CSCI 1320 - Computer Science I: Engineering Applications

Instructor: Maciej Zagrodzki

Fall 2018

Announcements

- Interview grading for Assignment 8 this week
- Lab 9 and Assignment 9 due this Friday and Sunday, respectively. (11/09 and 11/11)
- Practicum: C++, week of December 3rd
- Final Project: assigned after fall break; 2 weeks

- No lecture this Friday

Day's Objectives

- Recap: Arrays as function arguments
- Calling functions with array *elements*
- Driver programs

Arrays as function arguments

Function can be defined to take in array as an argument. Works much differently than in MATLAB.

- A pass-by-array parameter behaves like pass-by-reference.
- What is actually being passed?

  The address of first array element.
- Therefore, no need to return arrays.

pass-by-value
pass-by-reference
pass-by-array

Example declaration:

```
1  void foo(int myArray[], int size);
```

Actually, CAN'T return arrays in C++ !!

ILLEAGAL:

```
1  int foo( )
2  {
3      int someArray[] = {1,2};
4      return someArray;
5  }
```

Array element as function argument

What if we want to pass a single array element as a function argument?

- A single array element has the same properties as a normal variable.
- A function that takes in regular variable(s) can also be called with a single array element.

Example:

foo definition:

```
1  void foo( int x )
2  {
3      cout << x << endl;
4  }
```

Can be called:

foo call:

```
1  ...
2  foo(myArr[3]);
3  ...
```

CANNOT be called with the whole array:

foo call:

```
1  ...
2  foo(myArr);
3  ...
```

Array element as function argument

What if we want to swap two elements of an array?

- Could write a function that takes in an array and two pass-by-value int to indicate indicies for swap.

```
1  void foo(int arr[], int indexA, int indexB)
2  {
3  ...
4  }
```

Then call with an array:

```
1  ...
   foo(myArr, 2, 3); // swap elements 2 and 3
```

Array element as function argument

What if we want to swap two elements of an array?

- **Alternatively:**

```
1  void swapValues(int &var1, int &var2)
2  {
3  ...
4  }
```

Then call with individual array elements:

```
1  foo(myArr[2], myArr[3]); // swap elements 2 and 3
```

- **Takeaway:** A function designed to be used with variables can be used the same way with individual array elements.

Driver program

Programs are often comprised of multiple functions. Sometimes a function can have enough complexity it makes sense to test it outside of the program. In these scenarios a driver program can be used.

- Driver program's whole objective is to test a given function.
- Once function is tested, it can be used within the intended program.

Driver program

Consider the following program:

Complex program

```
1   ...
2   void fillArray(int arr[], int size);
3   int findMax(int arr[], int size);
4
5   int main( )
6   {
7       int score[arrSize];
8       const int arrSize = 5;
9       fillArray(score, arrSize);
10      int  maxArrInt = findMax(score, arrSize);
11      return 0;
12  }
13  void fillArray(int arr[], int size)
14  {
15      // function stub
16   }
17
18  int findMax(int arr[], int size)
19  {
20      // function stub
21      return -99;
22  }
```

- Multiple functions being called from main().
- Write a driver program to just exercise one function.

Driver program for *fillArray* function

Driver Program Example

```
1  ...
2  // function tested
3  void fillArray(int arr[], int size);
4
5  int main( )
6  {
7     // Only declare variables needed to test function
8     int score[arrSize];
9     const int arrSize = 5;
10
11    // Call the function
12    fillArray(score, arrSize);
13
14    // Check the result
15    for(int i = 0; i<arrSize; i++)
16       cout << score[i] << " " ;
17
18    cout << endl;
19
20    return 0;
21 }
22 void fillArray(int arr[], int size)
23 {
24    // start developing function
25 }
```

Okay to add additional functions to aid testing.

Array example

Take the program written in earlier example and modularize it.

Write a program that takes 5 user scores (integers), finds the max, and prints the max to console.

Algorithm:

1. Declare an array to hold 5 integers.
2. ~~Ask a user to enter 5 scores.~~ Generate 5 rand scores.

a) modularize - void fillArray(int arr[], int size)

b) Write a driver program

3. Print the 5 scores back to the user.
4. Find max:

a) modularize - int findMax(int arr[], int size)

b) Write a driver program

   - Assume first element in array holds the largest score and assign that value to new variable (max).
   - Compare each consecutive element of array to check if greater than max. If found, update max with new value.