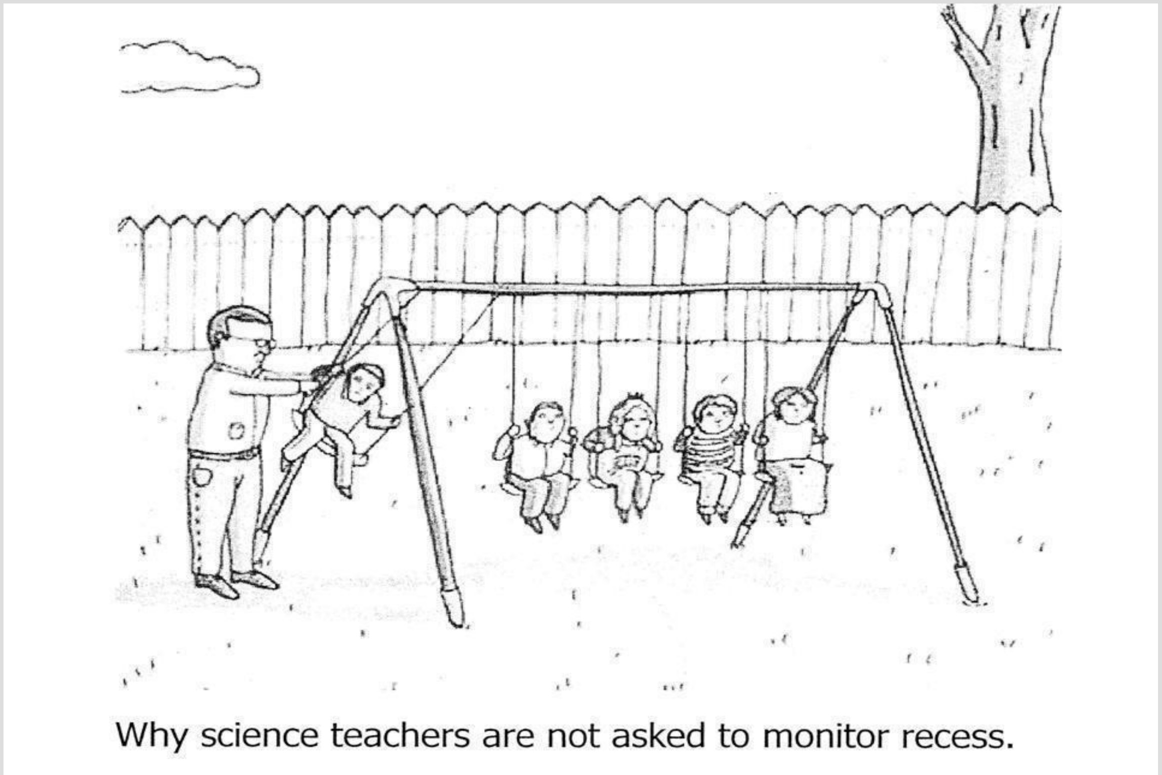
Arrays (and strings)

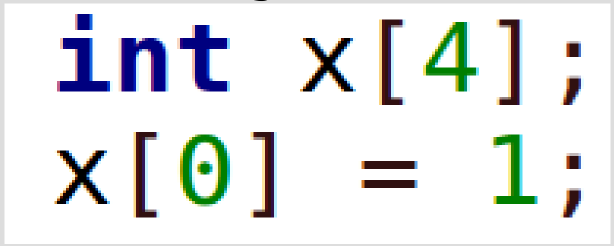
Ch 7



Why science teachers are not asked to monitor recess.

