In-class Worksheet #7

Fall 2022 , Nov 4, 2022

Computer Science I & Lab

0. Midterm review on expressions

```
    int x=10;

    cout << (3<x<5)<<endl;
    cout << (3<x<0) <<endl;
```

```
< and < are of the same precedence (same operator), the Associativity
is used to decide which operator is applied first.

For <, it's left-to-right.

(see C++ Operator Precedence:
https://en.cppreference.com/w/cpp/language/operator_precedence)
```

Practice:

- Why we don't need to use () below?

```
    (month<1 || month>12 || day > 31 || day <1 || year <0)
```

- What's evaluated first, next, and so on?

```
   (year > year2 || year==year2 && month < month2 ||
          year==year2 && month==month2 && day<day2)
```

- What will be the value of b after the following statements?

1. Code tracing (a way to make sense of the code).

```
int a=0;
int b=20;

if (a==10)
if (b==10)
cout << "*******";
else
cout <<"+++++++";
```

   First step: fix the indentation in order to understand the if/else and if statement, i.e., what does each of the cout statements belong to?

2. Calling a pre-defined function

   Syntax:

```
  #include <appropriate_header_file>

  function_name (argument list)
```

   - Can be used as a statement.
   - If the function "returns" a value, then the above function call can be used in an expression, or cout, if and while's condition so on and on.

   e.g.,

```
#include <cmath> // a header file that describes multiple math functions,
                  // e.g., it includes a line such as
                 // double floor(double x);
                 //you can type "main floor" to get a description
cout << "floor of 100.93 is "<< floor (100.93) <<endl;
```

3. Study the code to understand the whole program's structure, function declaration, function definition and function call.

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

//Returns the area of a circle with the given radius
//The formal parameter named price is the radius of the circle.
// The returned value is the area of the circle
double circle_area (double radius);

/* 1. What does the above statement mean?





 */

//2. What happens if we just have, i.e., there is no ( )…
//  double circle_area;



//2. How do you tell compiler that we will have a function that calculate the
circumference of a circle? It's taking a double type value as input, and
output a double type value as a return value.



/* Note: Similar to variable name, function names must follow the follow
rules:
```
Names can contain letters, digits and underscores

- Names must begin with a letter or an underscore (_)
- Names are case sensitive (myVar and myvar are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (like C++ keywords, such as int) cannot be used as names *

```
int main( ) // Why main is a very special function?
// what's the body of main?
{
    int diameter_small, diameter_large;
    double price_small, unitprice_small,
           price_large, unitprice_large;

    cout << "Enter diameter of a small pizza (in inches): ";
    cin >> diameter_small;
    cout << "Enter the price of a small pizza: $";
    cin >> price_small;
    cout << "Enter diameter of a large pizza (in inches): ";
    cin >> diameter_large;
    cout << "Enter the price of a large pizza: $";
    cin >> price_large;
    //the price we pay for each unit area for small pizza is calculated
    // by dividing the small pizza's price by its area

    unitprice_small = price_small/circle_area(diameter_small/2.0);
    //How to make sense of this line?




    unitprice_large = price_large/circle_area(diameter_large/2.0);

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "Small pizza:\n"
         << "Diameter = " << diameter_small << " inches\n"
         << "Price = $" << price_small
         << " Per square inch = $" << unitprice_small << endl
         << "Large pizza:\n"
         << "Diameter = " << diameter_large << " inches\n"
         << "Price = $" << price_large
         << " Per square inch = $" << unitprice_large << endl;

    if (unitprice_large < unitprice_small)
        cout << "The large one is the better buy.\n";
    else
        cout << "The small one is the better buy.\n";

    cout << "Buon Appetito!\n";

    return 0;
}

/* The following part of code define the function:
 Todo: label the function header, function body */

double circle_area(double radius)
{
    const double PI = 3.14159;
    double area;

    area = PI * radius * radius;
    return (area);
}
```

4. Details of function call

- What's a function call?
- Arguments are evaluated, and plugged in for the "formal parameter", i.e., the formal parameter (which is a variable itself) is assigned a value
- Body of the function is executed

- Until it reaches a “return” statement or reaches the end of the function body

  The function call is replaced by the value the function returns.

- The main resumes execution…

5. Inside a function’s body

   It’s a like a small program itself. The input are the parameters, output are the value returned.

   You can have any statements in the function body, as needed to implement the “functionalities”.

   Blackbox analogy:

6. Practice:

   - Can we declare a function that for calculating 2^n for a given positive integer n?
   - Can you provide the definition of the function? (i.e., implement it) ?

7. A function is like a small “program”.

	Program	Function
Well-defined functionalities		
Take some input		
Generate some output		

Implementation: how we deliver the functionalities		