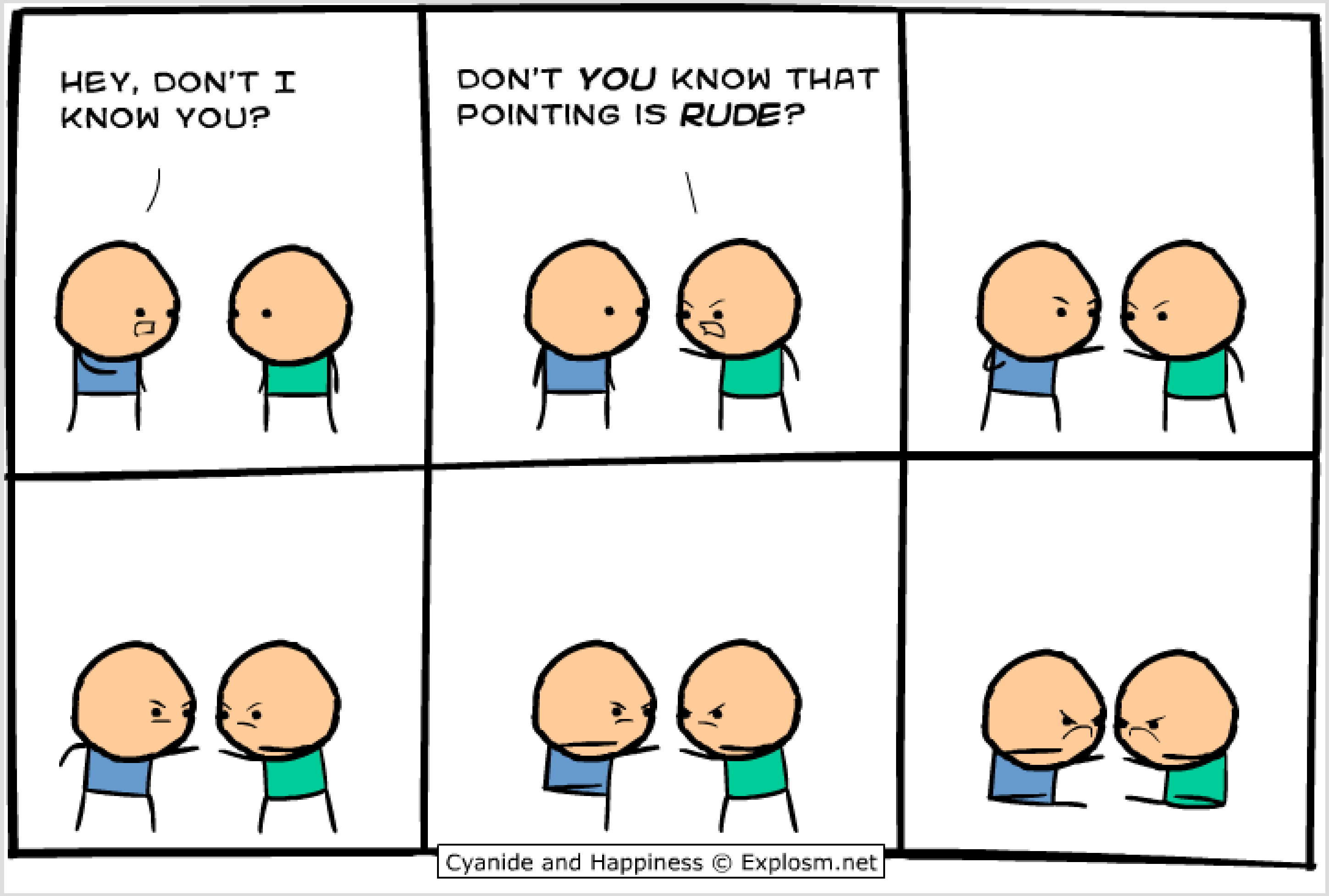
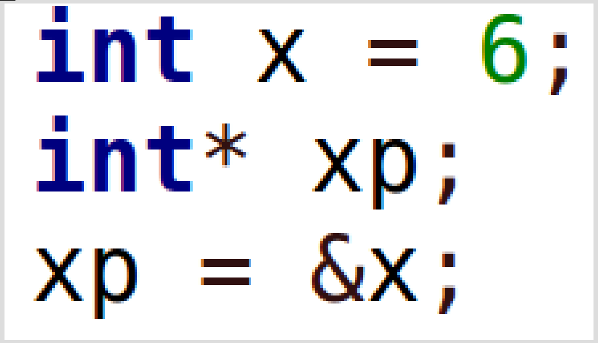
Pointers

