In-class Worksheet #8
Fall 2022 ， Nov 4, 2022
Computer Science I & Lab

1. Quick review of precedence rule and associativity

- logic error or syntax error?
  int a=10;
  if (a=2)
  cout <<"a is 2\n";

  int b;
  b=a=20+10;

- Prefix and Postfix increment (++) and Decrement (--)

  //what's the output?

```
int a=10;
int b=10;

  if (++a==11 && b++==11)
          cout <<"Here\n";

  cout <<"a="<<a<<endl;
  cout <<"b="<<b<<endl;

  cout <<a++<<endl;
  cout <<++b<<endl;

  if (a==11 || b++==12)
          cout <<"THere\n";
  cout <<b<<endl;
```

- If you use the `++` operator as a prefix like: `++var`, the value of `var` is incremented by 1; then it returns the value.
- If you use the `++` operator as a postfix like: `var++`, the original value of `var` is returned first; then `var` is incremented by 1.

2. Common pitfalls regarding function calls
   a. Call the function without using the return value

b. When calling a function with multiple parameters, passing arguments in wrong order.

   e.g., to calculate 3^10, should call pow (3,10), not pow(10,3)

   when not sure, check manual (man pow).

c. Make the same function call multiple times

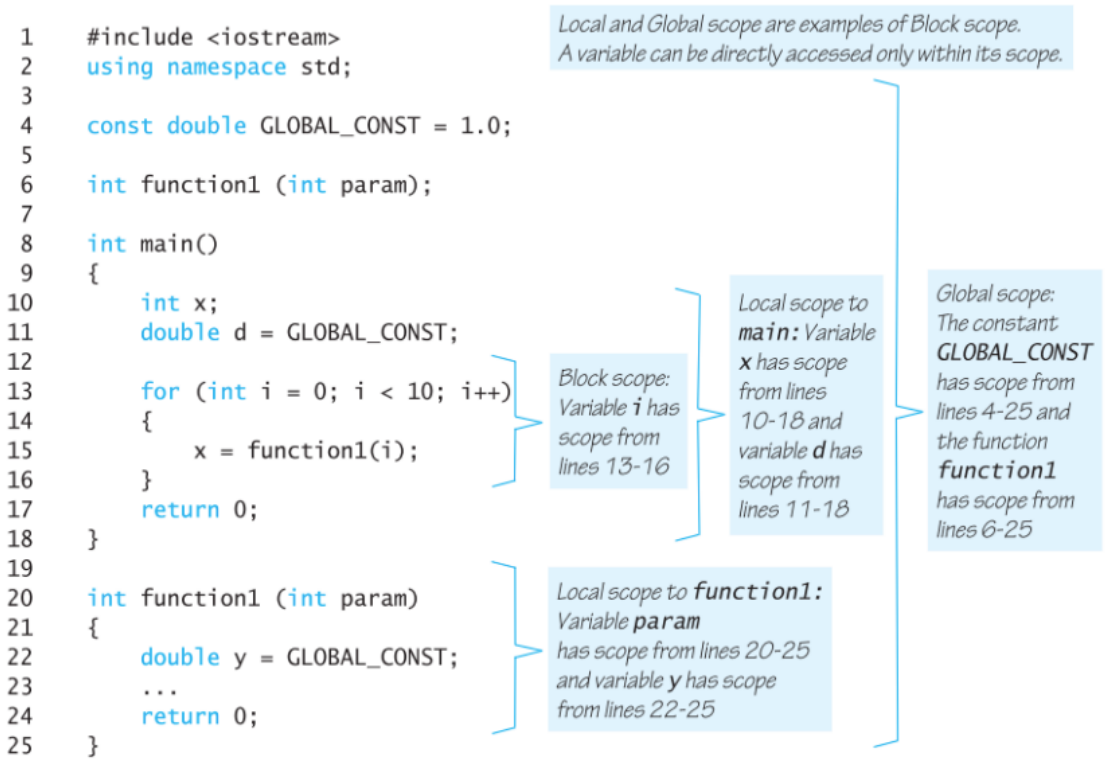
   In lab5, calling ToMilitaryHour( ) again and again to convert the hour given in am-pm format to military format …

   Instead, we shall save the conversion result in a variable, and use the variable subsequently when needing the hour in miliary format:

   hr1 = ToMiliaryHour (hr1, ampm);

3. Scope: Any identifier used in a C++ program (such as the name of a variable or object, the name of a type or class, or the name of a named constant) has a *scope*, i.e., a region of the program in which that name can be used. Often we are mostly concerned with the scope of variables.

   A variable's scope is decided by where it's defined/declared.




```
1   #include <iostream>
2   using namespace std;
3
4   const double GLOBAL_CONST = 1.0;
5
6   int function1 (int param);
7
8   int main()
9   {
10      int x;
11      double d = GLOBAL_CONST;
12
13      for (int i = 0; i < 10; i++)
14      {
15          x = function1(i);
16      }
17      return 0;
18  }
19
20  int function1 (int param)
21  {
22      double y = GLOBAL_CONST;
23      ...
24      return 0;
25  }
```

Some points of discussions:

- Can we access variable x in function1? Can we access variable y in main?

- Can we have two local variables in function1 with same name?

- Can we define another variable named x in function1?

- Can we define another variable named x inside the for loop's body, line 15?

- Can we declare a local variable named param inside function1?

- Suppose we want to modify x, the local variable in main within function1?

4. Function formal parameters are actually local variables of the function, and they are initialized (i.e., assigned values) at the function call.

   If argument's type does not match with formal parameter's type, it will be converted (if possible) to the parameter's type; if it's impossible, a syntax error will be reported:

```
void test(int a)
{
        cout <<"a="<<a<<endl;
}

int main()
```

```
{
  int a=1;

     test(3.2);
     test('A');
     test(3<a<20);
     test("Hello");
}
```

5. Some functions do not return value, instead they output the results to terminal

   Denote the return type as "void" to indicate that there is no return value.

   e.g., write a function that prints a divider line making up of 30 * to the terminal. How to make it so that you can tell it to print using different character, with different length?

6. Function without parameters: some functions do not take input from parameters, instead they read input from keyboard, or have other mechanism for input…

   e.g., rand( ) function generates and returns a random number:

   //to roll a virtual dice, i.e., get a random number between 1, 2, …6
   int value =

0. 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;

1. 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);
}
```

2. 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 main resumes execution at the point where the function call is made. The function call is replaced by the value the function returns.

3. Inside a function's body

    It's a like a small program itself.

    - Most of the time, input to functions are the parameters, output of functions are the value returned.
    - Sometimes, the input to a function can be from keyboard.
    - Sometimes, the output of a function can be to the terminal.

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

    Why a function is like a blackbox:

    - Interface:

    - Information hiding

- Local variables

4. 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) ?

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

	Program	Function
Well-defined functionalities		
Take some input		
Generate some output		
Implementation: how we deliver the functionalities		