Week 9

Intro to Computer Science - CSI 1430

Hello and welcome to the weekly resources for CSI 1430: Introduction to Computer Science! This week's resource will be covering the course material that will be covered in week 9 of the course.

A majority of the “base material” for this course has been covered, which means that we slow down conceptually to add complexity. Make sure you go over the concepts learned in the first half of the semester so that when we put a bunch of different ideas on top of one another, we are able to do so with ease. **Just as a reminder:** These documents are intended for going over concepts that are discovered in class, as they are a concise and to the point review of the materials. But remember, these documents are not a replacement for lecture and they cannot cover everything, so make sure you get help on any specific concepts you struggle with as well as go over lecture notes, programs, and the textbook.

Reminder: If you have any questions about these study guides, group tutoring sessions, private 30 minutes tutoring appointments, the Baylor Tutoring Youtube channel or any tutoring services we offer, please visit our website https://baylor.edu/tutoring or call our drop in center during open business hours. Monday-Thursday 9am - 8pm on class days (254) 710-4135.

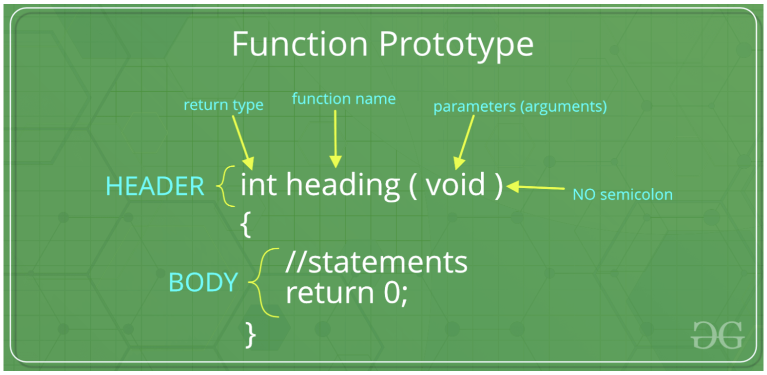
Keywords: *Functions*

TOPIC OF THE WEEK

Writing Functions

Functions are a way we can assign a name to a chunk of code bundled together to achieve a certain outcome. We use functions in coding, when we do not want to keep repeating the same blocks of code in a program over, and over, and over again. Throughout the course so far, you have been using many different types of STL functions (Ex. sqrt(), setprecision(), etc), but now let's learn how to define some of our own!

To start our discussion of user-defined functions, let's take a look at the prototype of what a function consists of:



```
int heading ( void )
{
    //statements
    return 0;
}
```

There are two major parts of defining a function.

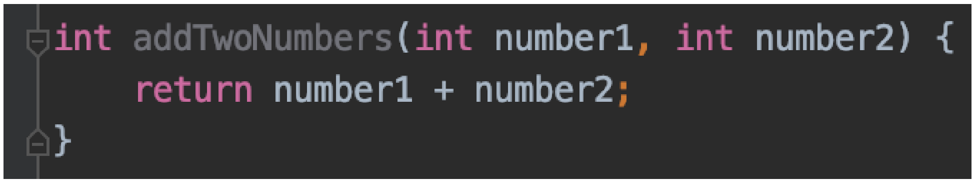
1. Function Header
2. Function Body

The function header essentially is going to define what type of data is going to be needed for the function to operate, as well as how we are going to call the function by name. The first part of the function header is the **return type**. The return type is going to define what is going to be produced from a function. For example, the sqrt() function finds the square root of a number and returns a double value which is the intended value to be produced from the function. This return type can be anything from numerical types to strings, arrays, etc. However, if a function does not need to return a value, then we can specify void. Second, in the function header, comes the **function name**. This is essentially how we will be calling the function and how one specific function is distinguished from another function with another name. These functions should be conventionally named for what their intended purpose is. For example, we want a function that adds two numbers, the function name should be along the lines of addTwoNumbers(). Of course, we can name a function whatever we would like, but similar to naming variables appropriately, we want to also name functions with meaningful names. Lastly, the final thing within a function header is the **parameters**. Parameters are variables that are passed to a function in order for the function to achieve their intended purpose. A function does not necessarily have to have parameters, however in most cases when we are calculating or performing some manipulation, then they will have to be included. For our addTwoNumbers() example we want to pass in two different integer variables so that they can be added.

