

11/5/04

CP+S 355

⑦ C++ Problem Areas - not to pick on C++ but to point out areas that may be especially problematic.

① C++ goals

- data abstraction and O-O features
- better static type checking
- compatibility w/ C: most C code should be legal C++.
- efficiency of compiled code.

②b

Casts and conversions

- safer than in C - i.e. there are safe casts
- ~~conversions change representation~~
- automatic conversions (coercions)
- interaction with overloading

Objects on stack

Object values vs object reference assignment necessarily has different semantics. (The value has to fit in the space available).

Care required to avoid creating dangling references to stack objects: stack object lifetime ends when AR goes away

Overloading

Interaction of dynamic lookup and static lookup w/ overloading.
Interaction with conversions/coercions.

Multiple inheritance

Implementation details necessarily leak into semantics

Constructor: initialize member data — like all methods
may be ~~be~~ overloaded.
method name \Rightarrow is class name.

Note constructor here is a class method ~ called
without ~~any~~ naming any particular instance.

C++ classes unlike \otimes python classes are not, themselves, objects.

Destructor:

Every object has a ~~default~~ destructor, either default or
programmer-provided.

Default: call destructor of each data member

destructor is an instance method — and needs to be made
virtual if class is used as a base class — so that
the right one is dynamically selected.

Visibility attributes of methods and data:

public — visible in any scope which can create or access
an object of this type

private — visible only in the class ~~itself~~ ^{itself}.

protected — visible in the class and in derived classes.

friend — explicit visibility of private data and methods for
some other class or function.

Virtual vs non-virtual (default)

only virtual methods can be overridden in derived classes.

example of the overloading problem:

a virtual member function is over-ridden in a derived class

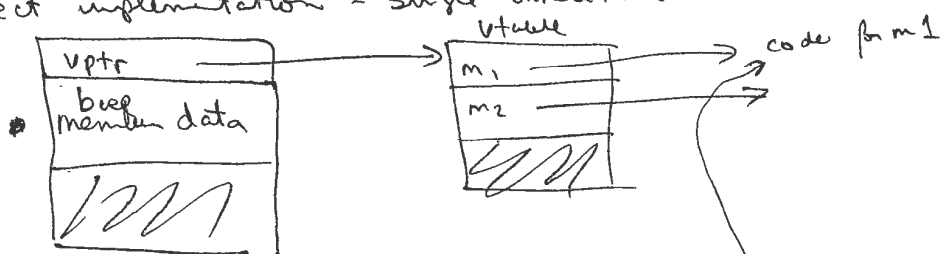
a non-virtual mf. is over-loaded in a derived class.

overriding implies dynamic selection

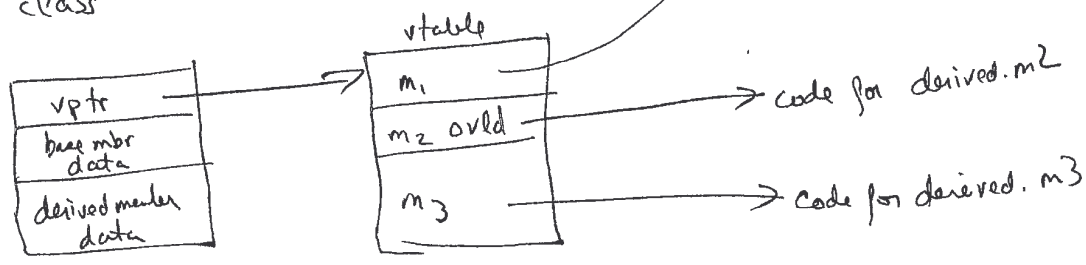
overloading implies static selection.

~~Virtual fun~~

Object implementation - single inheritance



~~derived class~~
derived class



Note non-virtual member functions not in vtable.

pure virtual member function: one that has ~~no~~ no implementation

abstract class: at least one p.v.f. : ~~can't~~ can't create an instance.

multiple inheritance in C++

in impl terms ~~can't~~ can create pointers and refs.
~~abstract~~ abstract class establishes layout of vtable.

next time