

   return maxScore;
}
```

- GetMaxScoreA()
- GetMaxScoreB()

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Another common error is modifying array elements in a function that should not modify the array. Arrays are automatically passed to functions using a pointer to the array. So, a function that modifies an array parameter will modify the array argument passed to the function, and not a local copy.

The FindMaxAbsValueIncorrect() function incorrectly modifies the inputVals array to find the array element with the largest absolute value.

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

Figure 6.12.1: FindMaxAbsValueIncorrect() incorrectly modifies inputVals array to find element with the largest absolute value.

```
#include <iostream>
#include <cmath>

using namespace std;

float FindMaxAbsValueIncorrect(float inputVals[], int numVals) {
   int i;
   float maxAbsVal;

   // Incorrectly updates inputVals to calculate absolute value
   // of array elements
   for (i = 0; i < numVals; ++i) {
      inputVals[i] = fabs(inputVals[i]);
   }

   maxAbsVal = inputVals[0];
   for (i = 0; i < numVals; ++i) {
      if (inputVals[i] > maxAbsVal) {
         maxAbsVal = inputVals[i];
      }
   }

   return maxAbsVal;
}

int main() {
   const int NUM_VALUES = 5;
   // Array of changes in temperatures
   float tempChanges[NUM_VALUES] = {10.0, 0.5, -5.1, -11.2, 3.0};
   float maxAbsChange;
   int i;

   // Print array before function call
   cout << "tempChanges array before function call: ";
   for(i = 0; i < NUM_VALUES; ++i) {
      cout << tempChanges[i] << " ";
   }
   cout << endl;

   // Find the largest temperature change, and print result.
   maxAbsChange = FindMaxAbsValueIncorrect(tempChanges,
NUM_VALUES);
   cout << "Max absolute temperature change: " << maxAbsChange <<
endl;

   // Print array after function call
   cout << "tempChanges array after function call:  ";
   for(i = 0; i < NUM_VALUES; ++i) {
      cout << tempChanges[i] << " ";
   }
   cout << endl;

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
tempChanges array before function call: 10 0.5 -5.1 -11.2 3
Max absolute temperature change: 11.2
tempChanges array after function call:  10 0.5 5.1 11.2 3
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

The FindMaxAbsValue() function performs the same computation but avoids modifying the inputVals array. The FindMaxAbsValue() also defines the inputVals as const, which prevents the function from modifying the parameter. *For arrays that should not be modified in a function, good practice is to define the array parameter as const.*

Figure 6.12.2: FindMaxAbsValue() computes the largest absolute value without modifying the array.

```
#include <iostream>
#include <cmath>

using namespace std;

float FindMaxAbsValue(const float inputVals[], int numVals) {
   int i;
   float maxAbsVal;
   float inputAbsVal;

   maxAbsVal = fabs(inputVals[0]);
   for (i = 0; i < numVals; ++i) {
      inputAbsVal = fabs(inputVals[i]);
      if (inputAbsVal > maxAbsVal) {
         maxAbsVal = inputAbsVal;
      }
   }

   return maxAbsVal;
}


int main() {
   const int NUM_VALUES = 5;
   // Array of changes in temperatures
   float tempChanges[NUM_VALUES] = {10.0, 0.5, -5.1, -11.2, 3.0};
   float maxAbsChange;
   int i;

   // Print array before function call
   cout << "tempChanges array before function call: ";
   for(i = 0; i < NUM_VALUES; ++i) {
      cout << tempChanges[i] << " ";
   }
   cout << endl;

   // Find the largest temperature change, and print result.
   maxAbsChange = FindMaxAbsValue(tempChanges, NUM_VALUES);
   cout << "Max absolute temperature change: " << maxAbsChange <<
endl;

   // Print array after function call
   cout << "tempChanges array after function call:  ";
   for(i = 0; i < NUM_VALUES; ++i) {
      cout << tempChanges[i] << " ";
   }
   cout << endl;

   return 0;
}
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

```
tempChanges array before function call: 10 0.5 -5.1 -11.2 3
Max absolute temperature change: 11.2
tempChanges array after function call:  10 0.5 -5.1 -11.2 3
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

PARTICIPATION ACTIVITY 6.12.3: Functions array parameters.

Based on the function names, should the following functions allow the array parameters to be modified?

1) `void PrintReverse(int inputVals[], int numVals)`

- Yes
- No

2) `void SortArrayAscending(int inputVals[], int numVals)`

- Yes
- No

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

3) `int FindIndexSmallest(int inputVals[], int numVals)`

- Yes
- No

6.13 LAB: Nth from end

This section's content is not available for print.

6.14 LAB: Shift right

This section's content is not available for print.

6.15 LAB: Dice Game

This section's content is not available for print.

6.16 LAB: Contact list

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

A contact list is a place where you can store a specific contact with other associated information such as a phone number, email address, birthday, etc. Write a program that first takes as input an integer N that represents the number of word pairs in the list to follow. Word pairs consist of a name and a phone number (both strings). That list is followed by a name, and your program should output the phone number associated with that name. Importantly, you will include your first name in the list, and if your first name matches the name in the input, your output will also state "*** MY NAME ***" in addition to the phone number.

Ex: If the input is:

```
3 Joe 123-5432 Linda 983-4123 Frank 867-5309
Frank
```

the output is:

```
867-5309
```

You can assume there will be no more than 20 word pairs, and there are no spaces in the names or phone numbers.

Your program must define and call the following function. The return value of GetPhoneNumber is the phone number associated with the specific contact name.

©zyBooks 09/29/24 11:10 2292451
Dhwani Pandit
TXSTATECS1428Vargas-GarzonFall2024

`string GetPhoneNumber(string nameArr[], string phoneNumberArr[], int size, string contactName)`

If the name is not found, the function should return the empty string: "". That's two double quotes with NO space in between.

Do not use vectors.

602398.4584902.qx3zqy7

LAB ACTIVITY | 6.16.1: LAB: Contact list | 0 / 20

main.cpp | Load default template...

```
#include <iostream>
using namespace std;

/* Define your function here */

int main() {
   /* Type your code here */

   return 0;
}

```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

©zyBooks 09/29/24 11:10 2292451
Dhwani Pandit
TXSTATECS1428Vargas-GarzonFall2024

Run program

Input (from above) → main.cpp (Your program) → Output (shown below)

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin working
on this zyLab.
```

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024

©zyBooks 09/29/24 11:10 2292451
Diwas Pandit
TXSTATECS1428Vargas-GarzonFall2024