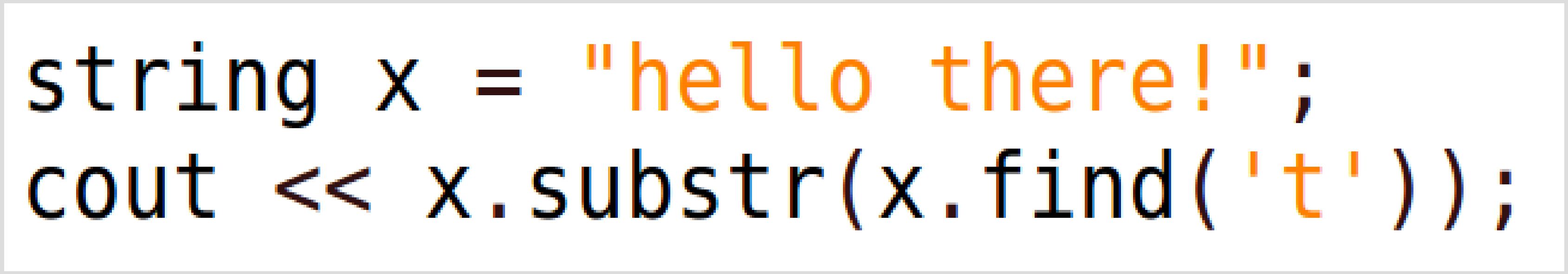
Highlights

- arrays



```
int x[4];
x[0] = 1;
```

- string functions



```
string x = "hello there!";
cout << x.substr(x.find('t'));
```

string

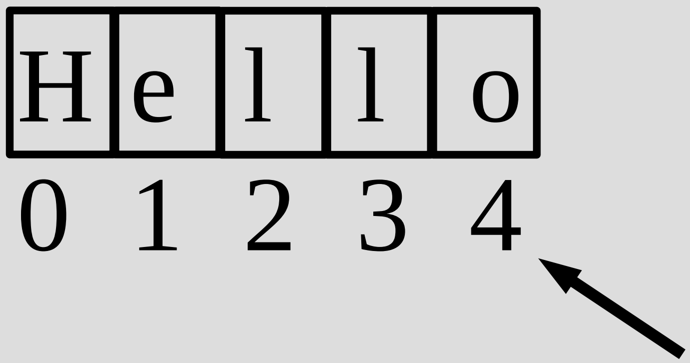
We have been using strings to store words or sentences for a while now

However, when we type “string x” it does not turn blue, as it is not a fundamental type (like char)

strings are basically a grouping of multiple chars together in a single variable

string index

String greeting = “Hello”;



The position of a character is called its <u>index</u>.

Note that the index starts from zero, not one (this is just to make your life miserable)

string functions

String greeting = “Hello”;

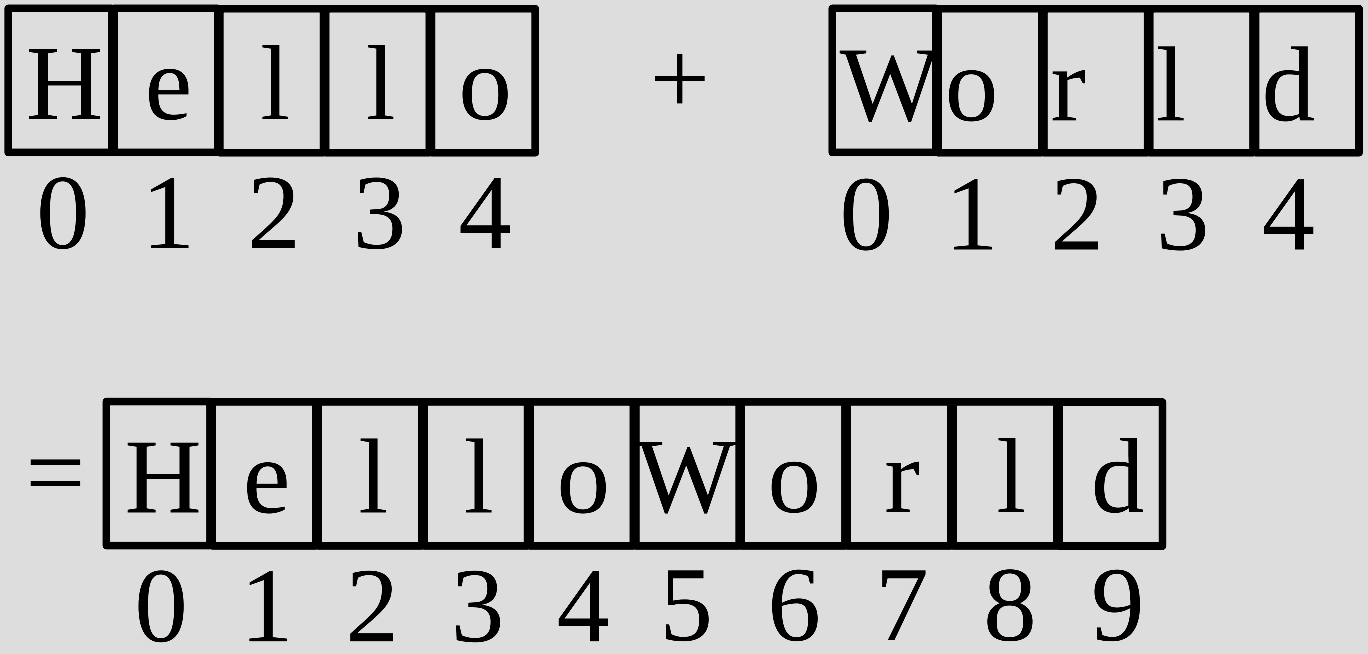
H	e	l	l	o
0	1	2	3	4

greeting.length();

