

CS481 - Object-Oriented Programming

Function Overloading

- ❖ Functions are operations
 - Especially member functions
- ❖ Operation variants exist
 - Print data item (int, float, char, etc.)
- ❖ One function name/variant?
 - C: yes, C++: programmer option

Constructor Effects

- ❖ Constructor function
 - Fixed function name (same as class)
 - Initializes an object (instance of class)
- ❖ What if more than one way?
 - E.g., complex number
 - ◆ Set real/imag value, or only real

C++ Solution

- ❖ Multiple functions
 - One for each operation variant
- ❖ One function name
 - Same name for all variants
- ❖ How to distinguish?
 - By variation in argument list

Overload Example

```
class Stack
{
  ...
public:
  ...
  void push(int element);
  void push(int elem1, int elem2);
  void push(double elem);
}
```

Overload Example

```
class Stack
{
  ...
public:
  ...
  void push(int element);
  void push(int elem1, int elem2);
  void push(double elem);
}
```

Function "signature"
(pattern of argument types)

Overload Example

```
void Stack::push
(int elem1, int elem2)
{
  push(elem1);
  push(elem2);
}
void Stack::push(double elem)
{
  push(int(elem+0.5));
}
```

CS481 - Object-Oriented Programming

Overload Example

```

void Stack::push
  (int elem1, int elem2)
{
  push(elem1);
  push(elem2);
}
void Stack::push(double elem)
{
  push(int(elem+0.5));
}
    
```

What is this?

7

Type Conversion

```

// cast notation
push((int)(elem+0.5));

// functional notation
push(int(elem+0.5));
    
```

Can also have automatic type conversion, as in C.

8

Mangling Revisited

- ❖ Common names across classes
 - Or other name scopes (e.g., global)
 - Generated unique names for linker
- ❖ Common names within class
 - For overloaded functions
 - Must still generate unique names

9

Constructor Example

```

class Complex
{
  double re;
  double im;
public:
  Complex();
  Complex(double real);
  Complex(double real,
           double imag);
}
    
```

10

Constructor Example

```

Complex::Complex()
{
  re = 0.0;
  im = 0.0;
}
Complex::Complex(double real)
{
  re = real;
  im = 0.0;
}
    
```

11

Constructor Example

```

Complex::Complex(double real,
                 double imag)
{
  re = real;
  im = imag;
}
    
```

12

CS481 - Object-Oriented Programming

Constructor Use

```
Complex c0; // Default ctor
Complex c1 = Complex(1.0);
Complex c2 = Complex(1.0,2.0);

Complex c1a = 2.2;
Complex c1b(4.4);
Complex c2a(2.2, 3.3);
```

13

Return Value?

- ❖ Overload on return value?
 - One function returns "int"
 - Another (same name) returns "float"
 - Same signature (argument types)
- ❖ No!
 - Can't always tell from context

14

Access Functions

```
typedef unsigned int year;
class Student
{
    year grad_yr;
    ...
public:
    void setGradYear(year yr);
    year getGradYear();
    ...
};
```

15

Access Functions

```
void Student::
    setGradYear(year yr)
{
    grad_yr = yr;
}
year Student::getGradYear()
{
    return grad_yr;
}

s1.setGradYear(1997);
yr = s1.getGradYear();
```

16

Access Functions

```
typedef unsigned int year;
class Student
{
    year grad_yr;
    ...
public:
    void gradYear(year yr);
    year gradYear();
    ...
};
```

17

Access Functions

Overloaded function name

```
typedef unsigned int year;
class Student
{
    year grad_yr;
    ...
public:
    void gradYear(year yr);
    year gradYear();
    ...
};
```

18

CS481 - Object-Oriented Programming

Access Functions

```
void Student::
    gradYear(year yr)
{
    grad_yr = yr;
}
year Student::gradYear()
{
    return grad_yr;
}
```

```
s1.gradYear(1997);
yr = s1.gradYear();
```

19

Access Functions

What does this statement do?

```
s1.gradYear(s1.gradYear()+1);
```

20

Default Arguments

```
Complex::Complex(double real)
{
    re = real; im = 0.0;
}
Complex::Complex(double real,
                  double imag)
{
    re = real;
    im = imag;
}
```

These functions are pretty similar.

21

Default Arguments

```
// declaration - set default
class Complex
{...
    Complex(double real,
            double imag = 0.0);
}
```

```
// definition - no default here
Complex::Complex(double real,
                 double imag)
{
    re = real; im = imag;
}
```

22

Default Arguments

```
// declaration - set default
class Complex
{...
    Complex(double real,
            double imag = 0.0);
}
```

```
// definition - no default here
Complex::Complex (double real,
                  double imag)
{
    re = real;
}
```

Default more than one argument?

23

Default Arguments

- ❖ Defaulted args at end of list
- ❖ Do:
 - Use defaults for "common" value
- ❖ Don't:
 - Use default as a "flag" modifying behavior (use overload instead)

24

CS481 - Object-Oriented Programming

Default Constructor

- ❖ Constructor with no args
 - Or with all defaulted args
- ❖ Called on variable declaration
 - When variable not initialized
- ❖ Always must have one
 - Compiler supplies if none declared₂₅

One Complex C'tor

```
class Complex
{ double re, im;
public:
  Complex(double r = 0.0, double i = 0.0)
  { re = r; im = i; }
};
```

26

