Procedural Abstraction and Functions that Return a Value II

CS 211

Adam Sweeney

September 16, 2017

Wichita State University, EECS

Procedural Abstraction and Functions that Return a Value II
CS 211
Adam Sweeney
September 16, 2017
Wichita State University, EECS

Introduction

- Procedural Abstraction sounds scarier than it really is

Introduction

- Procedural Abstraction sounds scarier than it really is

Agenda

- Procedural Abstraction
- Scope and Local Variables
- Overloading Function Names

Agenda

- Procedural Abstraction
- Scope and Local Variables
- Overloading Function Names

Procedural Abstraction

A Black Box

- What do we mean by black box in the context of programming?
- Functions that we use are considered black boxes
- This is procedural abstraction

A Black Box
- What do we mean by black box in the context of programming?
- Functions that we use are considered black boxes
- This is procedural abstraction

1. A: A black box takes its inputs and gives you output, without you knowing a single thing about how it happened
2. We know what the function `sqrt()` does, and we don't care how it does it

How Should We Blackbox Our Functions?

- Simply saying that a function can be a black box isn't enough
- Functions need to be designed to behave this way
- Two major requirements
  - Function declaration comment that explains function
  - All variables used in the function body should be declared in the function body behavior

How Should We Blackbox Our Functions?
- Simply saying that a function can be a black box isn't enough
- Functions need to be designed to behave this way
- Two major requirements
  - Function declaration comment that explains function
  - All variables used in the function body should be declared in the function body behavior

1. Requirements for arguments, if any. I.e., argument X is user input
2. Choosing smart names for our function and arguments lets us get away with smaller comments
3. This mostly speaks to not relying on global variables
4. Remember, arguments **are** local to the function body

Recognize Opportunities for Abstraction

- Nested Loops are great for abstraction
- See Demo (`https://repl.it/LDtx/10`)
- Is there a repetitive task that has to be done many times?
  - For example, trimming whitespace from the ends of user input
  - Doing a calculation that isn't in a library
  - Printing rows of a table (wink wink)

Recognize Opportunities for Abstraction
- Nested Loops are great for abstraction
- See Demo (https://repl.it/LDtx/10)
- Is there a repetitive task that has to be done many times?
  - For example, trimming whitespace from the ends of user input
  - Doing a calculation that isn't in a library
  - Printing rows of a table (wink wink)

Let Us Consider One of the Most Important Problems

- We want the best bang for our buck
- We buy pizza
- The largest pizza is not always the most economical pizza
- Pizza economy is determined by figuring out the price per square inch of pizza

Let Us Consider One of the Most Important Problems
- We want the best bang for our buck
- We buy pizza
- The largest pizza is not always the most economical pizza
- Pizza economy is determined by figuring out the price per square inch of pizza

The First Step of Our Plan

- We want to find the cheapest pizza based on minimum price per square inch
- Inputs
  - Information about 2 pizzas
  - Diameter
  - Price
- Output
  - Cost per square inch of each pizza
  - Statement of which is most economical
  - Tie goes to smaller pizza

The First Step of Our Plan
- We want to find the cheapest pizza based on minimum price per square inch
- Inputs
  - Information about 2 pizzas
  - Diameter
  - Price
- Output
  - Cost per square inch of each pizza
  - Statement of which is most economical
  - Tie goes to smaller pizza

Still Planning

- Break the problem down into sub-tasks
  1. Get input data for both pizzas
  2. Compute the price per square inch of small pizza
  3. Compute the price per square inch of large pizza
  4. Determine which is the better buy
  5. Output the results
- Let's step back for a minute and look at these tasks
- Tasks 2 and 3 look very similar to each other

Still Planning
- Break the problem down into sub-tasks
  1. Get input data for both pizzas
  2. Compute the price per square inch of small pizza
  3. Compute the price per square inch of large pizza
  4. Determine which is the better buy
  5. Output the results
- Let's step back for a minute and look at these tasks
- Tasks 2 and 3 look very similar to each other

1. Once we've restated a problem, going over our restatement can help us pick up on things
2. They both require finding the price per square inch of a pizza
3. I would need the same information to calculate the price of any pizza
4. Diameter and Price. I am getting the same result, or ouput, price / square inch
5. These seem like a prime candidate for a function
6. By taking the time to restate the problem and look at it again, I've just found that I can solve two tasks at once with a function

Signs of a Function

- Takes input
- Does stuff with it
- Returns a single result

Signs of a Function
- Takes input
- Does stuff with it
- Returns a single result

1. This makes it look like functions should be easy to spot
2. They can be, with practice

Algorithm Design

- Let's tackle the stuff we know we can do first
- Task 1, Get input data for both pizzas
  - We've done this a few times now, we know what to do
  - Use `cout` and `cin`
  - We will need some variables to store this information
    - Variables for size, and price, for each pizza
    - We'll choose `smallSize`, `smallPrice`, `largeSize`, and `largePrice`
    - What should their data types be?
- Task 4, Determine which is the better buy
  - Also a simple task; we just need to compare the two numbers
  - An if-else block should handle this

Algorithm Design
- Let's tackle the stuff we know we can do first
- Task 1, Get input data for both pizzas
  - We've done this a few times now, we know what to do
  - Use cout and cin
  - We will need some variables to store this information
    - Variables for size, and price, for each pizza
    - We'll choose smallSize, smallPrice, largeSize, and largePrice
    - What should their data types be?
- Task 4, Determine which is the better buy
  - Also a simple task; we just need to compare the two numbers
  - An if-else block should handle this

1. Data types for the variables: `int` for the sizes, `double` for the prices

Algorithm Design (cont.)

