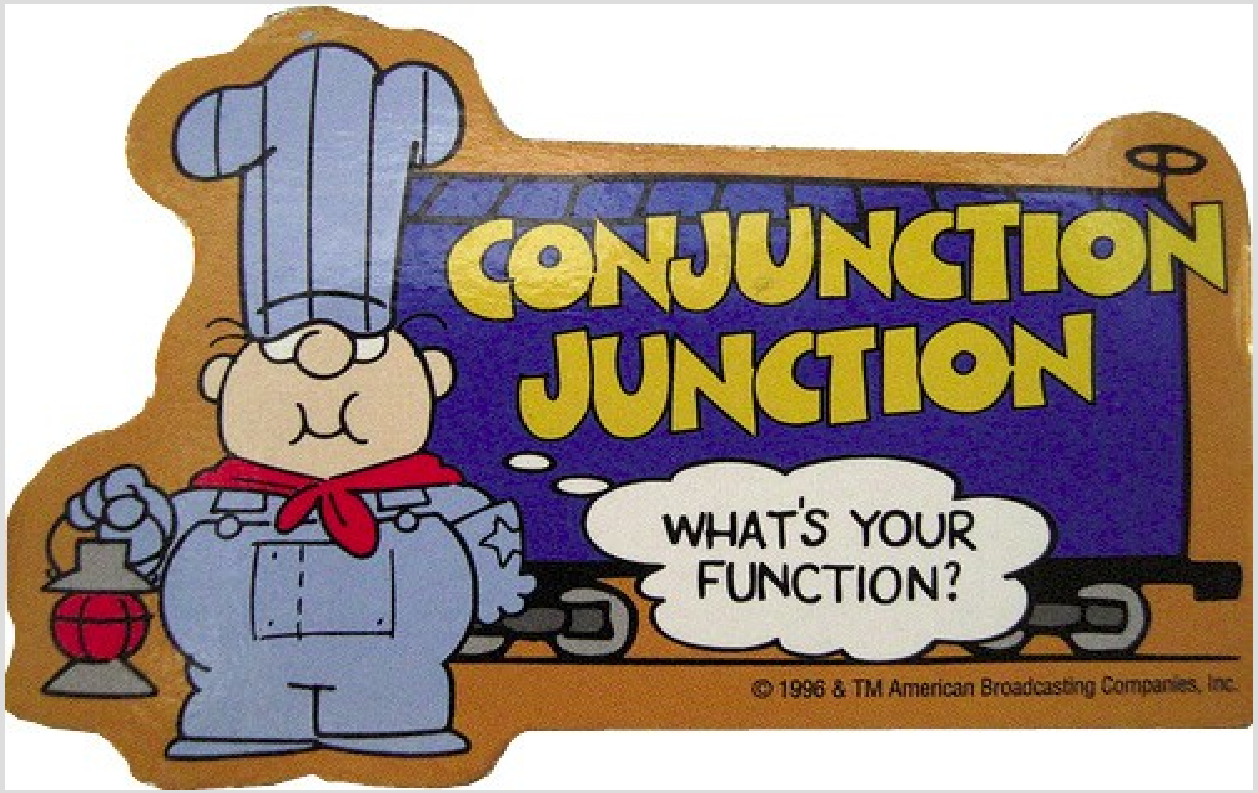
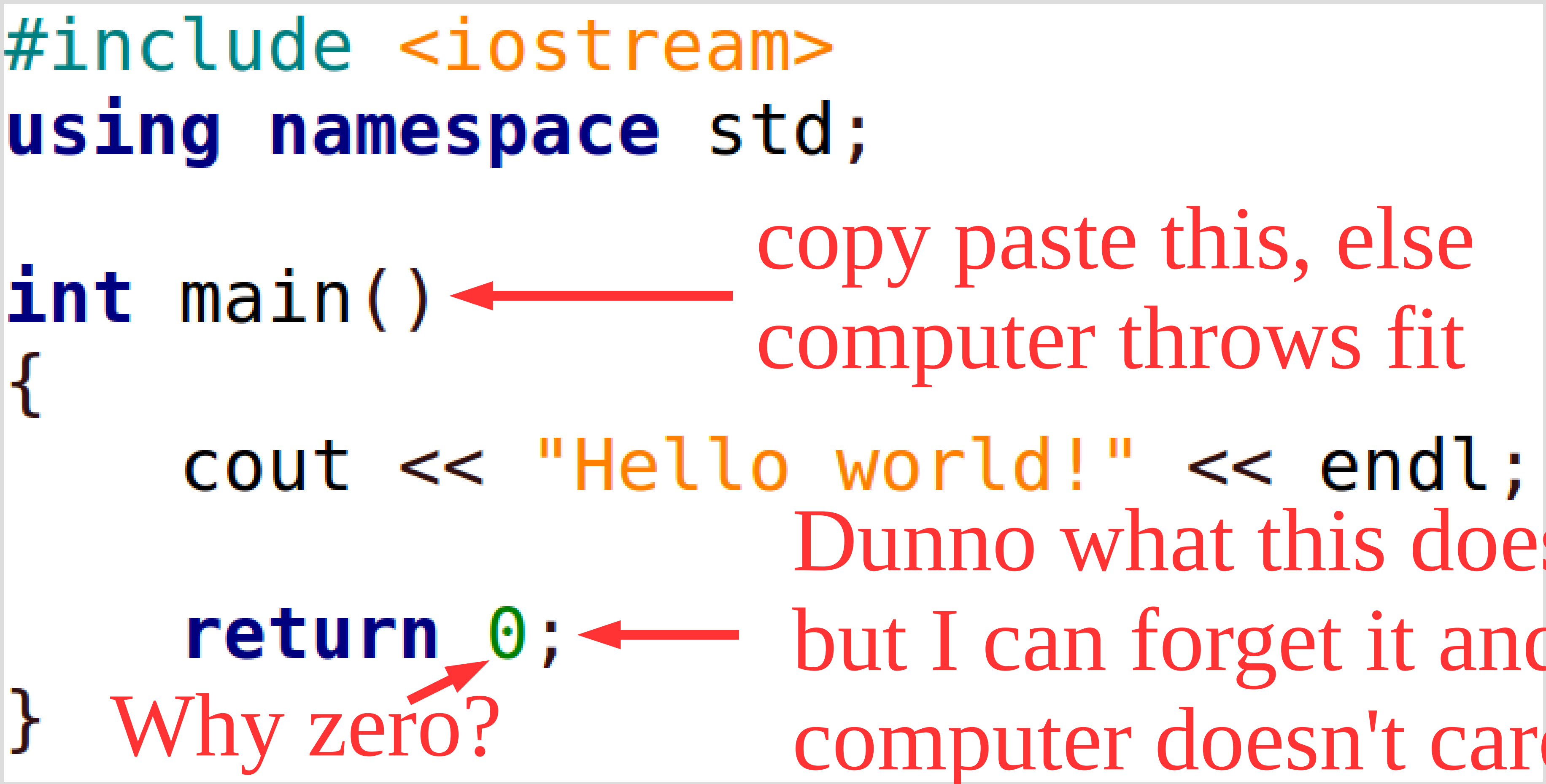
Functions

