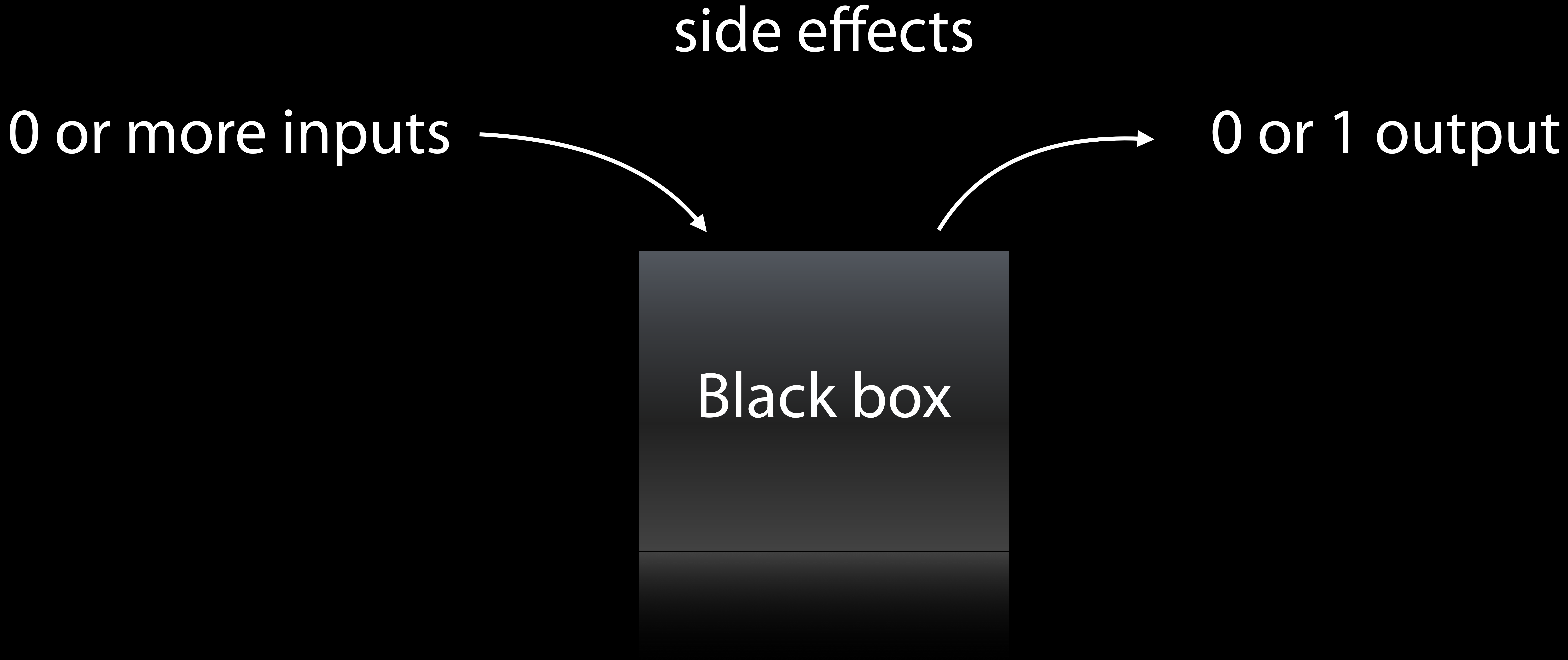
Week 4

EECS 183

Functions



RME

```
/**
 * Requires: n >= 0.
 * Modifies: Nothing.
 * Effects:  Returns square root of n.
 */
double sqrt(double n);
```

Defining a function

name parameter / argument list

return type

body

```
int add(int a, int b)
{
    int sum = a + b;
    return sum;
}
```

Using a function

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

int add(int a, int b);

int main()
{
    int sum = add(1, 2);
    cout << sum << endl;
}

int add(int a, int b)
{
    int sum = a + b;
    return sum;
}
```

Using a function

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

int add(int a, int b);

int main()
{
    int sum = add(1, 2);
    cout << sum << endl;
}

int add(int a, int b)
{
    int sum = a + b;
    return sum;
}
```

prototype / declaration

function call

implementation / definition

Scope

Local variables (defined inside functions) are known within functions wherein they were defined.

Local variables are passed by value to other functions.

Other functions receive a copy of the value, not the variable itself.

The variable in the original function is unchanged unless overwritten.

Scope

`increment-0.cpp`

`increment-1.cpp`

`increment-2.cpp`

scope.cpp

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

// prototypes
void foo();
void bar();

// global variable
int x = 5;

int main()
{
    cout << x << endl;
    int x = 7;
    cout << x << endl;
    foo();
    cout << x << endl;
    bar();
}
```

```
void foo()
{
    x = 10;
    int x = 42;
    cout << x << endl;
}

void bar()
{
    cout << x << endl;
}
```

Swap

$$\begin{tikzcd}
2 & a \arrow[r, bend left] & b \arrow[l, bend left] & 5
\end{tikzcd}$$

Swap

