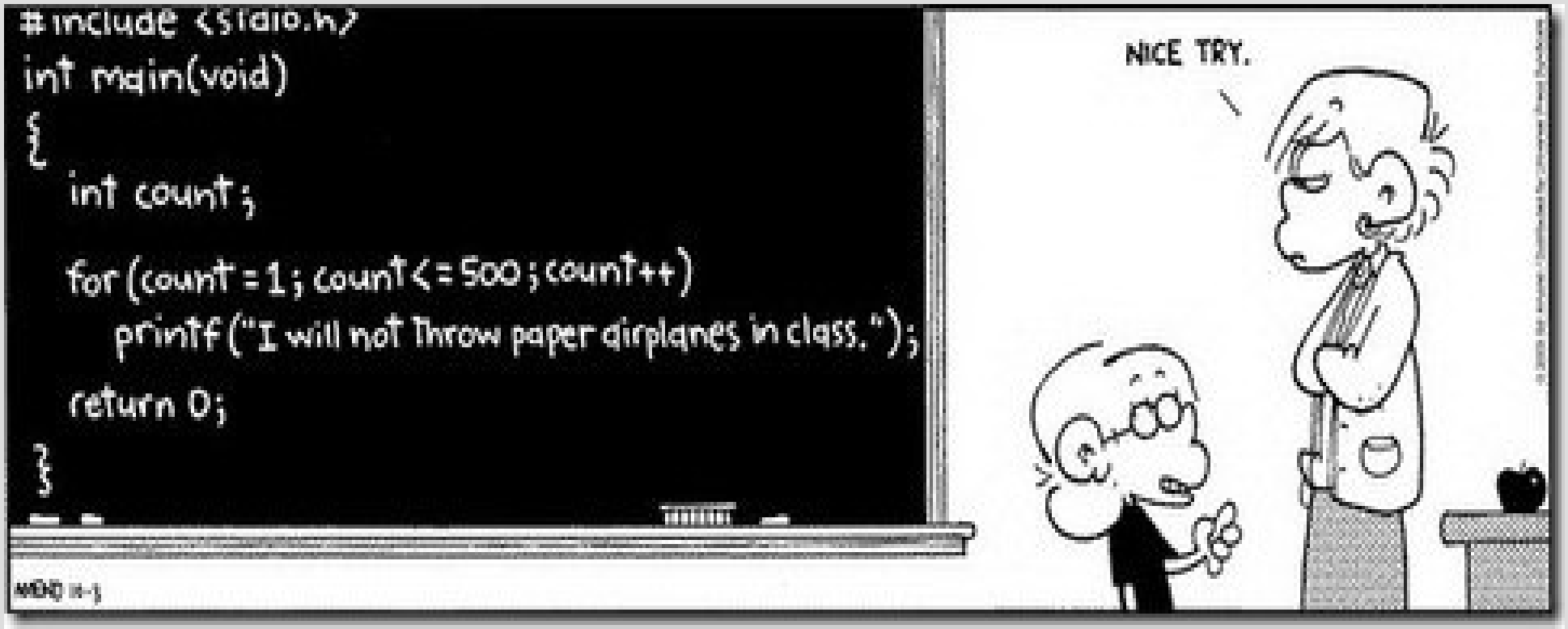
More loops

Ch 3.3-3.4



Announcements

Maybe get started on this HW early...

For remote connecting, new instructions if you are having trouble using “excute” from geany

Quiz next week!
-Covers up to (and including) HW1 (week 1-3)
-Topics: cout/cin, types, scope, if/else, etc.