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

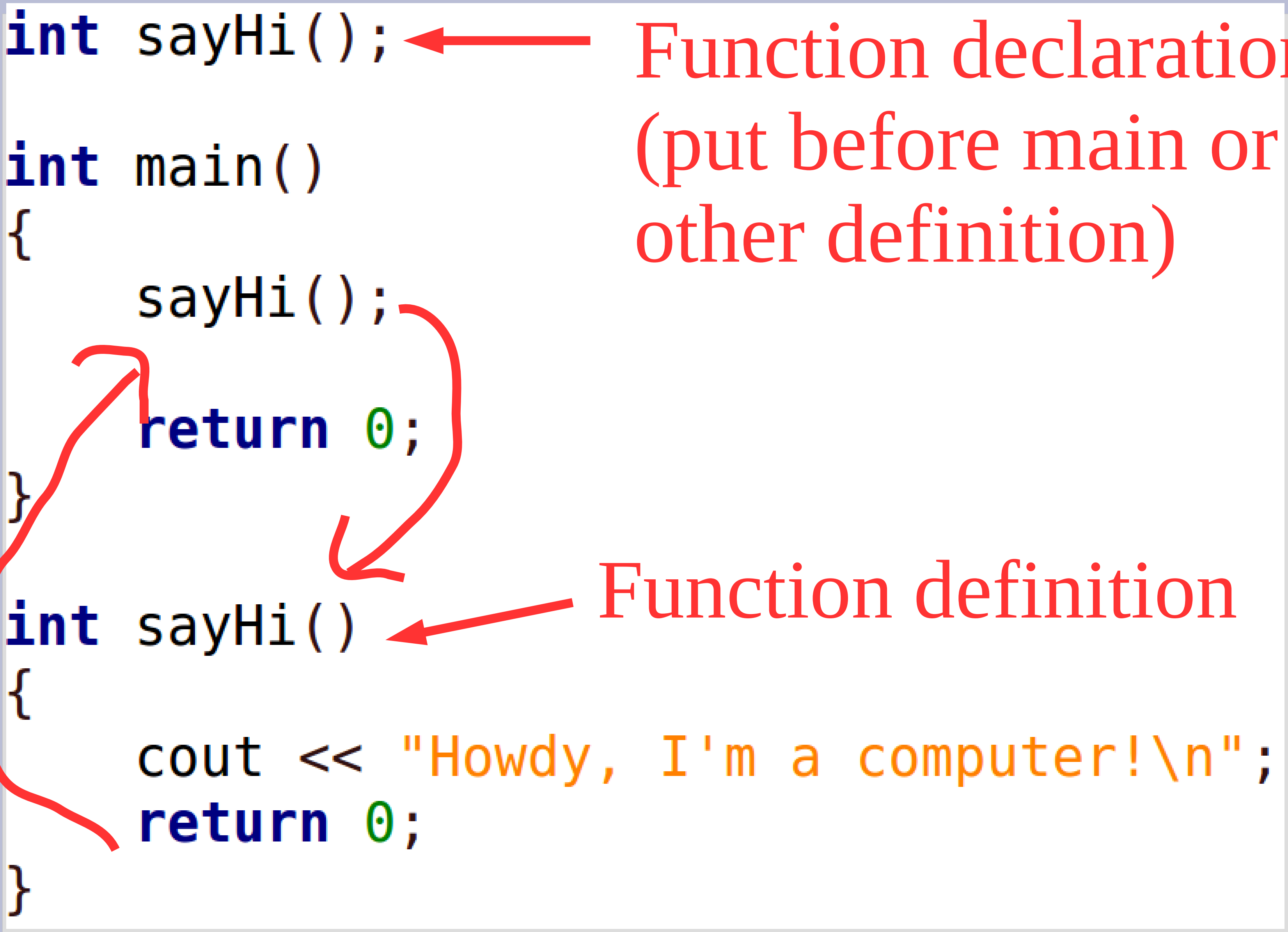
We have used functions before, such as sqrt(), pow() or possibly round()

