More Functions

ITP 165 – Fall 2015

Week 7, Lecture 1

Function Parameters – Review

- Two ways to write parameters:
  - Pass-by-value
  - Pass-by-reference
- PBV makes copies of the input
- PBR passes the original input in
  - Only works if you call the function with variables as parameters

Function Creation

- Anything can go in our function!
- Let’s do a simple math calculation of the average of an array of numbers
- So here are the requirements of my function:
  1. Take in an array of `int` values
  2. Calculate the average of these integer values
  3. Return the average as a `double`

Function Creation

- Our function signature then looks like this:

```
//Function: avg
//Purpose: Calculates the average
//of an array of ints
//Parameters: array of ints
//Returns: double containing the average
double avg(int arr[])
{
//calculate average here

}
```

Function Creation

- What is the algorithm to calculate an average, given an array of integer values?

1. Sum up all elements of the integer array
2. Divide by the total number of elements

Function Creation

- How many elements are in the array?
- It depends...
- Arrays don’t know their size after they are declared
  - You must use a specialized function to calculate them

Function Creation

- Whenever we create arrays, we declare their size either explicitly or implicitly

```
int arr[9];
int arr[] = { 1, 3, 4, 1, 4, 19 };
```

Function Creation

- Always know the size of an array at declaration
- Pass it in as a parameter to a function when you pass in the array!

Function Creation

- Our new function signature is:

```
//Function: avg
//Purpose: Calculates the average
//of an array of ints
//Parameters: array of ints, int for size
//of array
//Returns: double containing the average
double avg(int arr[], int size)
{
//calculate average here

}
```

Function Creation

- Now that we have all the necessary information, we can begin our algorithm

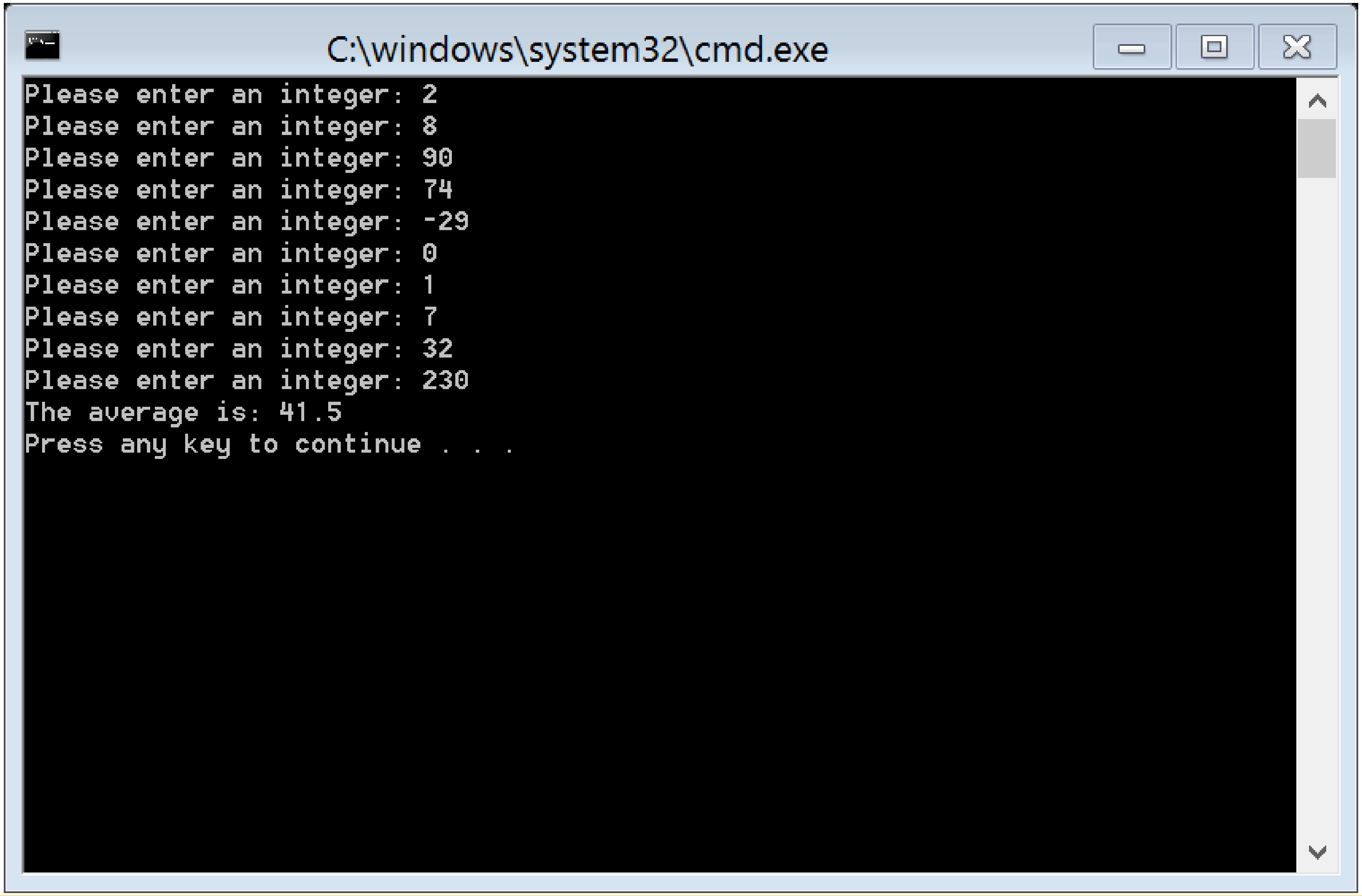
```
double avg(int arr[], int size)
{
    double sum = 0;
    for (int index = 0; index < size; index++)
    {
        sum += arr[index];
    }
    double retVal = sum / size;
    return retVal;
}
```