Ch 9 & 13.1


HEY, DON'T I KNOW YOU?
DON'T *YOU* KNOW THAT POINTING IS *RUDE*?
Cyanide and Happiness © Explosm.net

Highlights

- pointers



```
int x = 6;
int* xp;
xp = &x;
```

object vs memory address

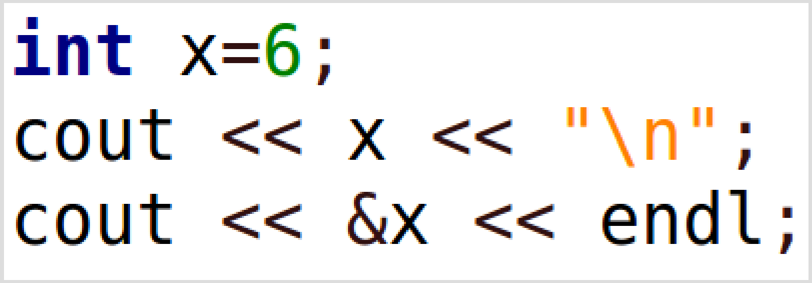
An object is simply a box in memory and if you pass this into a function it makes a copy

A memory address is where a box is located and if you pass this into a function, you can change the variable everywhere

Memory address	Object (box)
arrays (pointers)	int, double, char, ...
using &	classes

Review: address vs value

Consider the following:



```
int x=6;
cout << x << "\n";
cout << &x << endl;
```

x is a variable (a box containing value 6)

&x is a memory address (sign pointing to box)

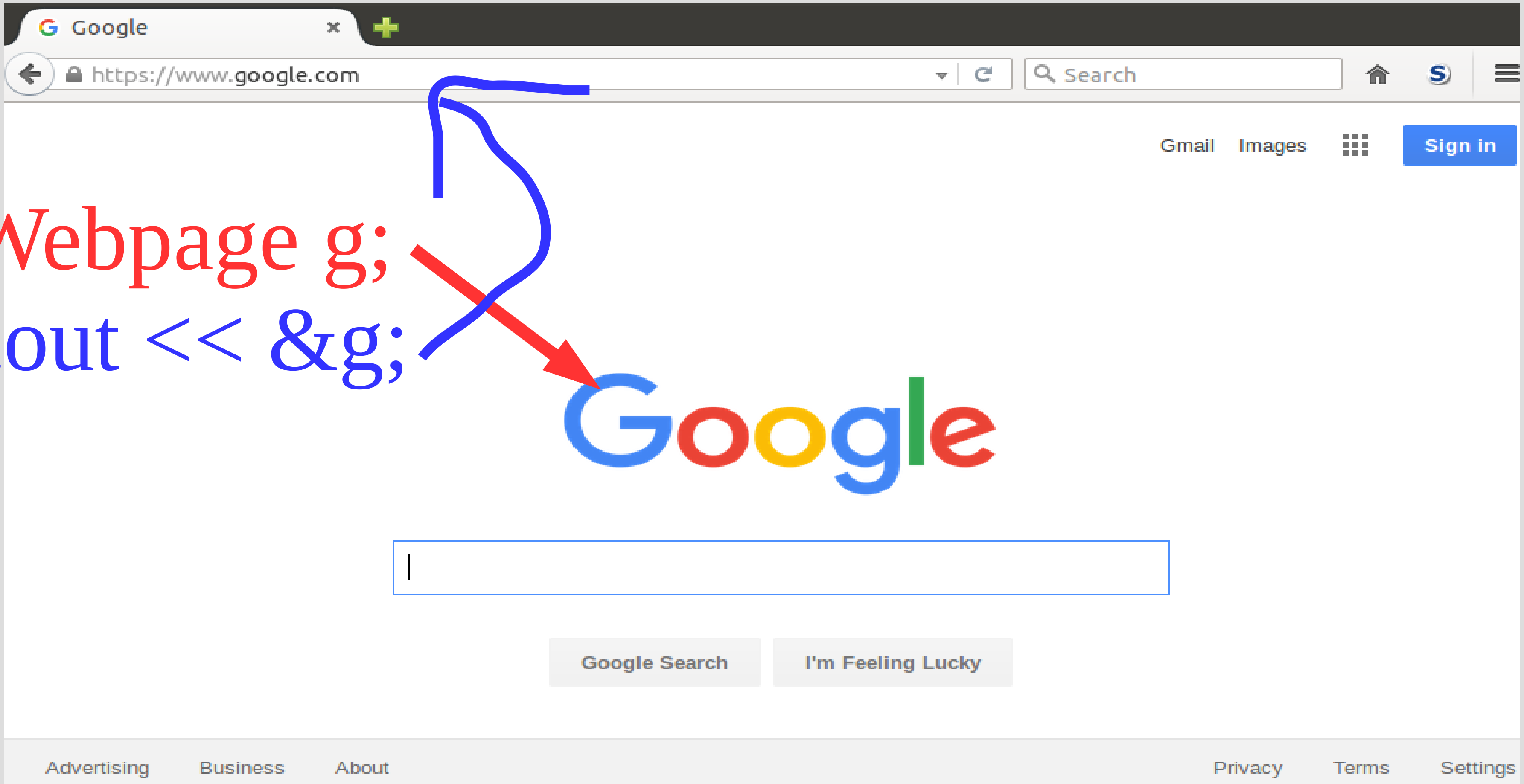
- Rather than giving the value inside the box, this gives the whole box

