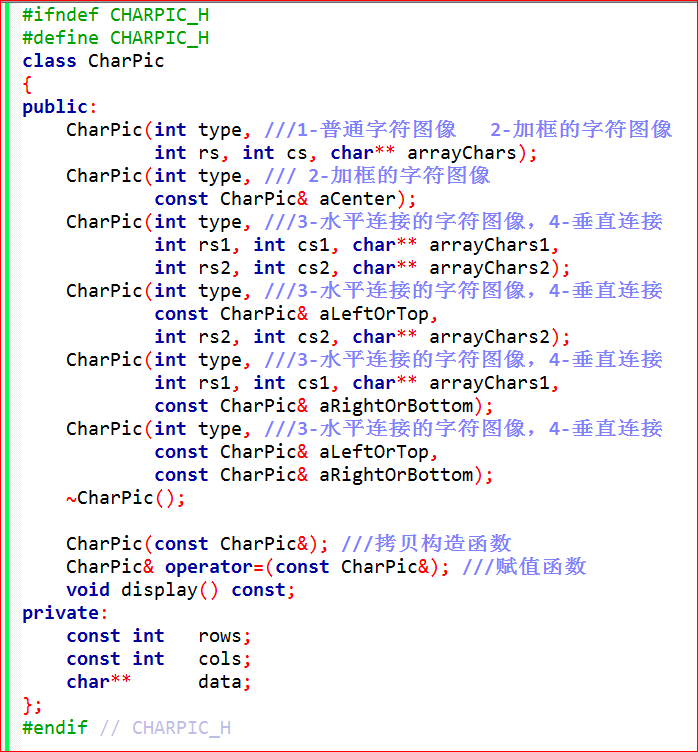
作业02 字符图像

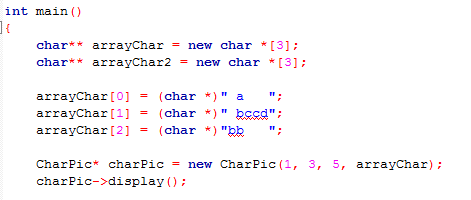
目标：开发字符图像的类。一个字符图像是由多行ANSI可打印字符组成的，各行字符数量可以不同；一个字符图像可以通过加框(Frame)形成一个新的字符图像；两个字符图像可以通过顶端对齐的方式经水平连接(HCat)形成一个新的字符图像；两个字符图像可以通过左端对齐的方式经垂直连接(VCat)形成一个新的字符图像；

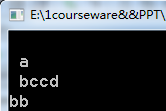
1. 现给出了一个字符图像的类定义，如下



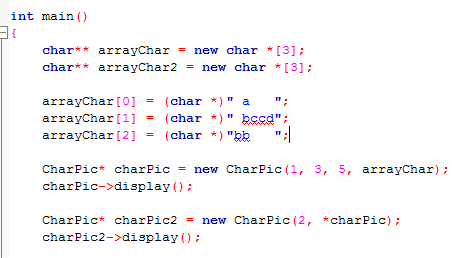
实现上边的CharPic类。

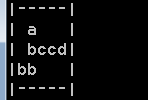
1. **构造函数 CharPic(int type, int rs, int cs, char\*\* arrayChars)：**



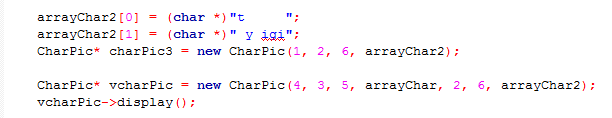


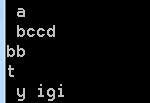
1. **构造函数 CharPic(int type, const CharPic& aCenter);**

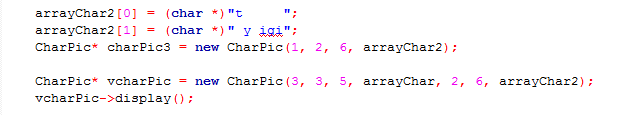


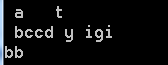


1. **构造函数 CharPic(int type, int rs1, int cs1, char\*\* arrayChars1, int rs2, int cs2, char\*\* arrayChars2);**