Function Creation

```
double avg(int arr[], int size) {
    double sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    double retVal = sum / size;
    return retVal;
}

int main(){
    const int SIZE = 10;
    int myArr[SIZE];
    for (int index = 0; index < SIZE; index++) {
        std::cout << "Please enter an integer: ";
        std::cin >> myArr[index];
    }
    std::cout << "The average is: " << avg(myArr, SIZE) << std::endl;
    return 0;
}
```

Function Creation



```
Please enter an integer: 2
Please enter an integer: 8
Please enter an integer: 90
Please enter an integer: 74
Please enter an integer: -29
Please enter an integer: 0
Please enter an integer: 1
Please enter an integer: 7
Please enter an integer: 32
Please enter an integer: 230
The average is: 41.5
Press any key to continue . . .
```

Encrypt Function

- Let’s create a function to encrypt an entire line of text using our substitution cipher method

- Things we need:
  - The message to encrypt
  - The cipher

- Things we get back:
  - The encrypted message

Encrypt Function

- Which of the following signatures should we use?

```
std::string encrypt(std::string cipher, std::string message)
{

}

void encrypt(std::string& cipher, std::string& message)
{

}
```

Encrypt Function

```
//Function: encrypt
//Purpose: encrypt message
//Parameters: string for cipher, and message
//Returns: nothing (PBR)
void encrypt(std::string& cipher, std::string&
message)
{

}
```

Encrypt Function

- What is our algorithm?

1. Calculate the length of the message
2. Grab the first letter
3. If it’s an uppercase letter (A-Z), subtract 65 and replace the letter in the string with the cipher at that index
4. If it’s a lowercase letter (a-z), subtract 97 and replace the letter in the string with the cipher at that index
5. If it’s not an uppercase or lowercase letter, leave it alone
6. Repeat steps 2-5 for the remaining letters