- Task 5, Output the result
  - We know how to do this, just a cout statement or two
  - Very closely related to Task 4
- Tasks 2 and 3, Calculate the price per square inch of the pizzas
  - This is toughest part of our problem
  - But, we're going to use the momentum we've gained from planning everything else and realizing how simple those parts are to help us knock out this function
  - Let's start with some pseudocode for the function body

Algorithm Design (cont.)
- Task 5, Output the result
  - We know how to do this, just a cout statement or two
  - Very closely related to Task 4
- Tasks 2 and 3, Calculate the price per square inch of the pizzas
  - This is toughest part of our problem
  - But, we're going to use the momentum we've gained from planning everything else and realizing how simple those parts are to help us knock out this function
  - Let's start with some pseudocode for the function body

Function Pseudocode

- What does our function have to do?
- From Tasks 2 and 3, Calculate the price per square inch of a pizza
- Price per square inch is `price / area`
- We will know the price, it's one of the variables we decided we need
- What is the area of a circle?
  - πr^2
  - We can get the radius from the diameter, which is another variable we already decided we needed
  - $\pi(d/2) * (d/2)$

Function Pseudocode
- What does our function have to do?
- From Tasks 2 and 3, Calculate the price per square inch of a pizza
- Price per square inch is price / area
- We will know the price, it's one of the variables we decided we need
- What is the area of a circle?
  - πr^2
  - We can get the radius from the diameter, which is another variable we already decided we needed
  - $\pi(d/2) * (d/2)$

Function Pseudocode (cont.)

```
Function declaration
Calculate Area: p * r^2
return price / area
```

We've figured out the body, let's figure out the prototype/declaration

Function Pseudocode (cont.)
Function declaration
Calculate Area: p * r^2
return price / area
We've figured out the body, let's figure out the prototype/declaration

Function Pseudocode (cont.)

- What is the function giving back?
- A dollar amount of a square inch of pizza
- Sounds like our value returned should be a `double`
- Our name should have an action word, and describe what we're doing
- For now, we'll go with `calculatePizzaPricePerArea`
- It's a long name, but IDE's will auto-fill that for us as we advance through programming
- Best to get good practices established

Function Pseudocode (cont.)
- What is the function giving back?
- A dollar amount of a square inch of pizza
- Sounds like our value returned should be a double
- Our name should have an action word, and describe what we're doing
- For now, we'll go with calculatePizzaPricePerArea
- It's a long name, but IDE's will auto-fill that for us as we advance through programming
- Best to get good practices established

Function Pseudocode (cont.)

```
double calculatePizzaPricePerArea()
Calculate Area: p * r^2
return price / area
```

We're starting to mix some actual C++ in our pseudocode, but it's important that the function be properly defined

Function Pseudocode (cont.)
double calculatePizzaPricePerArea()
Calculate Area: p * r^2
return price / area
We're starting to mix some actual C++ in our pseudocode, but it's important that the function be properly defined

Our Function Arguments

- What information does our function need?
- What are the data types of the information?

Our Function Arguments
- What information does our function need?
- What are the data types of the information?

1. A: What information do we have, to give?
2. A price, and a diameter (size)
3. `int` for the size, `double` for the price

The full function Prototype

```
double calculatePizzaPricePerArea(int size, double price);
```

- This is the complete declaration/prototype for our function
- Using the pseudocode, we can create the whole function

The full function Prototype
double calculatePizzaPricePerArea(int size, double price);
- This is the complete declaration/prototype for our function
- Using the pseudocode, we can create the whole function

Some of our actual code

```
#include <iostream>
using namespace std;
double calculatePizzaPricePerArea(int size, double price);
int main()
{
    int smallsize, largeSize, smallEconomy;
    double smallPrice, largePrice, largeEconomy;
    // Get input
    smallEconomy = calculatePizzaPricePerArea(smallSize, smallPrice);
    largeEconomy = calculatePizzaPricePerArea(largeSize, largePrice);
    // Compare prices per square inch
    // State outcome
    return 0;
}

double calculatePizzaPricePerArea(int size, double price)
{
    const double PI = 3.14159;
    double area = PI * (size / 2) * (size / 2)
    return = price / area;
}
```


1. Only included parts relating to the function due to space
2. Note that the function is called twice, once for each pizza
3. It is not called once to calculate two pizzas, that is too specific to our problem, and poor design

Scope and Local Variables

Scope and Local Variables

- Let's look at our function again

```
double calculatePizzaPricePerArea(int size, double price)
{
    const double PI = 3.14159;
    double area = PI * (size / 2) * (size / 2)
    return = price / area;
}
```

- Let us consider the constant, PI

Local Variables

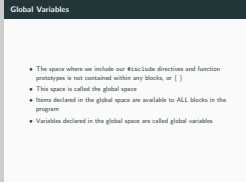
- π is a universal constant
- Where in our program is it used?
- Because it is used exclusively in our function, it makes sense to declare it within our function
- Doing so limits its scope so that it only exists where it is needed
- Variables declared inside a function are called local variables
- What if a variable is needed across many functions?