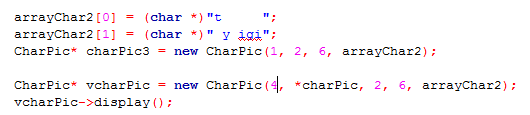


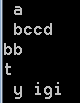


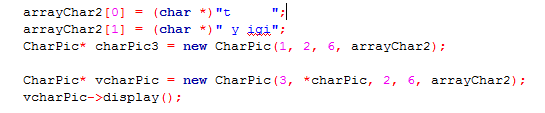


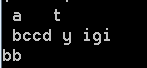


1. **构造函数 CharPic(int type, const CharPic& aLeftOrTop, int rs2, int cs2, char\*\* arrayChars2);**

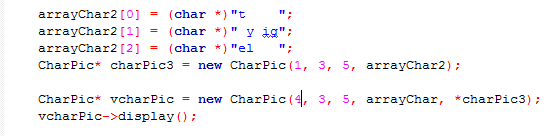


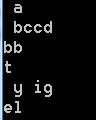


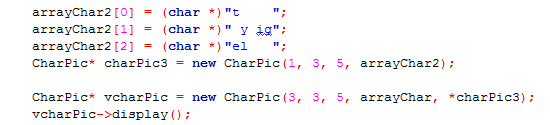


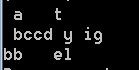


1. **构造函数 CharPic(int type, int rs1, int cs1, char\*\* arrayChars1, const CharPic& aRightOrBottom);**

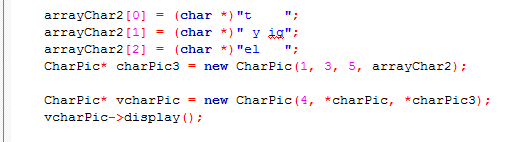


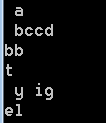


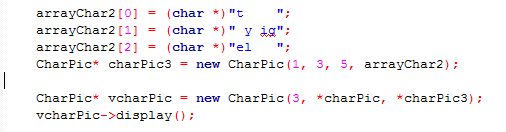


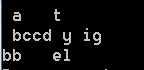


1. **构造函数 CharPic(int type, const CharPic& aLeftOrTop, const CharPic& aRightOrBottom);**