(see: memAddress.cpp)

Review: address vs value

Similar to a URL and a webpage

-A URL is not a webpage, but a link to one



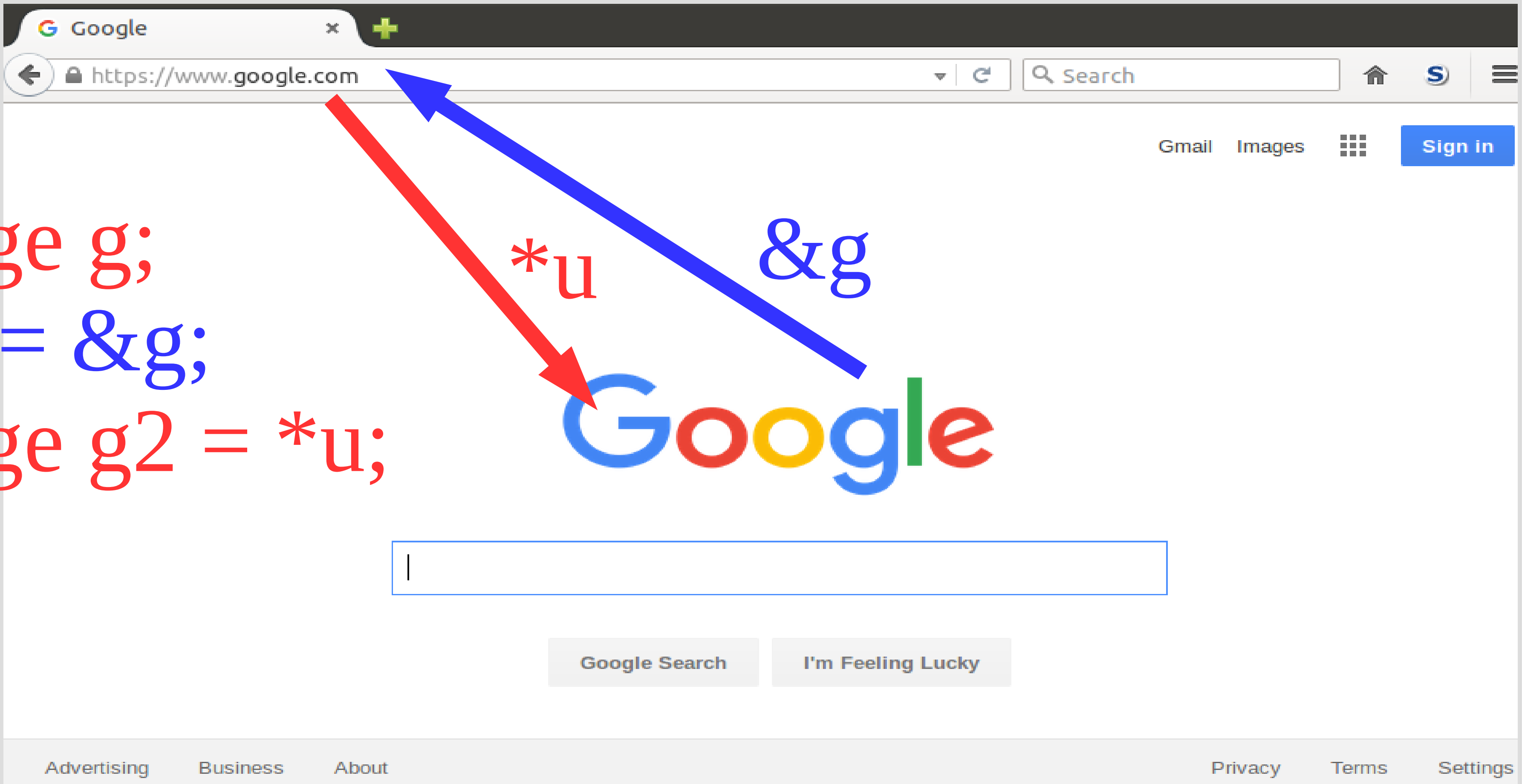
Webpage g;

