Lab: Arrays

Tutorial Lab 1: Using an Array

An array is a data structure that groups data together into a collection of **elements**. Each element has its own **index** or position within the array. These elements can be initialized, accessed, modified, and printed. Copy the code below into the text editor on the left. Then click on the `TRY IT` button to see the resulting output.

```
string classes[] = {"Math", "English", "Science", "Social
        Studies", "Spanish"};


classes[4] = "French";


for (int i = 0; i < sizeof(classes)/sizeof(classes[0]); i++) {
  cout << classes[i] << endl;
}
```

Program Summary

1. An array called `classes` is created and initialized with elements (i.e. `"Math"`, `"English"`, etc.).
2. At index 4, `"Spanish"` is modified and replaced with `"French"`. Keep in mind that array indices start at `0`, not `1`.
3. A `for` loop iterates through the elements in the array. It starts at index 0, ends right before index 5, and increments by 1 after each iteration.
4. Each element from index 0 through index 4 gets printed with a newline.

Lab: Vectors

Tutorial Lab 2: Using a Vector

A vector is another data structure that has many of the same functionalities as an array. However, they are more flexible in the functions that they are able to use. These functions include adding and removing elements within the vector, meaning vectors can **dynamically** change their size, something arrays cannot do. Copy the code below into the text editor on the left. Then click on the `TRY IT` button to see the resulting output.

```
vector<string> veggies(0);
veggies.push_back("carrot");
veggies.push_back("tomato");
veggies.push_back("celery");
veggies.push_back("spinach");

veggies.erase(veggies.begin()+1);
veggies.at(1) = "potato";

for (auto a : veggies) {
  cout << a << endl;
}
```

Program Summary

1. A vector called `veggies` is created.
2. `carrot`, `tomato`, `celery`, and `spinach` are added to the vector as elements.
3. The element at index 1 (`tomato`) is removed.
4. The element `potato` replaces the element at index 1, which is currently `celery` since `tomato` was deleted previously.
5. An enhanced `for` loop is used which creates an iterating variable `a` (typed auto) to take on the value of each element.
6. Each element `a` is then printed with a newline.

Lab: 2D Arrays

Tutorial Lab 3: Using a 2D Array

A 2D array is an array inside another array. This data structure can be visually compared to a table where there are rows and columns and each element exists inside each "cell." To access or modify elements, both the row index and column index are needed. Like arrays, 2D arrays are **static** so elements cannot be added or removed after initialization.

```
int coordinates[5][2] = { {-4, 3},
                          {2, 1},
                          {0, -8},
                          {-11, 9},
                          {-5, -7} };

int row = sizeof(coordinates) / sizeof(coordinates[0]);
int col = sizeof(coordinates[0]) / sizeof(int);

for (int i = 0; i < row; i++) {
  for (int j = 0; j < col; j++) {
    if (j == 1) {
      cout << coordinates[i][j] << endl;
    }
    else {
      cout << coordinates[i][j] << " ";
    }
  }
}
```

Code Visualizer

Lab Challenge: 2D Chessboard

2D Chessboard

You are trying to create a chessboard representation using the alphabetical letters `O` and `X`. The `O` represents the lighter spaces while the `X` represents the darker spaces.

```
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
```

So far you have the following code within the text editor to your left:

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

int main() {

  string chessboard[8][8];

  int row = sizeof(chessboard) / sizeof(chessboard[0]);
  int col = sizeof(chessboard[0]) / sizeof(string);

  //add code below this line



  //add code above this line

  for (int i = 0; i < row; i++) {
    for (int j = 0; j < col; j++) {
      if (j == col - 1) {
        cout << chessboard[i][j] << endl;
      }
      else {
        cout << chessboard[i][j];
      }
    }
  }

  return 0;

}
```

Chessboard Challenge

challenge

Assignment:

For this challenge, you will use your knowledge of 2D arrays to produce the chessboard pattern:

```
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
```

Requirement:

Your program **cannot make any changes** to the existing code in the program. If you do, you will **not earn any credit** for this challenge. If you accidentally delete any existing code, you can copy the original code shown above back into your program.

Hint: It is probably much easier to use nested `for` loops in your code to populate the 2D array with either `O` or `X` than to go through each (row, column) index to modify the elements.