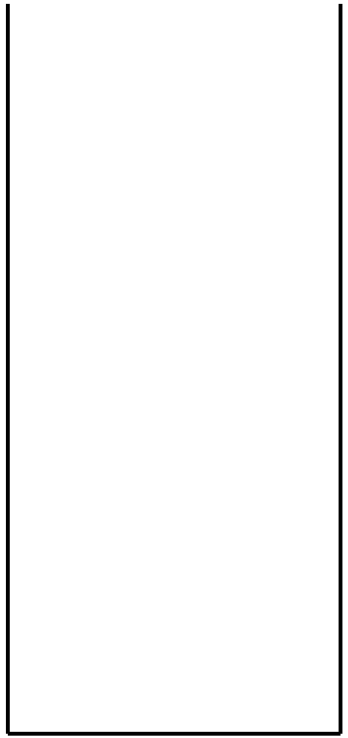
line 16:

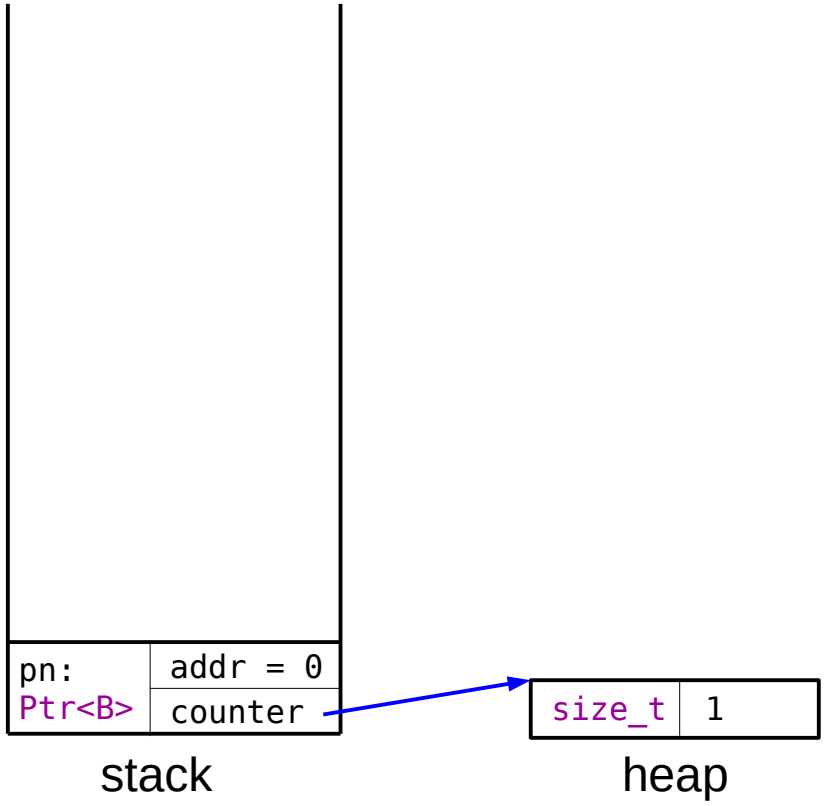


stack

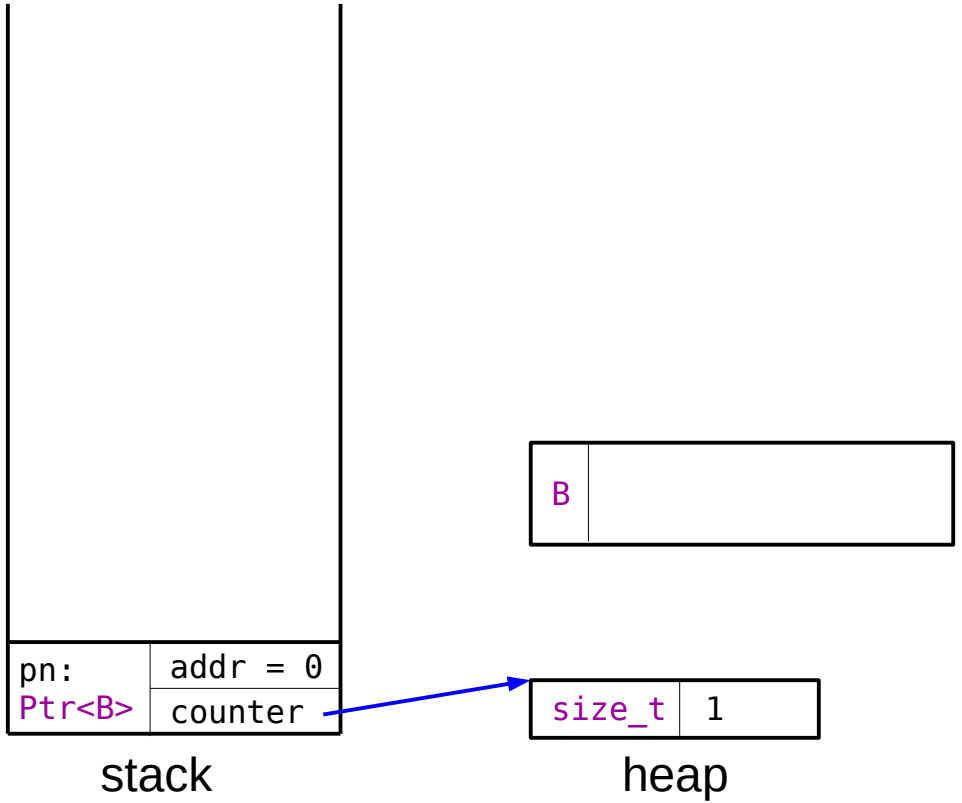
heap

line 17: Ptr<B> pn = 0;

calls: Ptr<B>(B*)