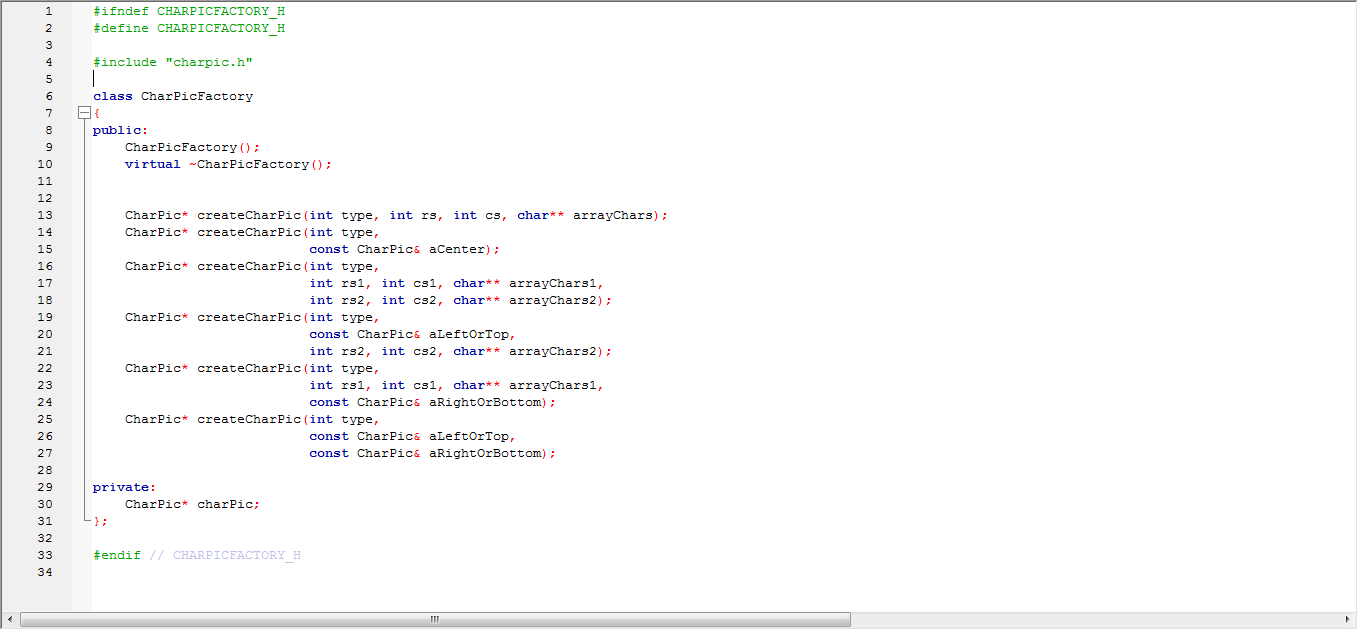


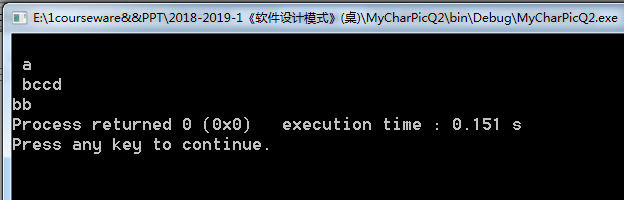






1. 上边CharPic中，通过重载构造函数创建字符图像，但构造函数的语义不是非常直观，使用也需要特别小心。请试一试将重载的构造函数改成(简单)工厂方法，使得语义更清晰。





1. 考察完成的代码，你会发现：
   1. 在构造、拷贝、赋值时，会将各子图都重新构造了一遍，拷贝和赋值使用的也是深拷贝和深赋值，这样在合成复杂的字符图像过程中，子图需要频繁地复制或赋值，效率不高；
   2. 在使用一个字符图像对象的过程中，不会改变字符图像中的字符，因此重新构建子图、深复制、深拷贝显得没有必要；
   3. 合成一个新的字符图像后，新图像不会含有关于”合成过程”的信息，即新图像不知道是通过水平连接得到的，还是垂直连接得到的，也不会知道“左子图”是什么等。

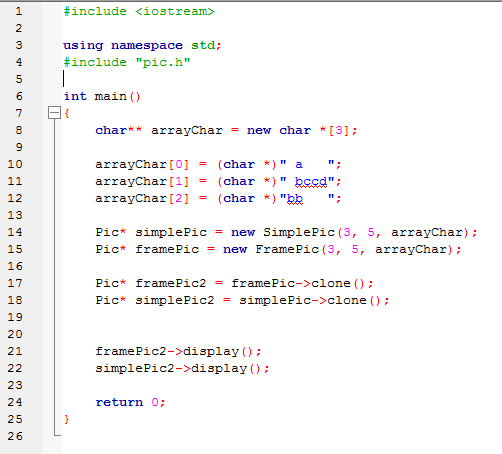
为此，重新设计了CharPic(为方便对比，类名改为Pic)，希望通过子类型化和**共享**来避免abc中的问题。类的结构图(部分)如下：

