Arrays and Functions

Last updated 6/19/23

These slides introduce using arrays in functions

Arrays and Functions

- Passing array values
  - Passing array values works just like any other value

```
void fun1 (int zoo);
void fun2 (float * soo);

fun1(foo);                  // passes the value of foo to function
                            // fun1
fun1(myArray[3]);           // passes the value of myArray[3]
                            // to function fun1

fun2(&boo);                 // passes a pointer to boo (the address)
                            // to function fun2
fun2(&myFloatArray[3]);     // passes a pointer to myFloatArray
                            // element 3 (the address)
                            // to function fun2
```

Arrays and Functions

- Passing the whole array to a function
  - If we pass all the elements of a large array to multiple functions, we use up a lot of data memory
  - Instead, we pass the address of the array (by reference)
    - Remember – the name of the array is already the address of the beginning of the array (a pointer)

function declaration

```
void fun3(int ary[ ]);      // the array notation type name[]
                            // tells the compiler it is expecting an
                            // address – equivalent to int * ary
…
```

call

```
fun3(myArray);              // the array name is already an
                            // address
```

C does not have a way to pass a copy of an array to a function

Arrays and Functions

- Accessing the array inside a function
  - We passed a pointer to the array into the function

```
void fun3(int ary[ ]);          // function expecting a pointer
…
fun3(myArray);                  // pass a pointer to the function
```

  - Inside the function, ary has the value of the pointer
    - This is the address of the array
  - To access an element of the array we can use the normal array notation since the name we are using is already a pointer
    - This is the normal array situation

inside the function

```
foo = ary[3];

ary[2] = 5;

scanf("%i", &ary[7]);
```

the pointer ary = the pointer myArray so the index (offsets) point to the correct memory location

Arrays and Functions

- Passing the whole array
  - Array average program

```c
/*
 * array_ave.c
 *
 *  Created on: Dec 18, 2020
 *      Author: johnsontimoj
 */
//////////////////////////
//
// array passing example for class
//
////////////////////////////////
#include <stdio.h>

// Function Prototypes (declaration)
float average(int myArray[ ]);

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // local variables
   float ave;
   int valArray[5] = {3, 7, 4, 3, 2};

   // calculate average
   ave = average(valArray);
   printf("Average is: %f", ave);

   return 0;
}// end main
```

passing the name – a pointer to the function

Expecting a pointer to an array of ints

```c
// Function Definitions
float average(int myArray[ ]){
    int sum = 0;
    int i;

    for(i = 0; i < 5; i++){
        sum += myArray[i];
    }

    return (sum / 5.0);
}// end average
```

Normal array notation using the name as a pointer

Arrays and Functions

- Passing array values
  - What if we want to pass the whole array to a function but we do not want the function to modify the array?
  - Declare the array as a constant in the function declaration and definition

```
float average(int myArray[ ]);                  // modifiable
```

→

```
float average(const int myArray[ ]);            // non-modifiable
```

Arrays and Functions

- Passing the whole array
  - A better array average program

```c
/*
 * array_ave2.c
 *
 *  Created on: Dec 18, 2020
 *      Author: johnsontimoj
 */
//////////////////////////
//
// A better array passing example for class
//
////////////////////////////////
#include <stdio.h>

// Function Prototypes (declaration)
float average(int size, const int myArray[ ]);

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // local variables
   float ave;
   int valArray[5] = {3, 7, 4, 3, 2};

   // calculate average
   ave = average(5, valArray);
   printf("Average is: %f", ave);

   return 0;
}// end main
```

```c
// Function Definitions
float average(int size, const int myArray[ ]){
    int sum = 0;
    int i;

    for(i = 0; i < size; i++){
        sum += myArray[i];
    }

    return (sum / (float)size);      // avoid int divide
}// end average
```

Guarantee the array is not changed

Re-use the function for any size array