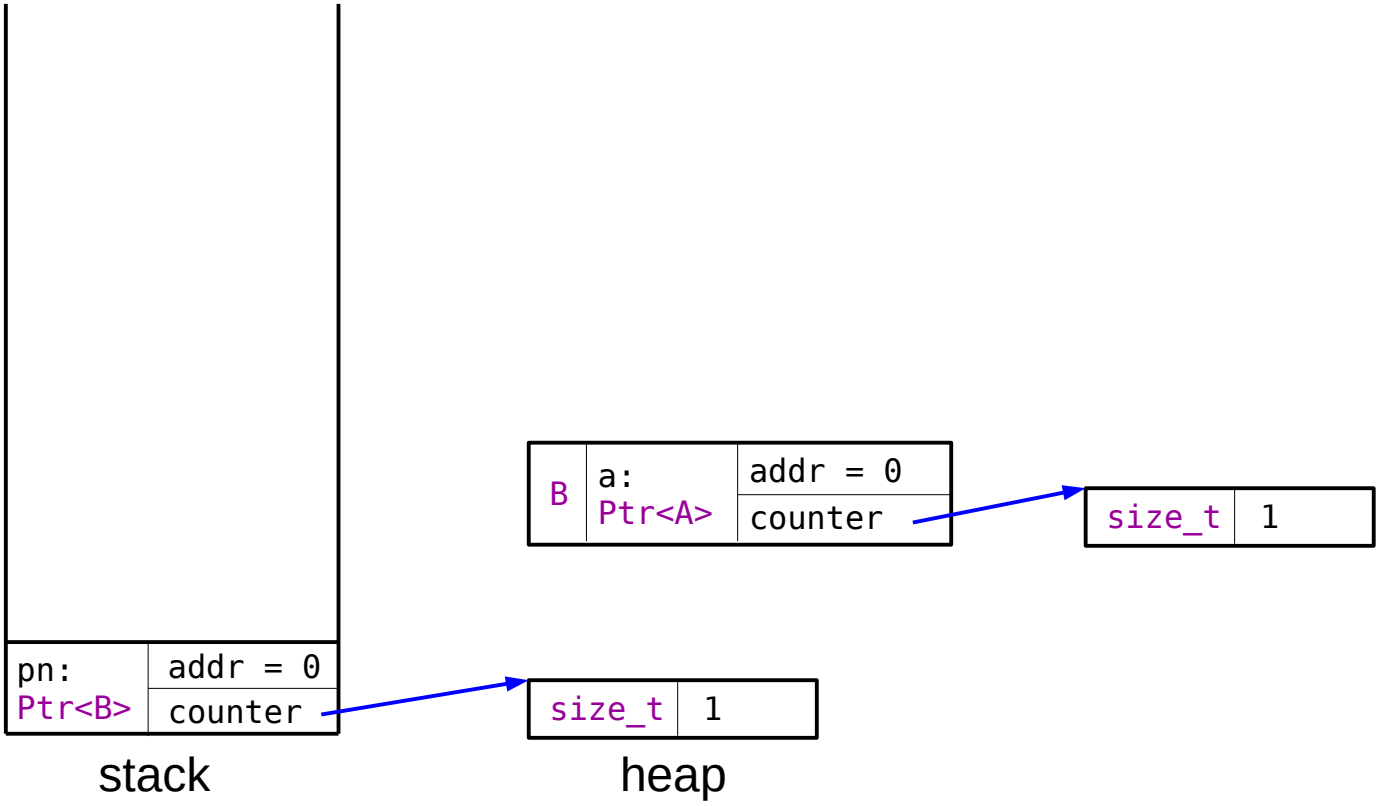


line 19: Ptr<B> pb = new B(0); calls: B(A*)