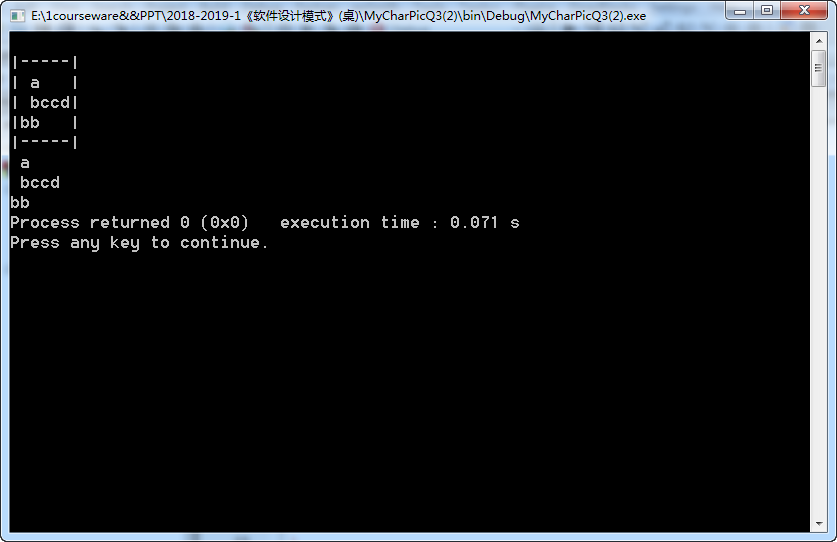


其中，由于子类型化的存在，字符图像的拷贝采用了原型方法模式 (即虚拟拷贝构造函数)；同时，禁止了图像对象间的赋值。

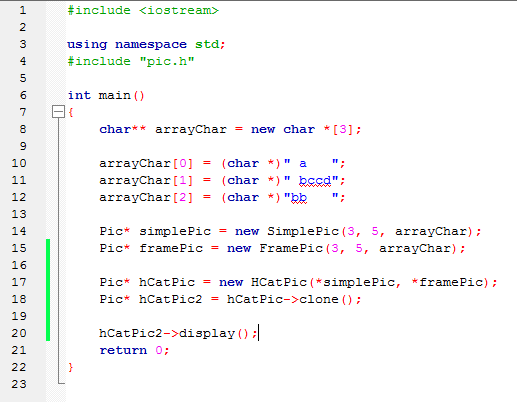
请定义并实现上述类图中的相关类，以及用于示例的相应主程序。

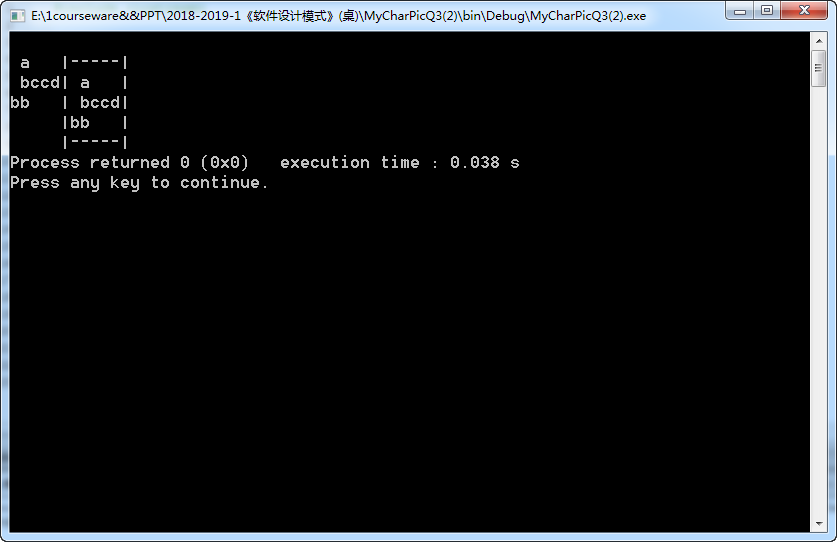
1. **克隆**
2. **SimplePic和FramePic的克隆**



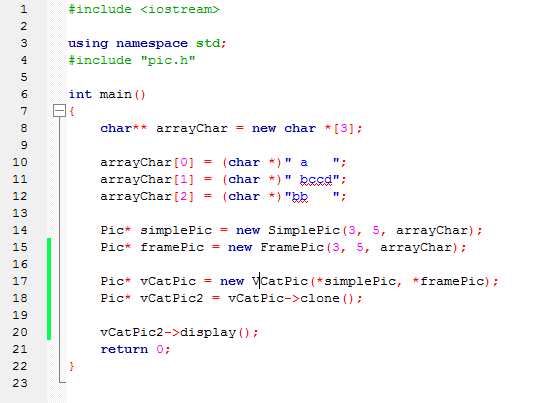


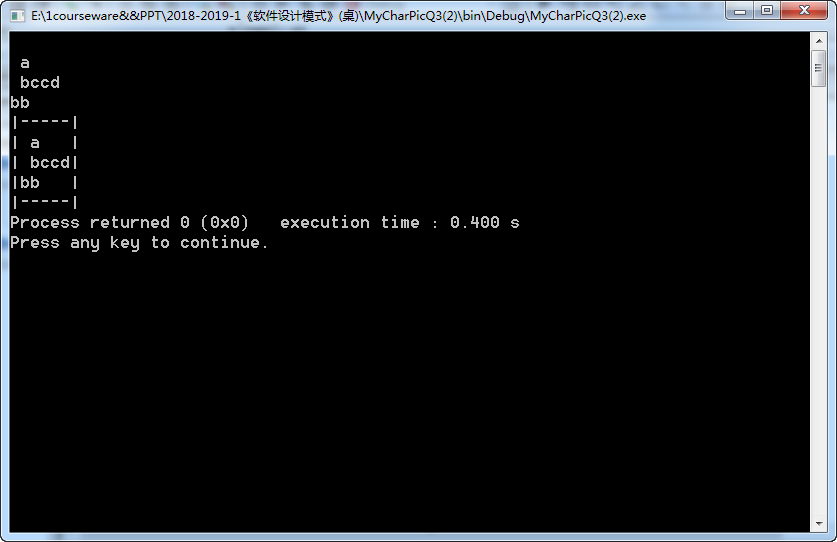
