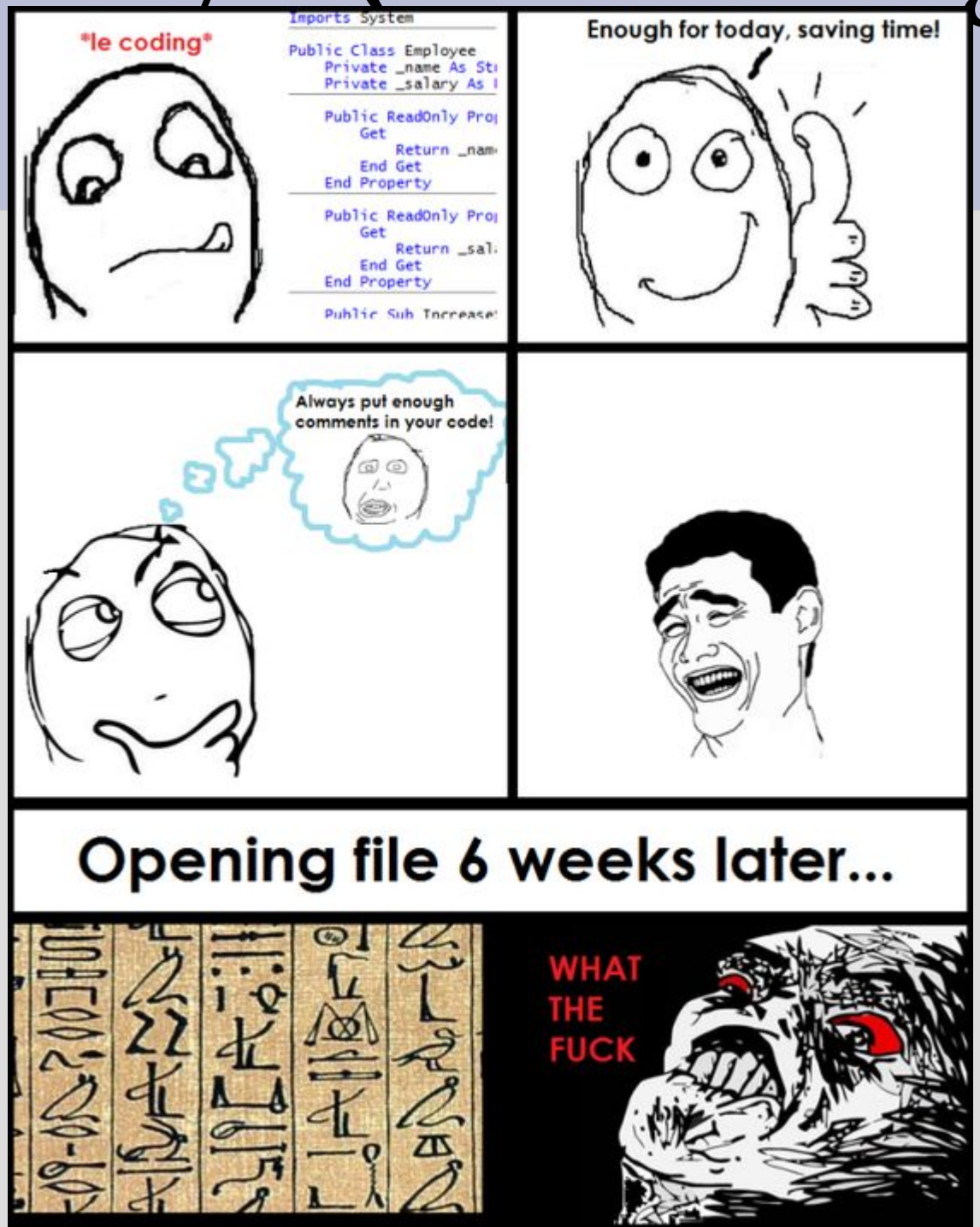
Arrays (and strings)



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
```

variable name

Type in array

[] for array, length of array between

Arrays - elements

To access an <u>element</u> 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:

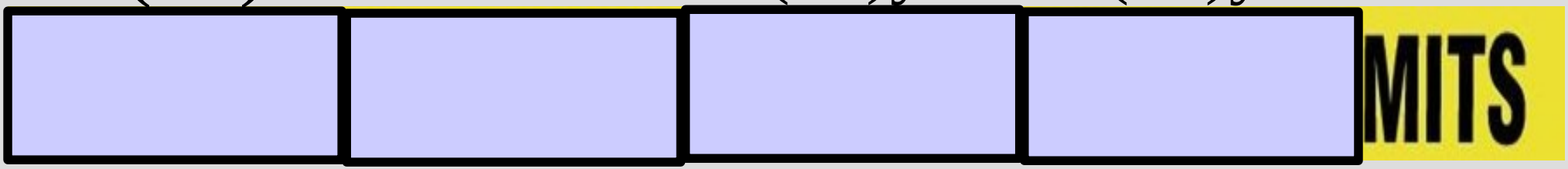
```
int x;
```

Memory (declaration)

Memory:

y is the address of y[0]

#0 (int) x #1(int)y[0] #2(int)y[1] #3(int)y[2]



Code:

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

C-Strings and strings

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

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

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

C-String

A <u>string</u> (the thing you #include) is a more complicated type called a <u>class</u> (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)