C++ Rule of Three

Compiling and running C++ code

- **See:**

https://courses.engr.illinois.edu/cs225/fa2017/resources/own-machine/

- Mac OS X, Linux: very straight-forward
- Windows: best option for this class might be FastX
  - https://it.engineering.illinois.edu/user-guides/remote-access/connecting-ews-linux-fastx

Compiling C++

- **Clang: C language compiler**
  - clang for C programs
  - clang++ for C++ programs

- **Important arguments / options:**
  - Names of the C/C++ source files (not header .h files)
  - -std=c++0x *To specify which version of C++ standard*
  - -o outputfilename *By default it creates a file called a.out*

- **For example:**
  - clang++ -std=c++0x main.cpp number.cpp -o number

Makefiles (and build scripts in general)

- **A way to automate (complex) tasks**
  - Supports incremental updates via dependences
  - Used for building computer programs

- **Consist of rules (with the following structure)**

```
thing_to_make: list of things that it uses
        commands to execute to make the thing
```

- **For example:**

dependance

```
number: main.cpp number.cpp number.h
      clang++ -std=c++0x main.cpp number.cpp -o number
```

Makefiles, cont.

$$\begin{tikzcd}[column sep=small] & \text{all} \arrow[d, no head] & \\ & \text{q2} \arrow[dl, no head] \arrow[dr, no head] & \\ \text{q2.cpp} & & \text{heap-int.cpp} \end{tikzcd}$$

- Allow you to define variables

```
EXENAME = q2

CXX = clang++
CXXFLAGS = -std=c++0x -g -O0 -Wall -Wextra
```

Variable Definitions

```
                   q2
all : $(EXENAME)

     q2
$(EXENAME): q2.cpp heap_int.cpp
        $(CXX) $(CXXFLAGS) q2.cpp heap_int.cpp -o $(EXENAME)
```

Variable Use

- First rule is the default rule

Review: Copy Constructors

- **What happens when we copy an object?**

ExpressionValue myExpr(1.0);
ExpressionValue myOtherExpr = myExpr;

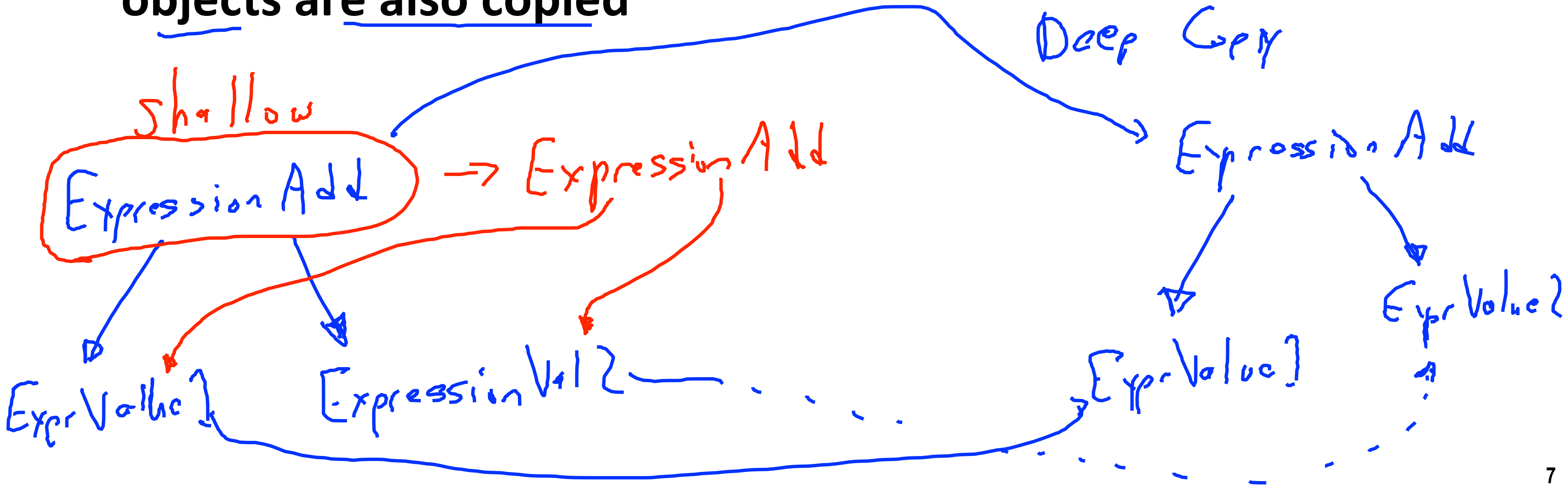
- **It invokes a copy constructor**
  - Be default, it does a bit-wise copy of the object
  - Can override, by declaring:

```
ExpressionValue(const ExpressionValue&);
```

Why override default copy constructor?

- Generally, when we want a deep copy.

- Shallow copy: bit-wise copy of the object that copies any pointers/references contained, but not the pointed to/ referenced objects
- Deep copy: occurs when all of the pointed to/referenced objects are also copied



Operator Overloading

- Unlike in Java, in C++ you can define how standard op behave

Operators that can be overloaded in C++							
Arithmetic	$\bigcirc\!\!\!\!+$	-	*	/	%	++	--
Bitwise	&	|	^	~	<<	>>	
Assignment	=						
Comparison	==	!=	>	<	>=	<=	
Logical	!	&&	||				
Other	[]	()	->				

Ev1 + Ev2

Assignment Operator

automatic: bit wise copy

- **Type &operator=(const Type &rhs);**
  - Again, useful for deep copies

Expression Value EV1; ~~= EV2;~~

EV1 = EV2;

Which is being invoked?

A) Assignment operator
B) Copy Constructor
C) Default Constructor
D) None of the above

```
ExpressionValue ev1, ev2;        //  #1

ExpressionValue ev3 = ev2;       //  #2

ev3 = ev1;                       //  #3
```

Destructors

- A function called when the object is deleted
- Defined as: ~Type()
- Again: useful when the object contains other objects, so we can delete those other objects (and not leak memory)

C++ Rule of Three

- **is a rule of thumb that if a class defines one (or more) of the following it should probably explicitly define all three:**
  - destructor.
  - copy constructor.
  - copy assignment operator.