1. **HCatPic的克隆**





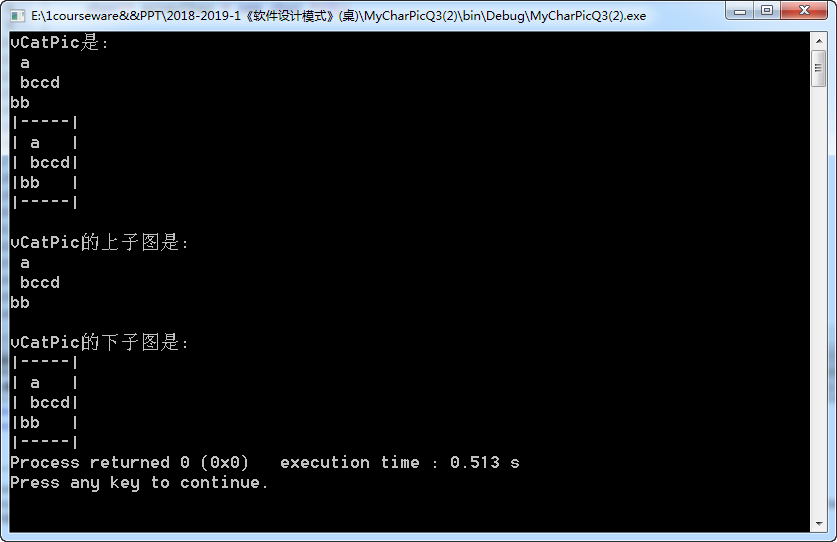
1. **VCatPic的克隆**





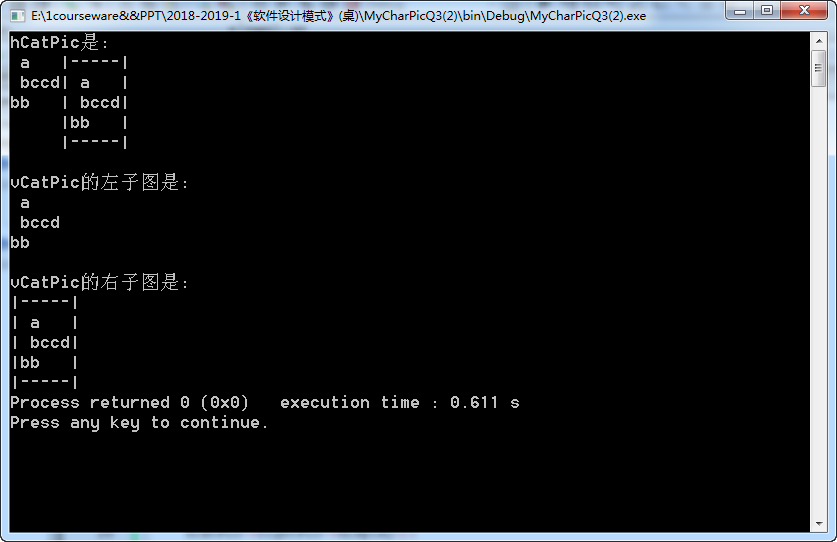
1. **保存子图**
2. **VCatPic 保存子图**





1. **HCatPic 保存子图**



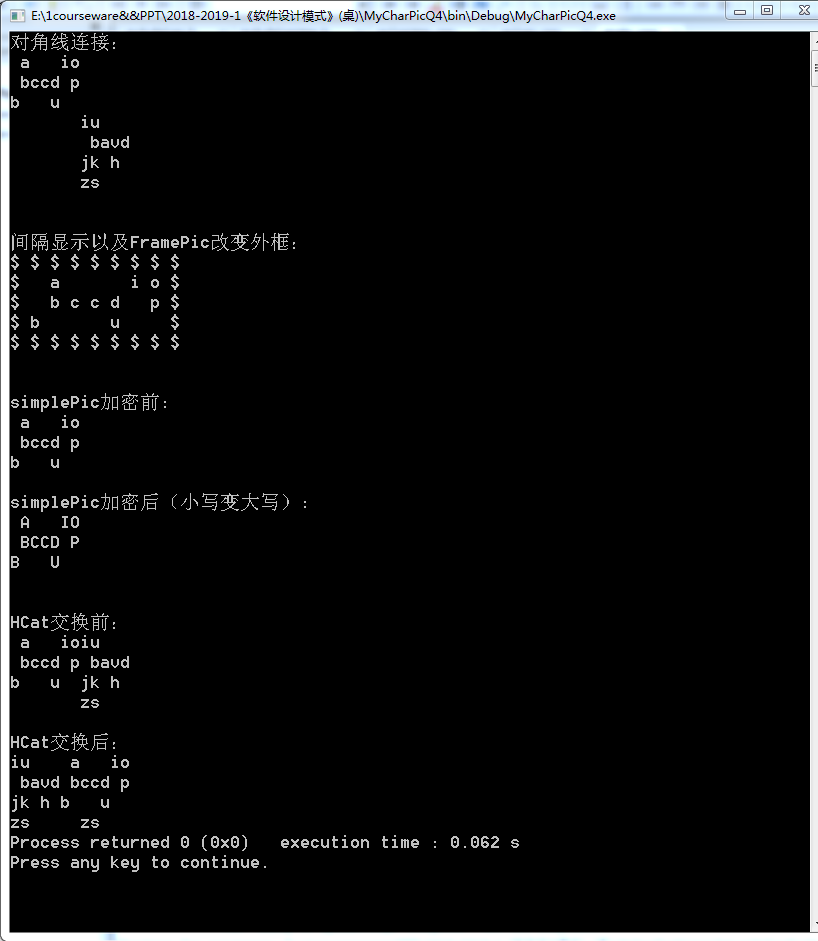


1. 若已知在复用3中的类时，可能还会增加有一些其它的需求，如：
   1. 增加对角线连接的字符图像--连接方式可变
   2. 显示时，可通过指定图像间隔(空白字符个数)显示字符图像；
   3. 变更FamePic中外框的字符（如变更为$号）
   4. 增加以加密的形式显示字符图像功能，如将字符图像中的A显示成Z，Z显示成Y。
   5. 增加图像的swap功能，即交换左右或上(SimplePic和FramePic的Swap为空即可)

请继续对3中的类，使用桥接模式重新设计，部分类图如下;同时指出如何适应a-e的变化需求.







1. 现有Menu类及相关类，如下图



在显示AppleMenu和WindowMenu时，是以图片的形式显示的;对于ConsoleMenu当然也以图片形式显示更一致些，但若完全实现该功能，可能会比较繁琐，可以考虑复用已有的字符图像类。为此，请分别以类适配器模式和对象适配器模式，复用字符图像类，实现ConsoleMenu的show功能。

