

PROCEDURAL ABSTRACTION AND FUNCTIONS THAT RETURN A VALUE II

CS 211


WICHITA STATE UNIVERSITY

INTRODUCTION

- Procedural Abstraction sounds scarier than it is

AGENDA

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



PROCEDURAL ABSTRACTION

A BLACK BOX

- What do we usually think of when we hear the term black box
 - Not quite the same in the context of code
- Functions (and later classes) we use are considered black boxes
 - We know what they do, but not how
- This idea is the definition of procedural abstraction
 - We abstract the procedure

SO WHY ARE WE HAVING A LECTURE?

- Simply saying that functions are black boxes isn't enough
- We need to design functions to be easy-to-use & portable black boxes
- Two major requirements
 - Comment above function declaration
 - All variables used in the function body should be declared in the function body, i.e., don't rely on global variables

COMMENTS

- They provide a brief description of the function, the parameters, and the value returned
- A much more compact version of what we find at cppreference.com
- The extra information can fill in the gaps that even well named functions and parameters can't fill
- This is a good habit
 - Doxygen is a tool that takes specially formatted comments and generates documentation

A decorative wavy line in yellow and white on the left side of the image.

SCOPE AND LOCAL VARIABLES

LET US SUPPOSE

- We want a program to compare two pizzas and give us the “better” deal
 - Price per square inch of pizza
- The area of a circle is pi times the square of the radius
 - Where should I declare pi?
 - What are my options?

PLACING PI

```
// Outside the function, in the global scope
```

```
double calculate_area(double diameter) {  
    // In the function  
}
```

- Either can work
- We need to decide what makes the most sense

GLOBAL PI

- Pros
 - Accessible by everything
- Cons
 - Function is not self-contained
 - Parameter list and function body are not enough to to its job
 - Global declarations can get very messy very quickly

LOCAL PI

- Pros
 - Function is fully self-contained
- Cons
 - Not available outside of function; would require re-declaration if needed elsewhere

NAMESPACE PI

- The real alternative to global declarations
- Not covered in this course

AN AREA FUNCTION

```
double calculate_area(double diameter)
{
    const double PI = 3.14159;
    return PI * (diameter / 2.0) * (diameter / 2.0);
}
```

- Generally, try to make all variables as local as possible



OVERLOADING FUNCTION NAMES

CONSIDER THE FOLLOWING

```
int find_max(int a, int b)
{
    return a > b ? a : b;
}
```

- A simple, well-named function
- What if I need to the maximum of 3 numbers, or of doubles?
 - What would those functions be called?

OVERLOADING

- C++ allows multiple functions to have the same name, so long as they are all unique from each other
- Compiler examines a function's signature to make this determination
 - Function name, and parameter list (types, quantity, order)
 - Return type is NOT part of the signature
 - Since the names will be the same, the parameter list must be different

OVERLOADING FIND_MAX()

```
int find_max(int a,  
             int b)  
{  
    return a > b ? a : b;  
}
```

```
double find_max(double a,  
                double b)  
{  
    return a > b ? a : b;  
}
```