EECS168/169-Lab7

Functions

Functions

```
returnValueType functionName(list of parameters)
{
 //Function Body
}
```

Have we used functions so far?

Function - main

```
int main(int argc, char** argv)
{
 //Main Function Body
  return (0);
}
```

What is `main`‘s:

- name,
- parameters,
- return value type
- body

Function - sqrt

```
the_root = sqrt(9.0);
```

What is `sqrt`'s:

- name,
- parameters,
- return value type
- body

```
double sqrt(double Value);
```

Declaration!

Function Declaration, Definition and call(s)

Function Declaration

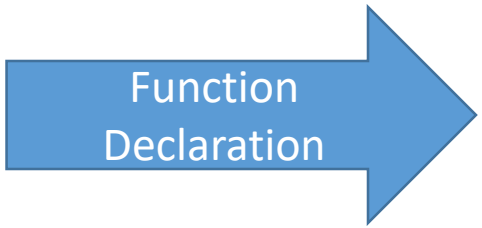
Function Call

Function Definition

Function Definition

```
void changeFirstElement(int arr[], int size);
void printArray(int arr[], int size); /*When you pass an array to a function , what you are
actually passing is the array reference (aka a pointer to the first block of memory in the array).
And since an array doesn't know its own size, the size will need to be passed as well. */


int main()
{
   int *nums = new int[3]; //Defining array Dynamically

   nums[0] = 10;
   nums[1] = 20;
   nums[2] = 30;

   printArray(nums, 3);
   changeFirstElement(nums, 3);

   cout << nums[0] << endl; //prints 99
   printArray(nums, 3);

   delete[] nums; //delete the array. Remember, there is one delete for every new.
}

void changeFirstElement(int arr[], int size)
{
    arr[0] = 99;
}

void printArray(int arr[], int size)
{
    int i = 0;
    for (i = 0; i < size; i++)
        cout << arr[i]
}
```

Function Declaration, Definition and call(s)




```
int add(int a, int b);


int main()
{
        int x = 5;
        int y = 10;
        int z = add(x, y);
        std::cout << z; //prints 15
}

/* This function takes two integers and returns the summing of
those two numbers */
int add(int a, int b)
{
        int answer = 0; //function variable
        answer = a + b; //doing some calculations
        return(answer);  //returning a value
}
```

```
1   #include <iostream>
2   using namespace std;
3
4   double total_cost(int number_par, double price_par);
5   //Computes the total cost, including 5% sales tax,
6   //on number_par items at a cost of price_par each.
7
8   int main( )
9   {
10      double price, bill;
11      int number;
12
13      cout << "Enter the number of items purchased: ";
14      cin >> number;
15      cout << "Enter the price per item $";
16      cin >> price;
17
18      bill = total_cost(number, price);
19
20      cout.setf(ios::fixed);
21      cout.setf(ios::showpoint);
22      cout.precision(2);
23      cout << number << " items at "
24           << "$" << price << " each.\n"
25           << "Final bill, including tax, is $" << bill
26           << endl;
27
28      return 0;
29  }
30
31  double total_cost(int number_par, double price_par)
32  {
33      const double TAX_RATE = 0.05; //5% sales tax
34      double subtotal;
35
36      subtotal = price_par * number_par;
37      return (subtotal + subtotal * TAX_RATE);
38  }
```

function declaration

function call

function heading

function body

function definition

Pitfalls

- In your main function test your functions
- I highly recommend that you **test each function after you write it**
- **Don't** try to write all the functions then test for the **first time**
- Your main **must call the functions you write** to solve the problems at hand
- Arguments in wrong order.
- Putting parameter type in function call.

Command line arguments

You can pass information into your program at the command - in other words, from terminal.
Example:

```
$>./myLab coffee eggs bread
```

In addition to lauching your program, you passed 3 pieces of information, "coffee", "eggs", and "bread" into the program. Where is it? In a 2D character array!
You main() will be update to use command line arguments.

```
//old main
int main ()
{
//stuff
 }

//new main
int main( int argc, char* argv[] )
{
//stuff
}
```

Question: What are argc and argv?

- argc
  - a count of how many command line arguments (include the program's name) were passed in.
  - Our example from above would set argc to 4
- argv
  - A 2D character array with all the words passed in

Command line arguments (cont.)

