15.3 Polymorphism

give multiple meanings to a func name
i.e. each child class has a
different function body
difference from redefining

can be used before it is defined
eg just from prototype

if function call in parent, uses
child body, not parent body

Late Binding

binding - tying prototype to body
late binding - bind at run-time

Virtual function - a function that
has late binding

use the same "command" for
different actions
uses the object type to determine
which body to run

Example, redefined vs virtual:

class Employee
{
    
    void print-check();
};

void Employee::print_check()
{
    printf("No salary info \n ");
}
class HourlyEmployee
{
  void print_check()
};

void HourlyEmployee::print_check()
{
  printf("Name: %s \n", get_name().c_str());
  printf("Worked: %0.f hours @ $ %0.2f \n", hours, rate);
  printf("Salary: $ %0.2f \n", hours * rate);
}

void print_a_check(Employee*e)
{
  e->print_check();
}

int main()
{
  HourlyEmployee sally;
  Employee *e;
  sally.set_hours(10);
  sally.set_rate(9.50);
  sally.print_check();
  e = &sally;
  print_a_check(e);
}
```cpp
print-a-check(e);
return 0;

class Employee
{
    virtual void print-check();
};
// same body as before

class HourlyEmployee
{
    virtual void print-check();
};
// same body as before

now the main would print the
same thing both times
C++ "waits" to get the body until
the function call is made
then uses body from type of
calling object

Rules
1) Must add keyword virtual to the
function prototype in the base
class definition
2) Only add virtual to prototype,
not to body
```

Limit use of virtual as it is more
Limit use of virtual as it is more costly & takes more time to do virtual functions.

Slicing Problem

C++ allows assignment from child to parent, but only parent vars are assigned.

Example:

```cpp
Employee tmp;
HourlyEmployee sally;

sally.set_name("Sally");
sally.set_hours(10);
sally.set_rate(9.50);

tmp = sally;
cout << tmp.get_rate(); // fails, tmp doesn't have
// rate or hours, only name
// & SSN
```

Pitfall - no body for virtual func.

If there is no body for a virtual function, it will issue cryptic compile error.

Tip - make destructors virtual

```cpp
HourlyEmployee *sally;
Employee *tmp;

sally = new HourlyEmployee;
tmp = sally;
delete tmp; // calls on sally
```
// Employee's destructor!!

if destructor is virtual, then Hourly Employee's destructor would be called instead