Review: Loops

We put a loop around code that we want to run more than once

If we have an easy sequence (0, 1, 2, ... 10) of values we want to go over, for loop is nice

Otherwise, the while loop is a bit more general and is typically more useful if we are asking the user to control the loop

Review: Loops

Write a program that asks the user to input a value, then show the sum from 1 to that value in the following format:

```
Find the sum from 1 to what value?   5

1+2+3+4+5 = 15
```

(See: sumToN.cpp)

Nested for loop

Now modify the code so it shows all sums less than or equal to the entered values, as such:

```
Find the sum from 1 to what
value?   4
1 = 1
1+2 = 3
1+2+3 = 6
1+2+3+4 = 10
```

(See: sumAllToN.cpp)

Nested for loop

Like nested if statements, we can also make nested loops (which can cause headaches)

It might help to think of each loop as an added dimensions:

- 1 loop = 1 dimension (line/ruler)
- 2 loops = 2 dimensions (plane/square/area)
- 3 loops = 3 dimensions (volume/cube)
- ...

(See: nestedLoop.cpp)

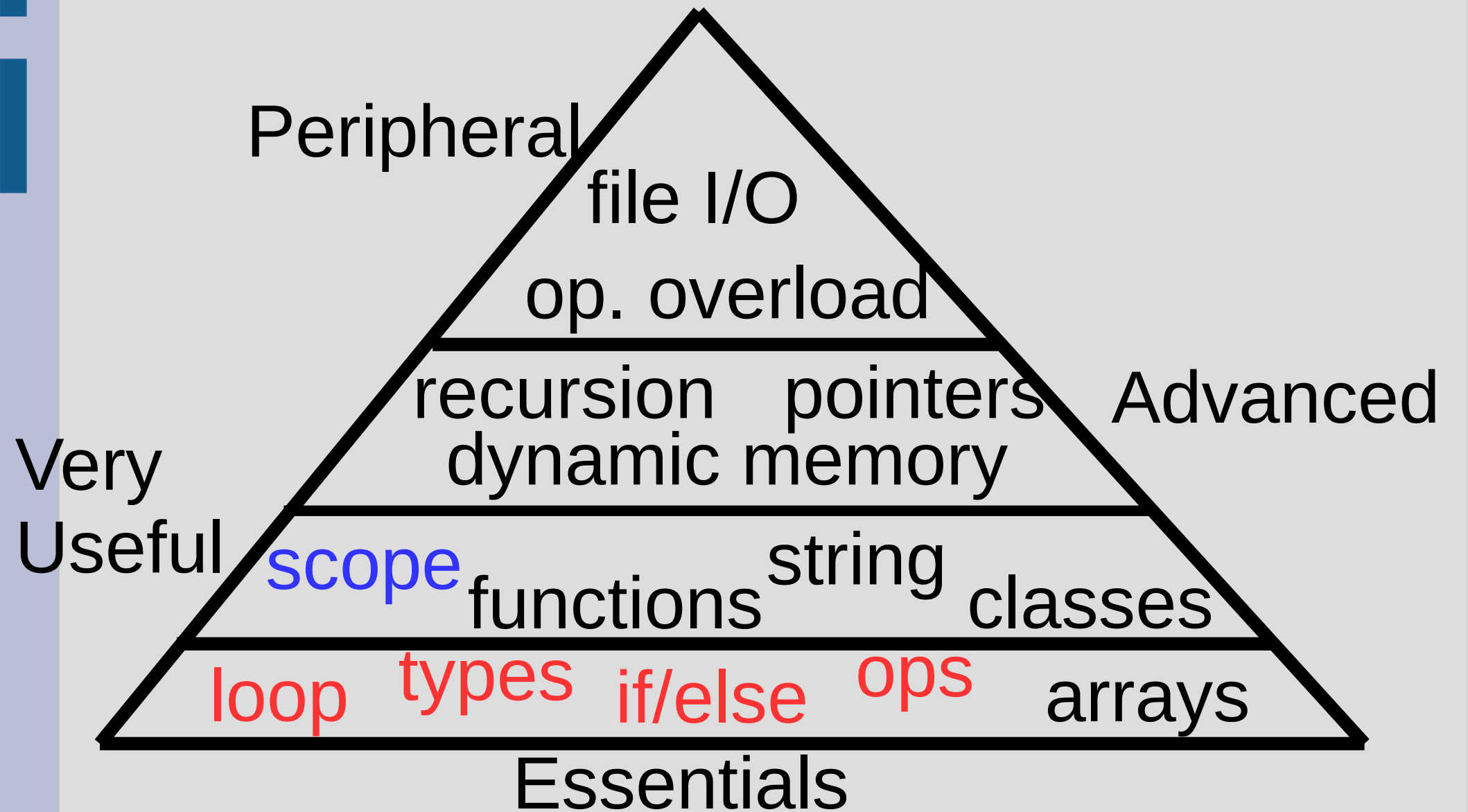
Nested for loop

Ask the user for a size of matrix, then show the identity matrix for that dimension:

```
What size?  4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

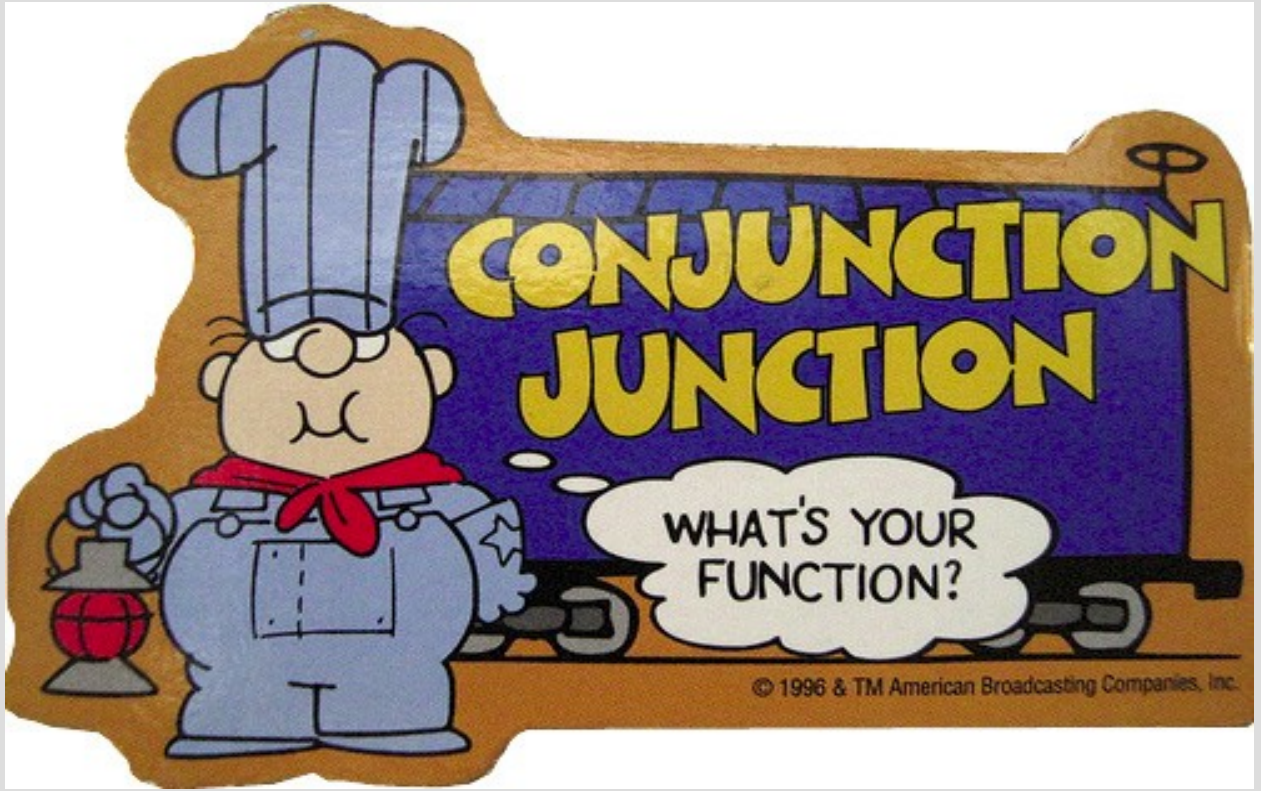
(See: identityMatrix.cpp)

Overview


Peripheral
file I/O
op. overload
recursion
pointers
Advanced
dynamic memory
Very
Useful
scope
string
functions
classes
loop
types
if/else
ops
arrays
Essentials

Functions

Ch 4-5



Functions

So far we have been writing code inside main() without understanding some parts of it

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

int main()
{
    cout << "Hello world!" << endl;

    return 0;
}
```