Encrypt Function

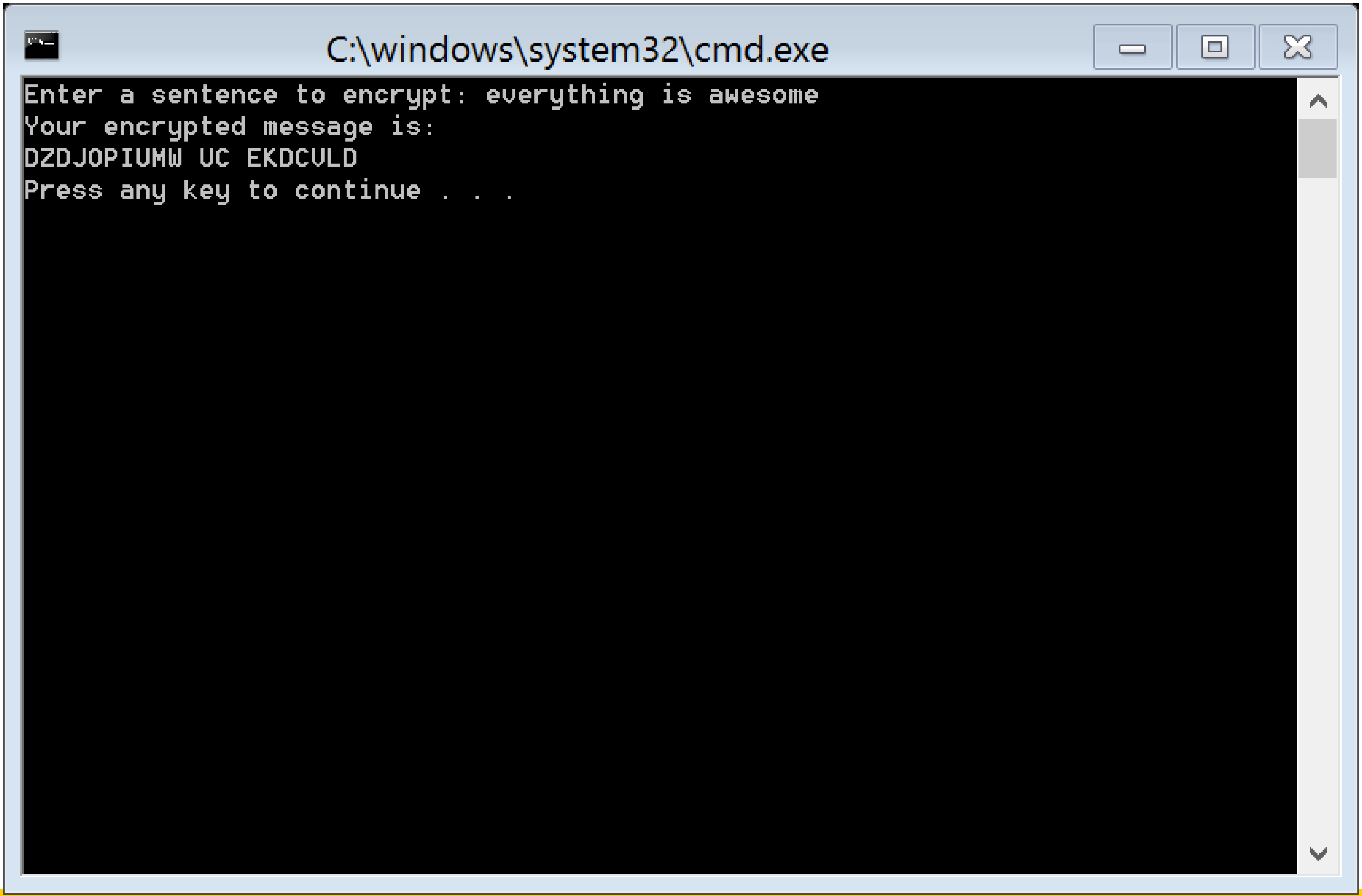
```
void encrypt(std::string& cipher, std::string& message) {
    int len = message.length();
    for (int index = 0; index < len; index++){
        if (message[index] >= 'A' && message[index] <= 'Z'){
            message[index] = cipher[message[index] - 65];
        }
        else if (message[index] >= 'a' && message[index] <= 'z'){
            message[index] = cipher[message[index] - 97];
        }
    }
}
```

Encrypt Function

```cpp
void encrypt(std::string& cipher, std::string& message) {
	int len = message.length();
	for (int index = 0; index < len; index++){
		if (message[index] >= 'A' && message[index] <= 'Z'){
			message[index] = cipher[message[index] - 65];
		}
		else if (message[index] >= 'a' && message[index] <= 'z'){
			message[index] = cipher[message[index] - 97];
		}
	}
}

int main(){
	std::string cipher = "EHTGDBWIUQRXLMVFSJCPYZKAON";
	std::string userIn;
	std::cout << "Enter a sentence to encrypt: ";
	std::getline(std::cin, userIn);
	std::cout << "Your encrypted message is: " << std::endl;
	encrypt(cipher, userIn);
	std::cout << userIn << std::endl;
	return 0;
}
```

Encrypt Function



```
Enter a sentence to encrypt: everything is awesome
Your encrypted message is:
DZDJOPIUMW UC EKDCVLD
Press any key to continue . . .
```

Lab Practical #11