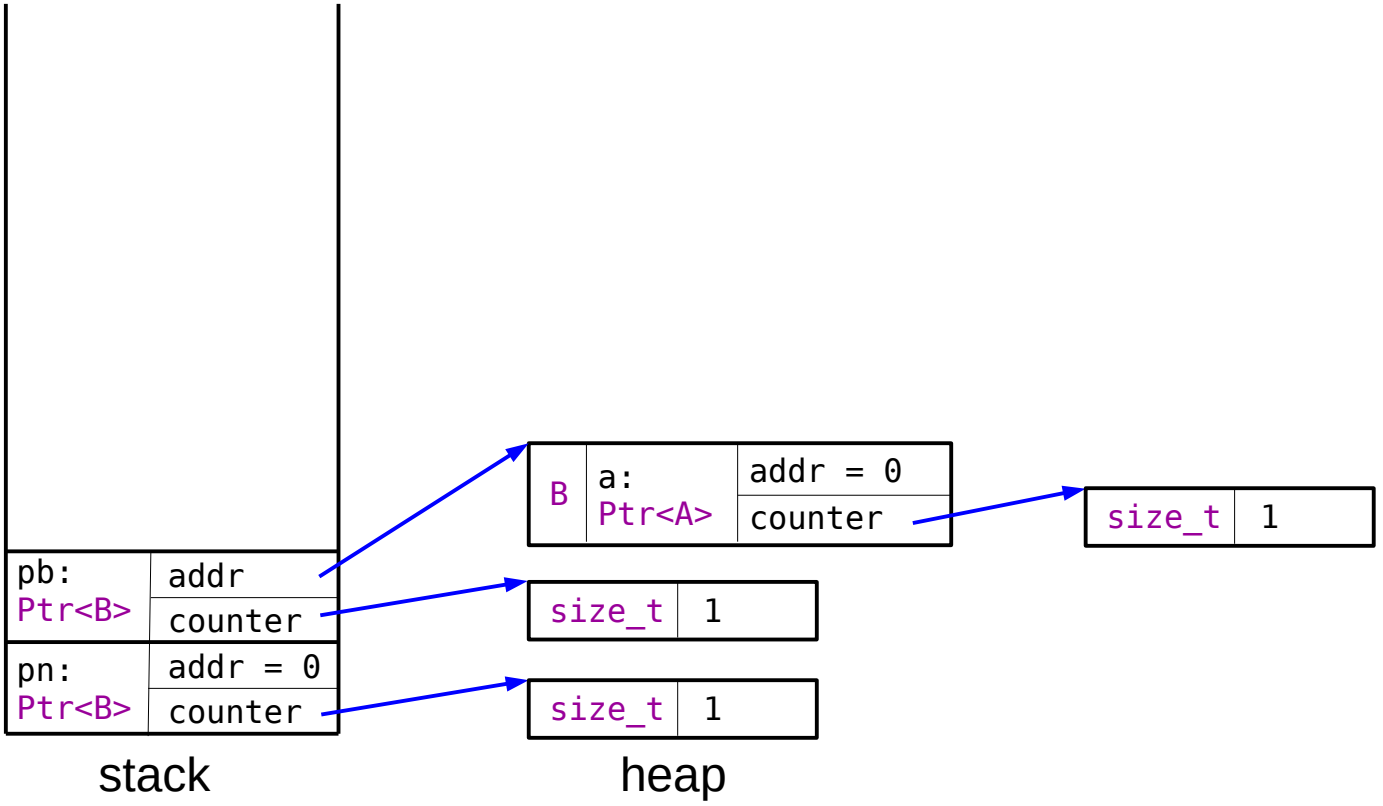
line 19: Ptr<B> pb = new B(0);

