CSCI 1300 CS1: Starting Computing

Naidu/Correll/Yeh/Hoefer - Fall 2021

Recitation 6 - the week of September 27th, 2021

# Functions

This week in lecture we’re learning how to group together multiple statements in functions. Functions are a fundamental building block of programming. They tend to solve a general but narrow problem, they should be easily implemented and understood, and they can be reused.

# Exercises

1. **Spot the Error: Championship Edition**

This week we’ll be doing something a little different in recitation. Below is a snippet of code that is free of errors; try running the file outfit.cpp in VS Code to see for yourselves!

#include <iostream>

using namespace std;

string outfit(double temp);

int main() {

double temp;

char answer;

do{

cout << "What's the temperature outside?" << endl;

cin >> temp;

cout << "You should wear " << outfit(temp) << "!" << endl;

cout << "Would you like another suggestion (y/n)?" << endl;

cin >> answer;

while(answer != 'y' && answer != 'n'){

cout << "Invalid input. Please type only y or n." << endl;

cout << "Would you like another suggestion (y/n)?" << endl;

cin >> answer;

}

}while(answer != 'n');

cout << "Goodbye!" << endl;

return 0;

}

string outfit(double temp){

//suggests an accessory for the weather

string suggestion;

if(temp <= 40){

suggestion = "a heavy coat";

}else if(temp >= 75){

suggestion = "sunglasses";

}else{

suggestion = "a scarf";

}

return suggestion;

}

**2a.** Your TA will begin by separating you into groups. Your task is to work as a team to BREAK this code. Introduce as many **unique** errors as you can, but **be sure to keep track of all of the errors you introduce**. Keep in mind that the errors you introduce **must be unique**. Avoid introducing multiples of the same error.

In about 5 minutes, you’ll give your code to another team for them to try and debug. Whichever team catches the most errors wins! Who doesn’t enjoy a little friendly competition?

Keep a list of the bugs your team introduces here:

| **Error** | **Line number** |
| --- | --- |
|  |  |

**2b.** Now it’s time for you to switch code with your competition. Your team’s job is to identify and fix as many errors as you can before time’s up! **Be sure to keep track of all the errors you find**. In order to win, your team needs to identify more bugs than the other team.

Keep a list of the bugs you’re able to find and fix below:

| **Error** | **Line number** |
| --- | --- |
|  |  |

**2c.** Now, give the list of bugs you found back to the other team. They should do the same for you. Go through the list of bugs the other team found and verify them using your team’s list from earlier. How many bugs did the other team find?

1. **Pumpkin Patch Fertilizer**

It’s near the end of September and you’re a humble pumpkin farmer looking forward to making money as people flock to your fields to pick their own pumpkins for halloween. To make sure that your crop looks its best, you need to keep the pumpkins well fertilized. Design two functions to track the amount of fertilizer you purchase and use. Both functions should take in an amount for your current stock of fertilizer and an amount to be used or added into the stock, and then return your new fertilizer levels. Here are two function headers to get you started:

double fertilize(double stock, double amount)

double restock(double stock, double amount)

You cannot fertilize pumpkins with more than what you have in your stock!

Here’s some example output (no user input):

| **Test Case** | **Expected output** |
| --- | --- |
| double stock = 100;  double amount = 20.5;  cout << restock(stock, amount) << endl; | 120.50 |
| double stock = 51;  double amount = 50;  cout << fertilize(stock, amount) << endl; | 1.00 |
| double stock = 71.4;  double amount = 20;  cout << fertilize(stock, amount) << endl; | 51.40 |

**2a.** Write an algorithm in pseudocode for the program above.

| **Pseudocode:** |
| --- |

**2b.** Let’s draw a flowchart of the solution

| **Flowchart:** |
| --- |

**2c.** Imagine what a sample run of your program would look like. Think about at least two examples

| **Sample Run 1:** |
| --- |

| **Sample Run 2:** |
| --- |

| **Sample Run 3:** |
| --- |

**2d.** Identify the values that you must test for. We call these values “**boundary conditions**”.

| **Answer:** |
| --- |

**2e.** Implement your solution in C++ using VS Code. Revise your solution, save, compile and run it again. Are you getting the expected result and output? Keep revising until you do. Make you sure you test for the values used in your sample runs, and for the boundary conditions.

1. **Pumpkin Patch Fertilizer Menu**

Using the functions from problem 2, create a menu that can track the fertilizer stores of a user by allowing them to fertilize pumpkins and add more fertilizer to their stores. It should take user input to indicate which menu item the user would like to choose, and have an option to exit.

Example output (**bold** is user input)

| Menu   1. Fertilize 2. Stock Up 3. Check Stores 4. Exit   **2**  How much fertilizer?  **22.50**  Restock Successful.  Menu   1. Fertilize 2. Stock Up 3. Check Stores 4. Exit   **3**  Storage level is 22.50 lbs.  Menu   1. Fertilize 2. Stock up 3. Check Stores 4. Exit   **4**  Exiting program. |
| --- |

**3a.** Write an algorithm in pseudocode for the program above.

| **Pseudocode:** |
| --- |

**3b.** Let’s draw a flowchart of the solution

| **Flowchart:** |
| --- |

**3c.** Imagine what a sample run of your program would look like.

| **Sample Run 1:** |
| --- |

**3d.** Implement your solution in C++ using VS Code. Revise your solution, save, compile and run it again. Are you getting the expected result and output? Keep revising until you do. Make sure you test for the values used in your sample runs, and for the boundary conditions.

1. Spot the errors

a)

#include <iostream>

#include <string>

using namespace std;

int address(string street, string apartment)

{

string s = "You stay in APT " + apartment + " on " + street + " St.";

}

// Main function prints the address

int main()

{

cout<<address("Pearl","123")<<endl;

}

b)

#include <iostream>

using namespace std;

int main()

{

cout << avg(8,13) << endl;

return 0;

}

int avg(int a, int b)

{

int x = (a+b) / 2.0;

return 0;

}

c)

#include <iostream>

#include <cmath>

using namespace std;

int pythagoras(int side1, int side2)

{

cout << "Enter side 1: " << endl;

cin >> side1;

cout << "Enter side 2: " << endl;

cin >> side2;

double hypotenuse = sqrt(pow(side1, 2) + pow(side2, 2));

return hypotenuse;

}

int main()

{

cout << pythagoras(5,6) << endl;

return 0;

}

d)

#include <iostream>

using namespace std;

string concatenate( a="default", b);

int main()

{

cout << concatenate(‘csci’,1300) << endl;

return 0;

}

string concatenate(string a, string b)

{

string c = a + b;

return c;

}

e)

#include <iostream>

using namespace std;

void justKeepLooping(char letter);

int main(){

justKeepLooping(^);

return 0;

}

void justKeepLooping(char letter){

//prints out 5 ‘^’ characters each on a new line

int i = 10;

do{

cout << letter << endl;

i += 2;

}while(i > 1)

return;

}

f)

#include <iostream>

using namespace std;

string hello();

void kenobi(string name);

int main(){

hello();

return 0;

void hello(){

cout << "Hello there!" << endl;;

kenobi("Kenobi!");

return;

}

void kenobi(string name){

cin << "General " << name << endl;

return;

}

g)

#include <iostream>

double surprise(int input){

string mysteryString = "surprise";

input == input % 8;

return mysteryString[input];

}

int main(){

int input;

char result;

cout << "Enter a number:" << endl;

cin > input;

result = surprise(input);

cout << result << endl;

return 0;

}