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

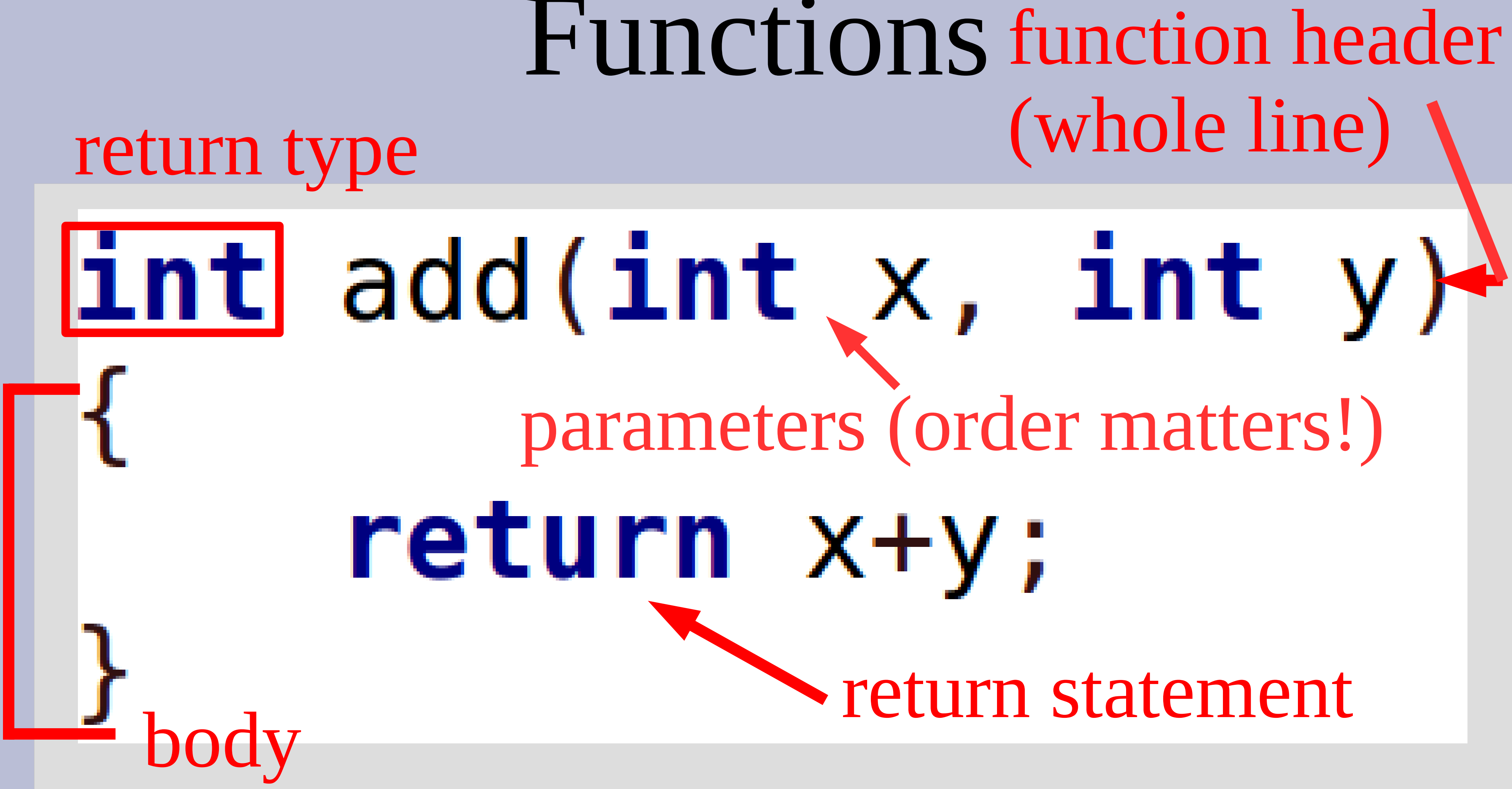
Functions, like variables, have types (int, double, char, etc.)

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



```
int add(int x, int y)
{
      return x+y;
}
```

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

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 <u>void</u>, 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