calls: B(A*)
Ptr<A>(A*)

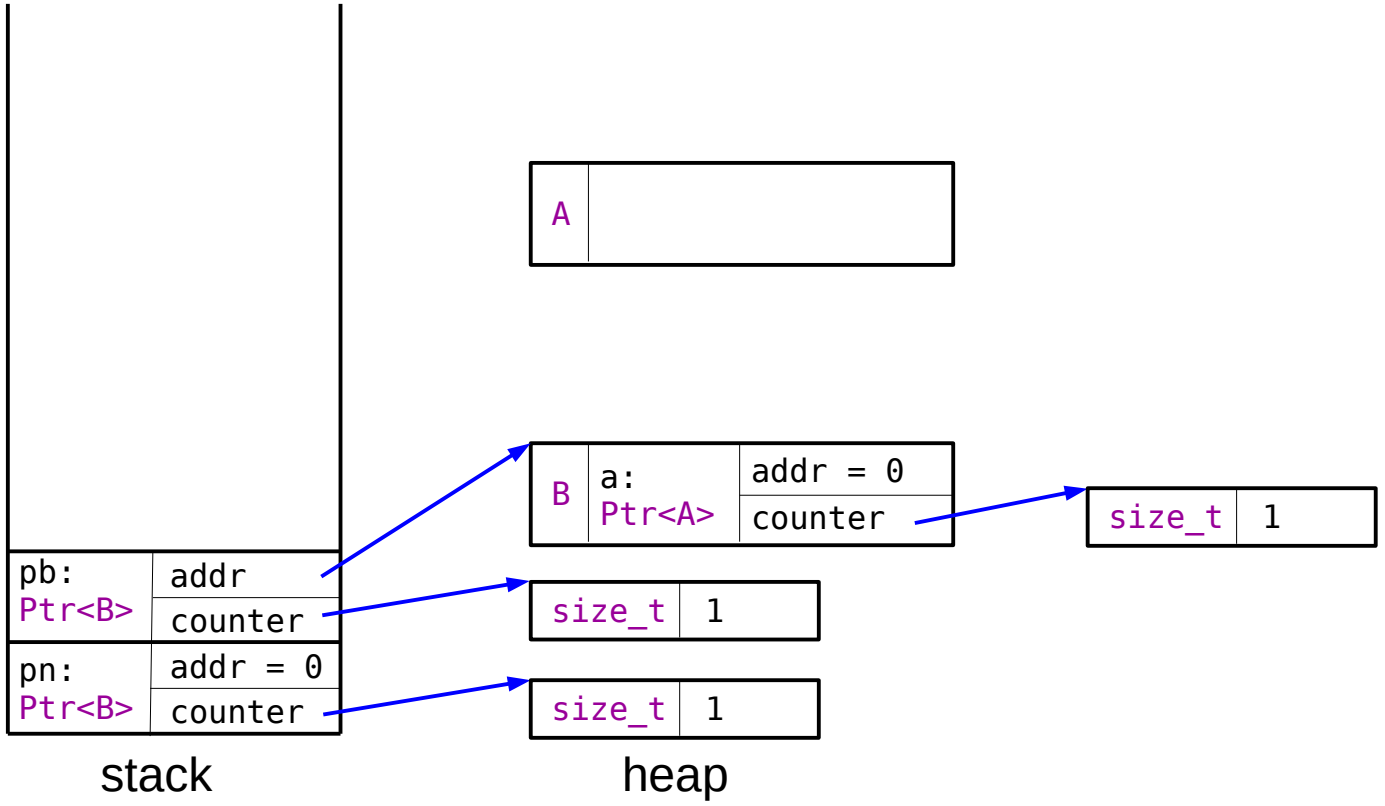


```
line 19: Ptr<B> pb = new B(0);
```

```
calls: B(A*)
       Ptr<A>(A*)
       Ptr<B>(B*)
```