- Storing one of the arguments in a string:

```
//new main
int main( int argc, char* argv[] )
{
   std::string myStr;
   //check to see if there's an argument to grab
   if( argc > 1 )
   {
      myStr = argv[1]; //copies the argument into your variable
   }
}
```

Exercise 1: Function basics

Your program will two numbers from the command-line and display the sum of all values from one to the other (inclusive).

```
$> ./exercise1 5 8 Summation from 5 to 8: 26
```

Order of the values doesn't matter

```
$> ./exercise1 8 5 Summation from 5 to 8: 26
```

Note the program displays "Summation from <smaller value> to <larger value>"

Define the following functions in main.cpp:

- int smaller(int n1, int n2);
  - returns the smaller of the two values (or either if tied)
- int larger(int n1, int n2);
  - Returns the larger of the two values (or either if tied)
- int sum(int n1, int n2);
  - Returns the sum of values from the smaller value to the larger value
  - Make calls to your previous defined functions to help!
- int main(int argc, char* argv[])
  - Handles all printing!
  - Your other function do not print anything!

Please refer to string to int/double converters in library <string>.

```
std::string str1 = "42";
std::string str2 = "2.5";

int x = 0;
double d = 0;

x = std::stoi(str1);  //string to int
d = std::stod(str2);  //string to double
```

Exercise 2: Your pal, the palindrome

- Make a program that takes a single int at the command-line and displays whether or not that integer is a palindrome.
- A palindrome a sequence that is the same forwards and backwards. Example 121 is a palindrom, but 122 is not.
- Define the following functions
- int lastDigit(int n)
  - Returns the last digit of an integer n (e.g. lastDigit(17) returns 7, lastDigit(1) returns 1)
- int removeLast(int n)
  - Returns the same value as n, but the last digit is removed (e.g. removeLast(123) returns 12)
  - NOTE: if removeLast is passed a 1-digit number it returns 0
- int reverse(int n)
  - Returns the reverse of an integer n (e.g. reverse(12345) returns 54321)
  - Use previous functions for help!
- bool isPalindrome(int n)
  - Returns true if n is a palindrome, false otherwise.
  - Use previous functions for help!

Exercise 3: Array Resizing

- You will create an array manipulation program that allows the user to do pretty much whatever they want to an array.
- When the program begins, you will prompt the user for an initial size of the array, then the values to fill the array. Your array will contain ints.
- After the user fills the initial array, present the user with a menu, detect their choice, and provide them any needed follow up prompts that are needed.
- Continue until they want to quit

Menu Options

Sample Menu

```
Make a selection:
1) Insert
2) Remove
3) Count
4) Print
5) Exit
Choice:
```

Insert	•The user will provide a position to insert a value •You must obtain a valid position before moving on •Obtain the value to insert and insert it into the array •NOTE: The array will be one element larger after
Remove	•The user will provide a position to remove a value •You must obtain a valid position before moving on •Once you have a valid position, remove that value •NOTE: The array will be one element smaller after
Count	•Obtain a value from the user •Tell them how many times that value is in the array
Print	•Print the contents of the array in the following format: [1, 3, 99]
Exit	•Exits the program

Functions

- For each of the options the user has access to, create a function to handle the work involved.
- int* insert(int arr, int size, int value, int position)
  - Inserts the given value at the specified position
  - Creates a new array, copies all old value over adjusting indices as necessary
  - Deletes the old array (arr)
  - Returns a pointer to the new array
- int* remove(int arr[ ], int size, int position)
  - Removes the value at the given position
  - Creates a new array, copies all old value over adjusting indices as necessary
  - Deletes the old array (arr)
  - Returns a pointer to the new array
- int count(int arr[ ], int size, int target)
  - returns a count of how many times the target value is in the array
- void print(int arr[ ], int size)
  - Prints array as required

Exercise -169

- Add an option in the menu "Check palindrome." When the user selects this, the program will tell the user if the array is a palindrome or not. You must create an isPalindrome function with appropriate parameters and return type.
- A palindrome is a sequence that is the same forwards and backwards, so you'll detect whether or not the order of values in the array would be same if reversed.

Memory Leaks

Don’t forget to check the memory leaks!

Thank you!