The final function header for the addTwoNumbers() function would resemble something similar to the following:

int addTwoNumbers(int number1, int number2);

The body of the function is responsible for defining the behavior of what the function will be responsible for doing. More or less, we have been doing this all along. However, now we are breaking it up into chunks of purpose-intended code. For our add two numbers example the total body function would look similar to the following.



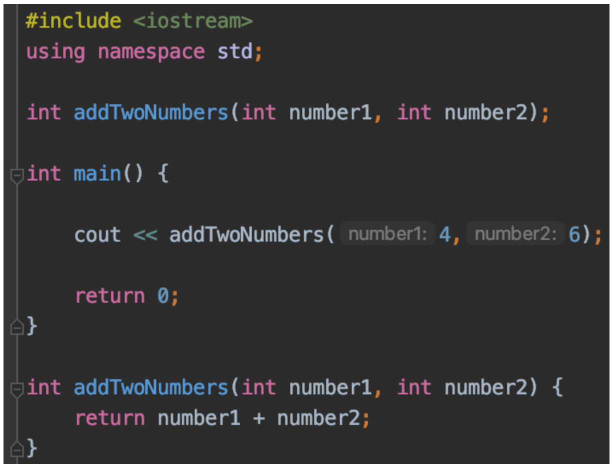
```
int addTwoNumbers(int number1, int number2) {
    return number1 + number2;
}
```

Highlights

Using Functions

Now let's talk about structure...In terms of where functions must be placed, any functions that we want to call within the main(), MUST be declared before the main(). There are 3 main ways that this can be achieved that do virtually the same thing:

1. The entire function must be declared before the main().
2. The function header must be declared before the main, and the body can be declared after.
3. The header and body can be declared in separate files (.h and .cpp respectively), and we can #include the header file (.h) within the file that contains the main().



```
#include <iostream>
using namespace std;

int addTwoNumbers(int number1, int number2);

int main() {

    cout << addTwoNumbers( number1: 4, number2: 6);

    return 0;
}

int addTwoNumbers(int number1, int number2) {
    return number1 + number2;
}
```

When specifically talking about functions (AND NOT classes),...then I prefer to do everything in the same file. Therefore, above I have presented an implementation of the addTwoNumbers() function that allows us to call an example add (of the numbers 4 and 6). However, there is merit to doing it in separate .h and .cpp files which we will explore later.

CHECK YOUR LEARNING

1. **What are the three main parts of a function header?**
2. **Where must a function or function header be declared in order for the main() to work as intended?**
3. **What should a function's return type be, if it does not return a value?**

THINGS STUDENTS MAY STRUGGLE WITH

1. There are all sorts of types of functions. Throughout the rest of the semester we will explore all of these types and see what the purpose is for each one.
    - Function with no return value and no arguments
    - Function with a return value and no arguments
    - Function with no return value and arguments
    - Function with a return value and arguments
2. If you choose (or are required) to use a .h and .cpp file for breaking up the function header and body into multiple files, then make sure that you #include “<filename>.h”.

This will make sure that the content of your user-defined file’s code is being noticed by the preprocessor before compilation.

Thanks for checking out these weekly resources! Don’t forget to check out our website for group tutoring times, video tutorials and lots of other course resources: https://baylor.edu/tutoring. The answers to the ‘Check Your Learning’ questions are below.

Answers:

1. Return type, name, and parameters
2. Before the main() in the file
3. Void

Note: All tables were taken from the Baylor CS 1430 zyBook