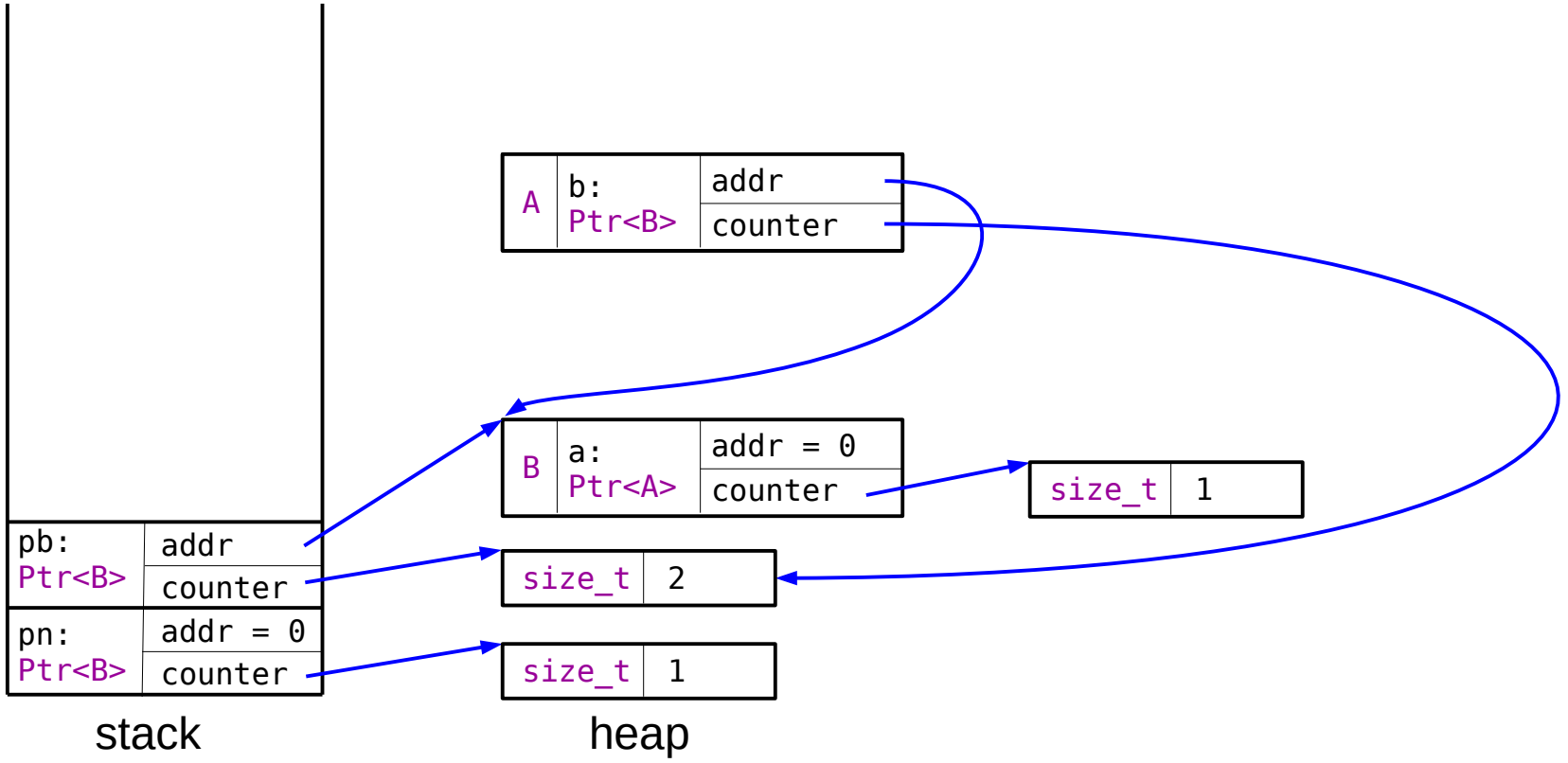


line 21: `Ptr<B> pa = new A(pb);` calls: `A(Ptr<B>&)`