copy paste this, else computer throws fit

Dunno what this does but I can forget it and computer doesn't care

Why zero?

Functions

Can think of methods as packaging multiple commands into one

Functions

An analogy might be a wallet/purse

If you want to pay someone, it is easier to find your cash/card/check if organized

Functions

(Side note: you want to keep functions as simple as possible... if you try to use them to do too many things, they get bulky and harder to use)

Functions

We have used functions before, such as sqrt(), pow() or possibly round()

You can also create your own similar to creating variables by:

(1) declaring the function
(2) defining what the function does

(See: sayHi.cpp)

Functions

```
int sayHi();

int main()
{
    sayHi();

    return 0;
}

int sayHi()
{
    cout << "Howdy, I'm a computer!\n";
    return 0;
}
```

Function declaration (put before main or any other definition)

Function definition

Functions

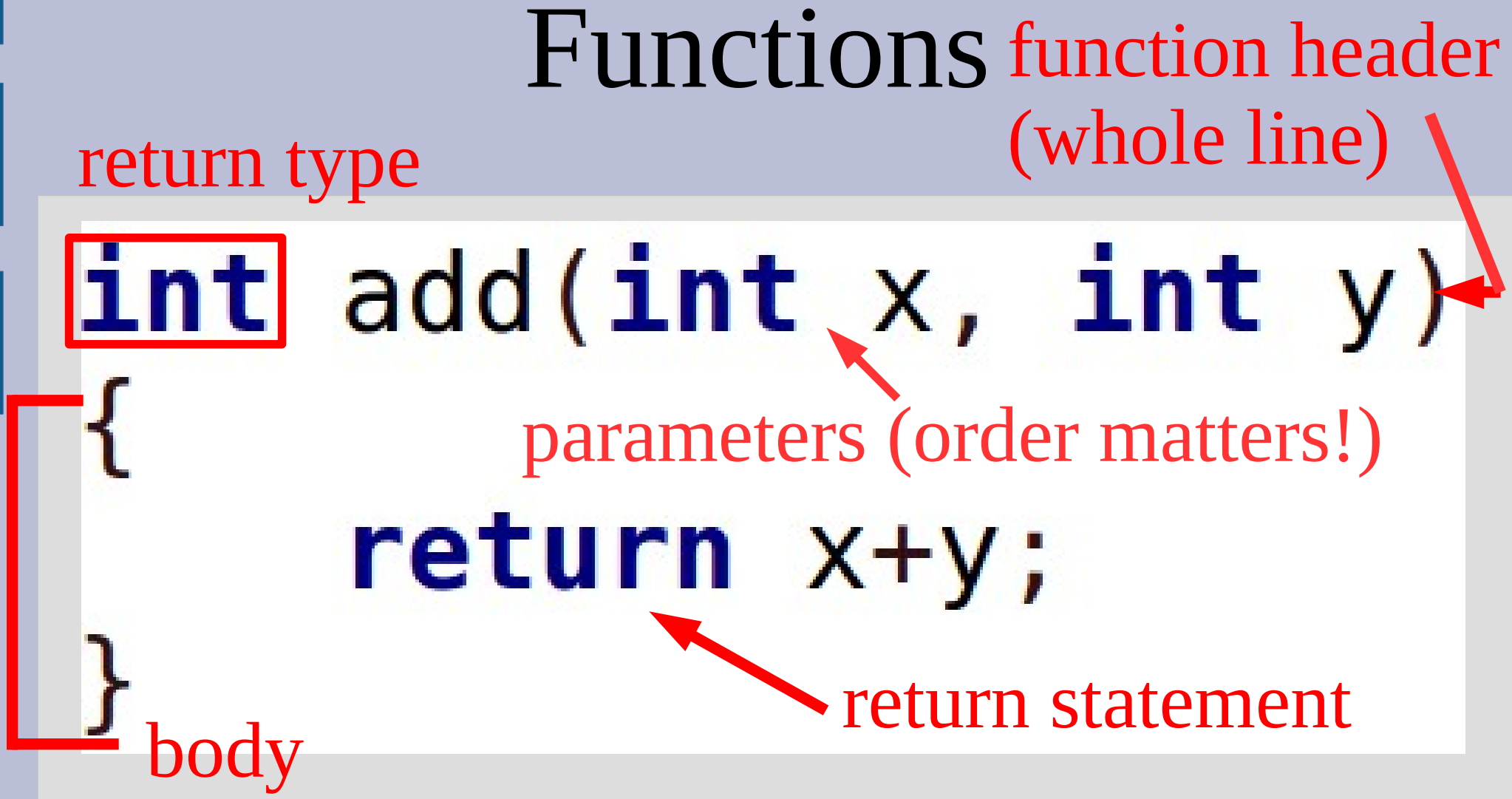
Functions, like variables, have types (int, double, char, etc.)