Local Variables
- π is a universal constant
- Where in our program is it used?
- Because it is used exclusively in our function, it makes sense to declare it within our function
- Doing so limits its scope so that it only exists where it is needed
- Variables declared inside a function are called local variables
- What if a variable is needed across many functions?

1. π is used exclusively in our function

Global Variables

- The space where we include our `#include` directives and function prototypes is not contained within any blocks, or { }
- This space is called the global space
- Items declared in the global space are available to ALL blocks in the program
- Variables declared in the global space are called global variables



Well, Then

- Wouldn't it seem so much easier to just declare everything in the global space?
  - Don't have to worry about scope at all
  - Everything has access to everything
- The reality is not so simple

Well, Then
- Wouldn't it seem so much easier to just declare everything in the global space?
  - Don't have to worry about scope at all
  - Everything has access to everything
- The reality is not so simple

The size of programs

- We are an intro course
- Meaningful programs that we write later on in college and more so in industry, will be so MUCH larger
- There will simply be too many variables to track, keeping every variable in RAM for the duration of the program execution is wasteful
- We will rely on scope to maintain our sanity

The size of programs

- We are an intro course
- Meaningful programs that we write later on in college and more so in industry, will be so MUCH larger
- There will simply be too many variables to track, keeping every variable in RAM for the duration of the program execution is wasteful
- We will rely on scope to maintain our sanity

So Why Are We Bothering Bringing This Up?

- There are circumstances where global variables make perfect sense
- For example of the constant variable PI, what if we needed to use PI in more than one function?
- In this scenario, a global declaration makes sense

So Why Are We Bothering Bringing This Up?
- There are circumstances where global variables make perfect sense
- For example of the constant variable PI, what if we needed to use PI in more than one function?
- In this scenario, a global declaration makes sense

Some Advice on Variable Scope

- A prime candidate for a global variable is a *constant* that will appear in more than one function
  - Function arguments do not apply due to how 'pass by value' works
- Try to make your variables as local as possible
  - Variables used only by one function should be declared in that function
  - The same applies to loops
  - Don't go out of your way. It should seem natural where a variable goes

Some Advice on Variable Scope
- A prime candidate for a global variable is a constant that will appear in more than one function
  - Function arguments do not apply due to how 'pass by value' works
- Try to make your variables as local as possible
  - Variables used only by one function should be declared in that function
  - The same applies to loops
  - Don't go out of your way. It should seem natural where a variable goes

Function Overloading

Function Overloading

- How would we write a function that returns the maximum value of two integers?
- Maybe something like this:

```
int findMax(int first, int second)
{
    if (first > second)
        return first;
    else
        return second;
}
```

- What if we also needed to compare doubles in the same program?
- Giving each function a different name makes for unnecessary clutter in our program
- It is possible to have multiple functions share the same name

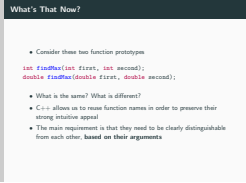
Function Overloading
- How would we write a function that returns the maximum value of two integers?
- Maybe something like this:
- What if we also needed to compare doubles in the same program?
- Giving each function a different name makes for unnecessary clutter in our program
- It is possible to have multiple functions share the same name

What's That Now?

- Consider these two function prototypes

```
int findMax(int first, int second);
double findMax(double first, double second);
```

- What is the same? What is different?
- C++ allows us to reuse function names in order to preserve their strong intuitive appeal
- The main requirement is that they need to be clearly distinguishable from each other, **based on their arguments**



1. Names of things are the same, data types of everything are different

How to Overload Functions

```
int findMax(int first, int second);
double findMax(double first, double second);
```

- Compare the parameter lists
- The number of parameters are the same, but the data types are different
- This gives the compiler a clear distinction between the two functions, and always allows the proper one to be called
- If the types are the same, the number of arguments must differ
- **WARNING**: Changing only the return type is NOT enough

How to Overload Functions

int findMax(int first, int second);
double findMax(double first, double second);

- Compare the parameter lists
- The number of parameters are the same, but the data types are different
- This gives the compiler a clear distinction between the two functions, and always allows the proper one to be called
- If the types are the same, the number of arguments must differ
- **WARNING**: Changing only the return type is NOT enough

Re-visiting the Pizza Problem

- What changes should we make to include rectangular pizzas
- Try it out

Re-visiting the Pizza Problem
- What changes should we make to include rectangular pizzas
- Try it out