

CS:35 Data Structures and Algorithms

Copies in C++
5/1/2013

```
int foo(int x) {
    return x;
}

int a = 2;
a = foo(a);
```

How many copies of `a` are made?

5/6/13

CS:35 - Copies in C++

2

```
class C {
    int *arr;
    // ...
};

C foo(C x) {
    return x;
}

C a;
a = foo(a);
```

Needlessly costly if `C` is large...

5/6/13

CS:35 - Copies in C++

3

Mental Model of Computation

```
int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}

int x = 1;
int *px = new int(2);
x = mystery(x, px);
```

5/6/13

CS:35 - Copies in C++

4

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}
int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack Heap

5/6/13 CS:35 - Copies in C++ 5

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}
int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack Heap

x	1
---	---

5/6/13 CS:35 - Copies in C++ 6

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}
int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack Heap

x	1
px	

→

2

5/6/13 CS:35 - Copies in C++ 7

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}
int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack Heap

x	1
px	
x	1
px	

→

2

5/6/13 CS:35 - Copies in C++ 8

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}

int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack

x	1
px	→
x	1
px	→
y	3

Heap

2

5/6/13 CS:35 - Copies in C++ 9

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}

int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack

x	1
px	→
x	2
px	→
y	3

Heap

2

5/6/13 CS:35 - Copies in C++ 10

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}

int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack

x	1
px	→
(t)	3

Heap

2

5/6/13 CS:35 - Copies in C++ 11

```

int mystery(int x, int *px) {
    int y = x + *px;
    x = *px;
    return y;
}

int x = 1;
int *px = new int(2);
x = mystery(x, px);
    
```

Stack

x	3
px	→

Heap

2

5/6/13 CS:35 - Copies in C++ 12

Exercise Your Own Mental Model

```
int foo(int x, int *y, int *z) {
    int a = x + *y;
    *y = a + 1;
    *z = *y + *z;
    y = &x;
    // POINT (B)
    return *y;
}
int a = 3;
int *b = new int(2);
int c = a + 1;

// POINT (A)
int d = foo(a, b, &c);
// POINT (C)
```

5/6/13

CS:35 - Copies in C++

13

Exercise Your Own Mental Model

```
int foo(int x, int *y, int *z) {
    int a = x + *y;
    *y = a + 1;
    *z = *y + *z;
    y = &x;
    // POINT (B)
    return *y;
}
int a = 1;
// POINT (D)
int d = foo(a, &a, &a);
// POINT (F)
```

5/6/13

CS:35 - Copies in C++

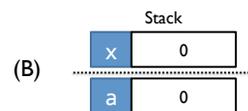
17

When Do Copies Occur?

1. Parameter passing.



2. Returns.



3. Assignment.

f(x);

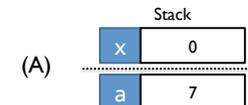
5/6/13

CS:35 - Copies in C++

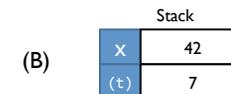
21

When Do Copies Occur?

1. Parameter passing.



2. Returns.



3. Assignment.

return a;

5/6/13

CS:35 - Copies in C++

22

When Do Copies Occur?

1. Parameter passing.



2. Returns.



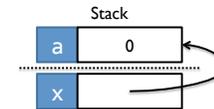
3. Assignment.

```
x1 = x2;
```

The Mighty Reference (Type)

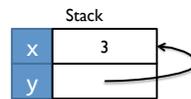
```
int f(int &x) {
    return x + 1;
}

int a = 0;
a = f(a);
```



For any type T , $T\&$ is a *reference to T* .

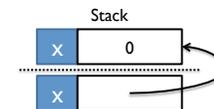
```
int f(int x) {
    int &y = x;
    return y + 1;
}
```



Dereferencing a reference is *implicit*.

```
int& f(int &x) {
    return x;
}

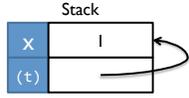
int x = 0;
f(x) = 1;
```



Reference types avoid copying values.

```
int& f(int &x) {
    return x;
}

int x = 0;
f(x) = 1;
```



Stack

x | 1

(r) | 1

Reference types avoid copying values.

5/6/13 CS:35 - Copies in C++ 27

```
int& f(int x) {
    return x;
}
```

But, like pointers, be wary...

5/6/13 CS:35 - Copies in C++ 28

References vs. Pointers

If references are just implicit pointers,
what use do they have?

5/6/13 CS:35 - Copies in C++ 29

Memory Management with Classes

```
class C {
    int *arr;
    // ...
};

C foo(C x) {
    return x;
}

C a;
a = foo(a);
```

What dictates how C is copied?

5/6/13 CS:35 - Copies in C++ 30

1. Parameter Passing.
2. Returns

The *Copy Constructor*.

```
class C {
public:
  C(const C& rhs) {
    // Copy fields of rhs to this new C
  }
};
```

(By default, performs a *shallow copy*.)

5/6/13

CS:35 - Copies in C++

31

3. Assignment

The *Copy Assignment Operator*.

```
class C {
public:
  C& operator=(const C& rhs) {
    // Assign fields of rhs into this existing C.
  }
};
```

(By default, performs a *shallow copy*.)

5/6/13

CS:35 - Copies in C++

32

The *Rule of Three*

If any one of the *copy constructor*, *copy assignment operator*, or *destructor* are implemented, then all three must be implemented (coherently) to ensure memory safety.

5/6/13

CS:35 - Copies in C++

33

Exercise!

debt-backend.cpp contains code that implements a back-end for a debt-collection agency.

Run it, test it, break it (if possible), then fix it!!

(Any questions or comments? posera@cis.upenn.edu or <http://www.cis.upenn.edu/~posera> thanks!)

5/6/13

CS:35 - Copies in C++

34