$\hookrightarrow$ returns value 5 (int)

Tells how many characters are in the variable

string concatenation



String concatenation does not automatically add a space
(see: stringConcatenation.cpp)

strings

There are also some other useful functions (see book or google for a full list)

Some of the more useful ones are:

- .at(int index): character at the index
- .find(): finds first character or string
- .substr(int start): pulls out part of the original string

(see: string.cpp)

Arrays

Arrays are convenient ways to store similar data types (like multiple chars for a string)

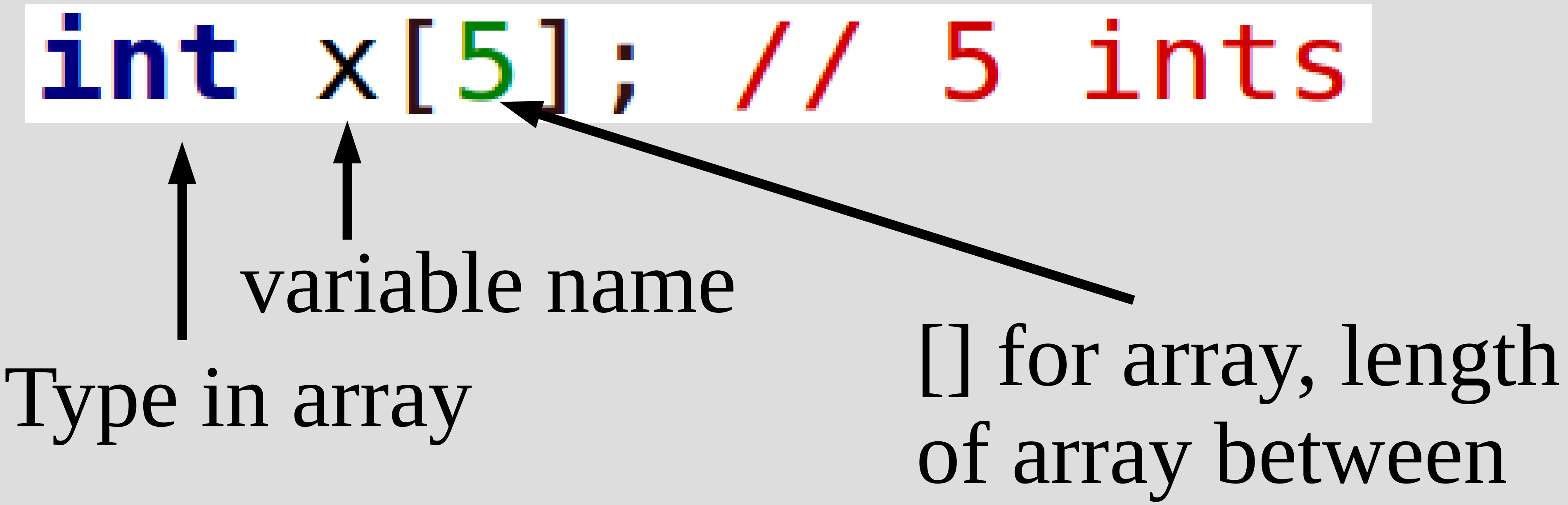
Arrays are indexed starting from 0,
so index 0 is the first element,
index 1 is the second element ...

Unlike strings, you can make an array of whatever type you want (any type!)