We call them the return value, as it is what the function will become after being finished

For example: sqrt(4) will become 2.0 (double) when it is finished

(See: addition.cpp)

Functions



The return statement value must be the same as the return type (or convertible)
(See addition2.cpp)

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Functions

You can actually have multiple functions with the same name, as long as the arguments are different either by:
- a different amount of arguments
- different types of arguments

This is called overloading a function

(See overloading.cpp)

Functions

You can make functions return type void, but not variables (an empty variable? ehh...)

This means nothing is returned, so you will get an error if you say:

```
void x();
 ... then ...
int y = x(); // x not an int! or anything!
```

void functions might just print out something

Functions

(See maze.cpp)

Functions

It is important to note that the code will resume after the function call where it was used

For example, sqrt(4) will return the value 2.0 where it was used and the rest of your code will continue

Where does the maze code return to?

Functions

Multiple function uses/calls create a “stack” much like pancakes: every time you use a function, it will add another pancake

When you return, the top pancake is removed

main() is the bottom pancake

Functions

How to make the person run?

(See: runForest.cpp)

Functions

You can also use functions that return bool types in an if statement or loop

This is commonly used if you have complex logic as it is normally easier to write a function that have a very complex bool expression

(See: findPrime.cpp)

scope

(See: sillySwap.cpp)

Typically the value of variables is copied and not given access to the real value

This is similar to moodle, the score you see for grades cannot change the score I give you!

scope

Blocks (inside { }) of code can only see variables from their parent blocks

You can also make global variables outside of all blocks
(almost as if your whole program has a start and end brace around it)

(See: globalVariable.cpp)

scope

You can give away your memory location by using “call by reference” with functions

This will share the variable between the two functions, namely the function that is using the references (&) can modify the value

```
void realSwap(int &x, int &y)
```

(See: callByReferenceSwap.cpp)

scope

We will talk more about the difference between a variable's memory location and value later

For now, a memory location (or reference) will give a function full access to modify the value

References

When memory does not actually hold the value of an object, but instead holds information about the actual location...

(See: changeInt.java)

... this is called a reference

References

If you use a normal function (call by value) then you will essentially make a photo copy of the variables
(makes 2 variables, does not effect other)

If you use call-by-reference, you have only one variable, but share it between you two

This is similar to a website link, if two people follow the link they end up in the same page

Debugging

-Test small pieces of code at a time
-Add cout statements to see values in loops (and to localize error in general)
-Test code on inputs you know the answer