cout << &g;

Pointers

Just as & goes from value (webpage) to address (url), * goes the opposite:

Webpage g;
URL u = &g;
Webpage g2 = *u;



Pointers

You can also think of pointers as “phone numbers” and what they point to as “people”

Trump
(object)

1-800-presdnt
(pointer)

Pointers

If multiple people have the same “phone number”, they call the same person (object)

Trump
(object)

1-800-presdnt
(pointer/
memory address)

1-800-presdnt

Pointers

A pointer is used to store a memory address and denoted by a * (star!)

```
int x = 6;
int* xp;
xp = &x;
```

Here variable "xp" has type "integer pointer"

```
cout << *(&x); // *(&x) same as  x
```

The * goes from address to variable (e.g. like hitting ENTER on a url, or "call" on a phone contact) (See: pointerBasics.cpp)

Pointers (phone analogy)

```
int* jacky;
```

Make a contact name called “jacky”

Make a phone-number for an person (int)

```
int Jackeline_Wu = 9;
```

Make a person (int) “Jacqueline Wu” exist

```
jacky = & Jackeline_Wu;
```

(& = address of)

Save Jacqueline Wu's phone number into the “jacky” contact

```
*jacky = 9001;
```

* = call up

Call the “jacky” contact (and connect with Jacqueline Wu)

Pointers

It is useful to think of pointers as types:

```
int* xp;
```

Here I declared a variable “xp” of type “int*”

Just like arrays and [], the use of the * is different for the declaration than elsewhere:

Declaration: the * is part of the type (`int* xp;`)
Everywhere else: * follows the pointer/address
(i.e. `*xp = 2;` puts 2 where xp is pointing to)

Pointers

Pointers and references allow you to change anything into a memory address that you want

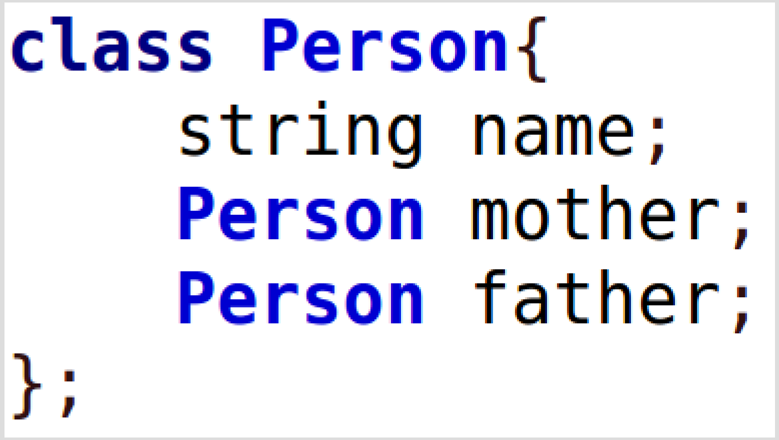
This can make it easier to share variables across functions

You can also return a pointer from a function (return links to variables)
(see: returnPointer.cpp)

Pointers

Why do we need pointers? (memory addresses are stupid!!!)