```
int temp = a;
a = b;
b = temp;
```

Swap

```
void swap(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

S	M	Tu	W	Th	F
27	28	29	30	1	2
4	5	6	7	8	9
11	12	13	14	15	16

Comparison Operators

x

<

<=

>

>=

==

!=

y

Conditions

```
if (condition)
{
    // do this
}
```

Conditions

```
if (condition)
{
    // do this
}
else
{
    // do that
}
```

Conditions

```
if (course == 183)
{
    cout << "EECS 183 is Elementary Programming Concepts"
         << endl;
}
else
{
    cout << "I'm not familiar with that class!" << endl;
}
```

Conditions

```
string pluralize(string singular, string plural,
                 int number)
{
    if (number == 1)
    {
        return singular;
    }
    return plural;
}
```

Conditions

```
if (condition)
{
    // do this
}
else if (condition)
{
    // do that
}
else
{
    // do this other thing
}
```

Conditions

```
if (course == 183)
{
    cout << "EECS 183 is Elementary Programming Concepts"
         << endl;
}
else if (course == 203)
{
    cout << "EECS 203 is Discrete Mathematics" << endl;
}
else
{
    cout << "I'm not familiar with that class!" << endl;
}
```

Conditions

```
if (x < y)
{
    cout << "x is less than y" << endl;
}
else if (x > y)
{
    cout << "x is greater than y" << endl;
}
else
{
    cout << "x is equal to y" << endl;
}
```

Conditions

```
conditions-0.cpp

conditions-1.cpp
```

True or false?

Every `if` statement needs an `else` statement.

Conditions

```
if (condition)
{
    // do something
}
else
{
    // do nothing
}
```

Conditions

```
if (condition)
{
    // do something
}
else
{

}
```

True or false?

Every `else` statement must go with an `if` statement.

True or false?

Order of `if` / `else if` / `else` statements matters.

Logical Operators

&&

||

!

Boolean Expressions

```
if (condition && condition)
{
    // do this
}
```

Boolean Expressions

```
if (condition || condition)
{
    // do this
}
```

Boolean Expressions

```
if (!condition)
{
    // do this
}
```

Conditions

```
if (condition)
{
    // do nothing
}
else
{
    // do something
}
```

Conditions

```
if (condition)
{
    // do nothing
}
else
{
    // do something
}
```

Conditions

```
if (!condition)
{
    // do something
}
```

Operator Precedence

Level	Operator
2	`++ -- () [] .`
3	`++ -- ! + - &`
5	`* / %`
6	`+ -`
7	`<< >>`
8	`< > <= >=`
9	`== !=`
13	`&&`
14	`||`
15	`= ×= ÷= %= += -=`

Boolean Expressions

Write a boolean expression in C++ that means
“a < b < c.”

Hint: this boolean expressions checks if *a* is equal to *b*:

```
a  ==  b
```

Boolean Expressions

Write a boolean expression in C++ that means
“$0 \leq x < 10$ or x is less than y.”

Hint: this boolean expressions checks if *a* is equal to *b*:

```
a == b
```

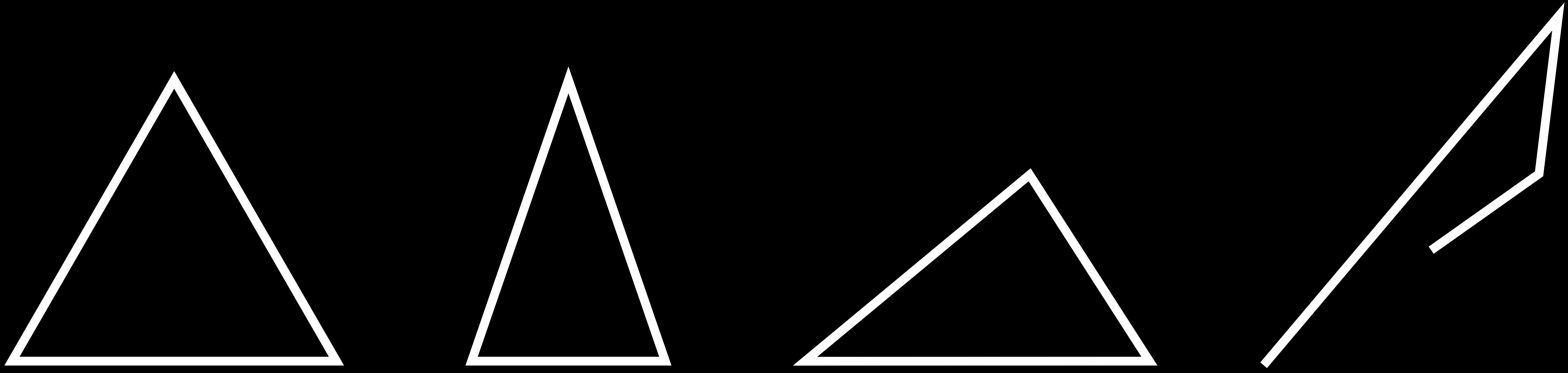
Conditions

Ask the user for an integer between 1 and 9

Say if that number is small (1, 2, 3), medium (4, 5, 6), large (7, 8, 9) or invalid.

`triangles.cpp`

Write a program that asks the user for side lengths of a triangle and then prints the type of the triangle (equilateral, isosceles, scalene, invalid).


equilateral
isosceles
scalene
invalid