Arrays - declaration

When making an array, you need both a type and a length

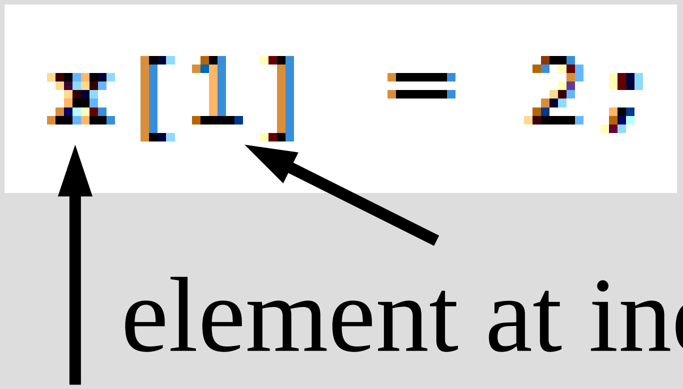
The format for making an array is below:



```
int x[5]; // 5 ints
```

Arrays - elements

To access an element of an array, use the variable name followed by the index in [ ]



```
x[1] = 2;
```

element at index

variable name

(See: simpleArray.cpp)

Arrays

Note that the number in the [ ] is inconsistent:

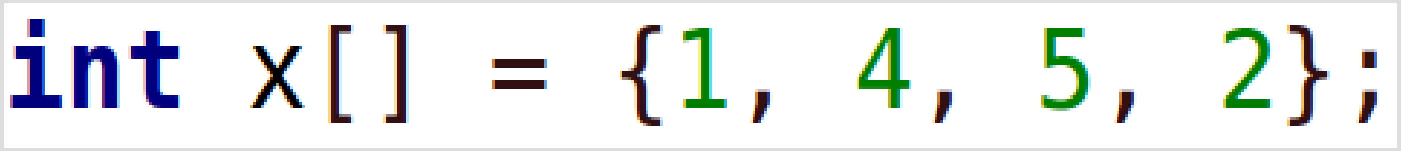
1. First time (declaration): this is the length

2. All other times: this is the index of a single
   value inside the array

If you want to indicate a whole array, just use the variable name without any [ ]
(more on this later)

Arrays - manual initialization

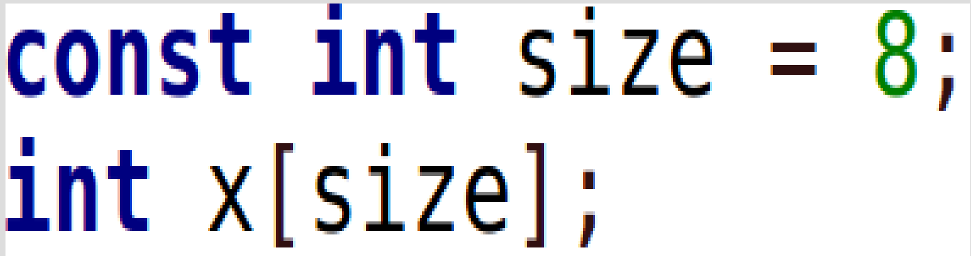
Arrays can be initialized by the following:
(must be done on declaration line!)



```
int x[] = {1, 4, 5, 2};
```

If you access outside of your array you will either crash or get a random value

You can also use a constant variable to set the size:
(See: average.cpp)



```
const int size = 8;
int x[size];
```

Arrays

When you make an array, the computer reserves space in memory for the size

The array variable is then just a reference to the first element's memory location

The computer simply converts the index into an offset from this initial location (see arrayAddress.cpp)

Memory

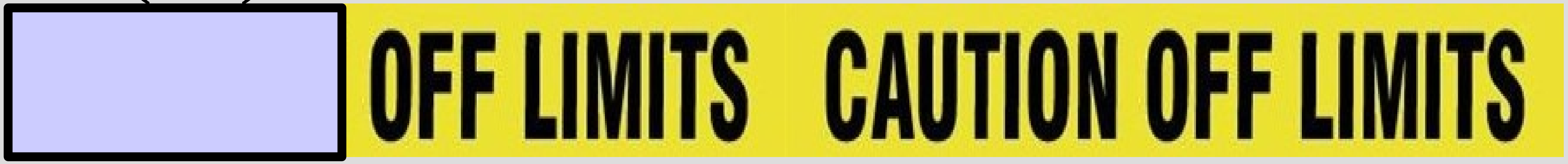
Memory:

Code:

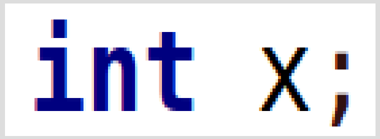
Memory (declaration)

Memory:

#0 (int) x



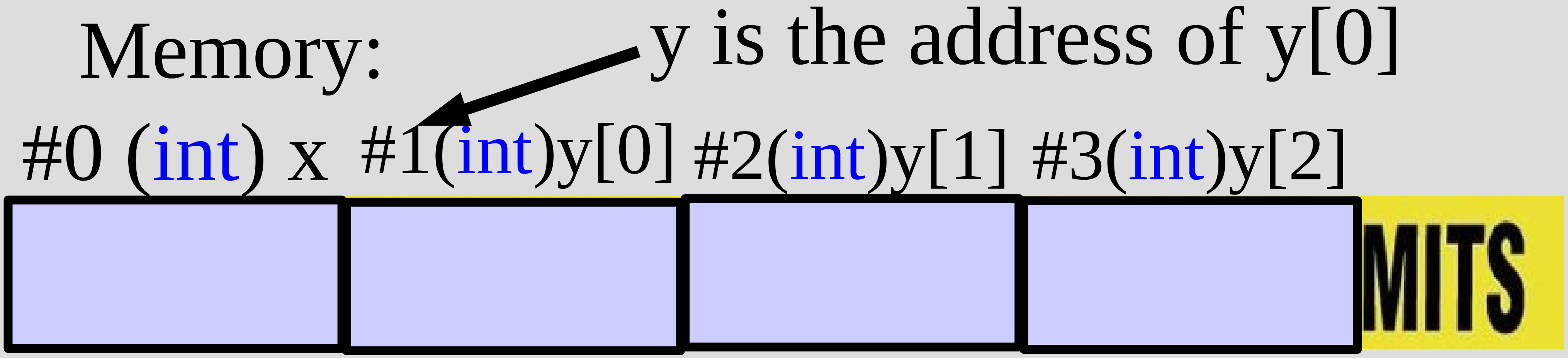
Code:



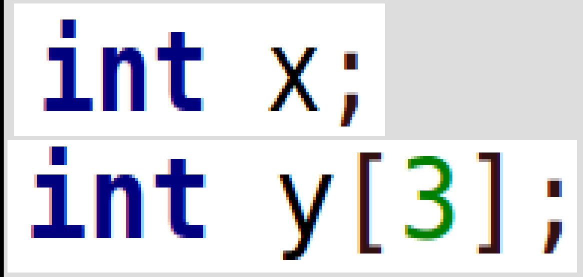
```
int x;
```

Memory (declaration)

Memory:



Code:

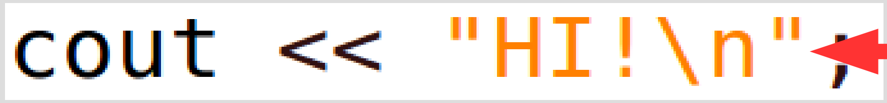


```
 int x;
int y[3];
```

C-Strings and strings

There are actually two types of “strings” (multiple characters) in C++

A C-String is a char array, and this is what you get when you put quotes around words



```
cout << "HI!\n";
```

C-String

A string (the thing you #include) is a more complicated type called a class (few weeks)

C-Strings and strings

It is fairly easy to convert between C-Strings and strings:

```
char cString[] = "move zig";
string IMAstring = cString;
cout << IMAstring.c_str() << endl;
// above converts it back to C-String
```

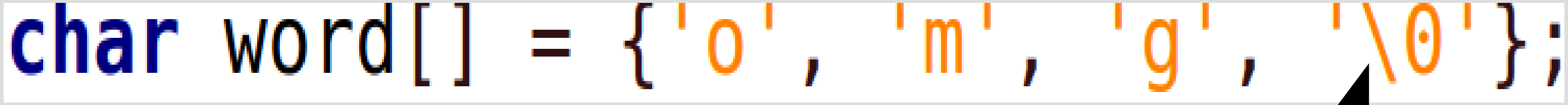
You can also convert between numbers and strings:

```
char number1[20];
string number2;
cin >> number1 >> number2;
cout << "sum is: " << (atof(number1) + stod(number2)) << endl;
```

(see: stringConversion.cpp)

C-Strings and strings

C-Strings are basically strings without the added functions



```
char word[] = {'o', 'm', 'g', '\0'};
```

You should end C-Strings with <u>null character</u>, as this tells cout when to stop displaying

This means you can initialize char arrays with quotes (**BUT NOT OTHER ARRAYS**) (see: cstring.cpp)