Suppose we had the following class:



```
class Person{
    string name;
    Person mother;
    Person father;
};
```

Will this work?

Pointers

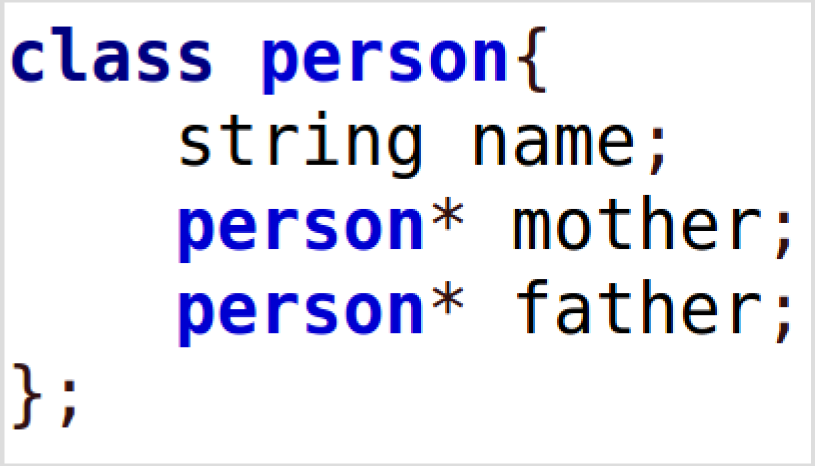
As is, it will not... it is impossible to make a box enclose two other equal sized boxes

The only way it can enclose something like itself is that thing is smaller

Pointers

To do this we can use pointers instead!

A pointer does not store the whole class data, it only remembers where it is (like a URL)



```
class person{
    string name;
    person* mother;
    person* father;
};
```

(See: person.cpp) (more on this shortly)

->

When dealing with classes, often you need to deference (*) and access a member (.)

There is a shortcut to de-reference and call a member (follow arrow and go inside a box)

You can replace (*var).x with var->x, so...

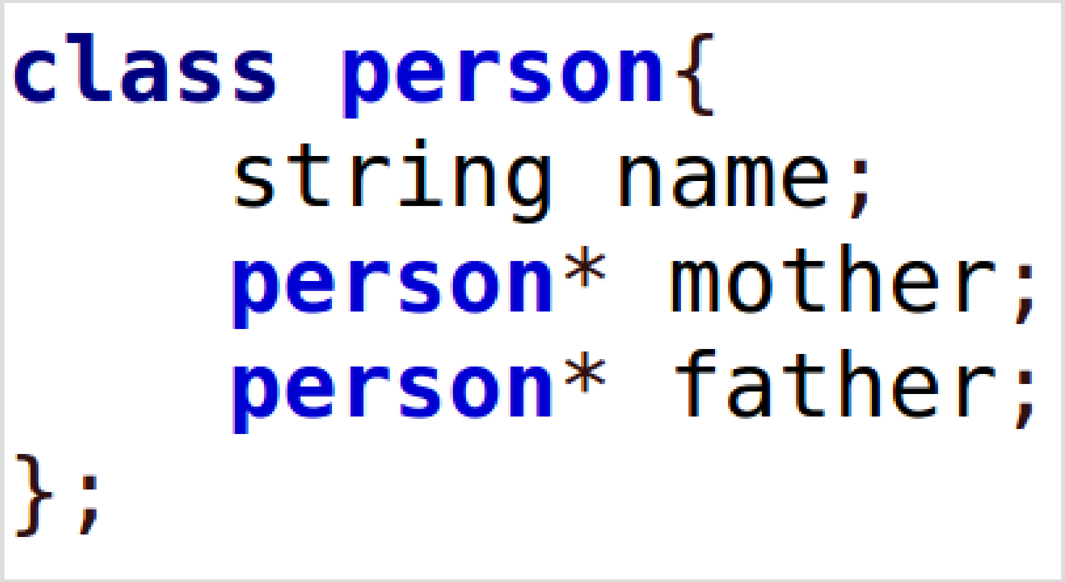
```
(*(me.mother)).name;
```

... same as ...

```
me.mother->name;
```

Person class

How would you make your grandmother?
How could you get your grandmother using only yourself as a named object?



```
class person{
    string name;
    person* mother;
    person* father;
};
```

(See: personV2.cpp)