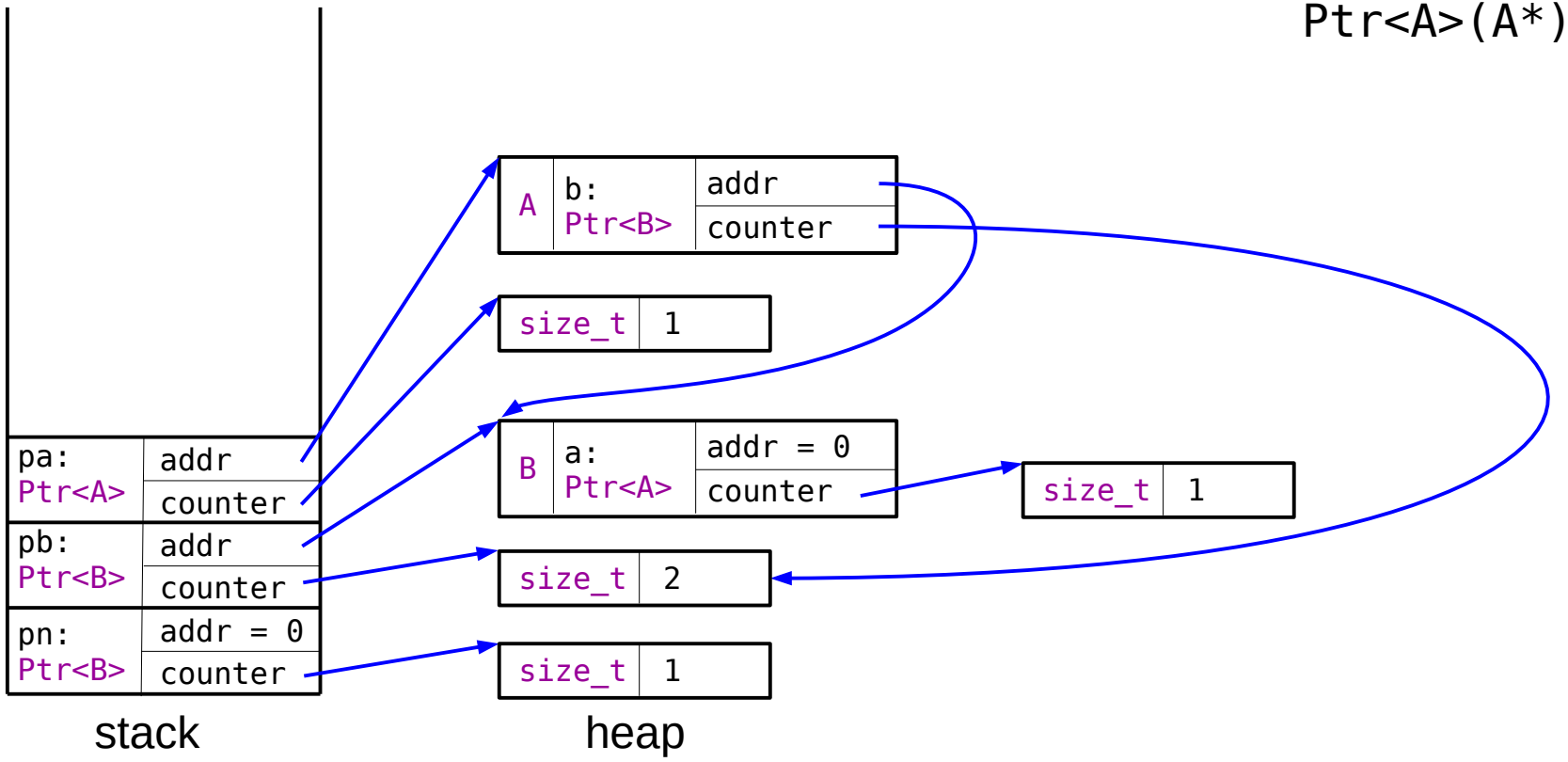
line 21: `Ptr<B> pa = new A(pb);`

calls: `A(Ptr<B>&)`
`Ptr<B>(const Ptr<B>&)`



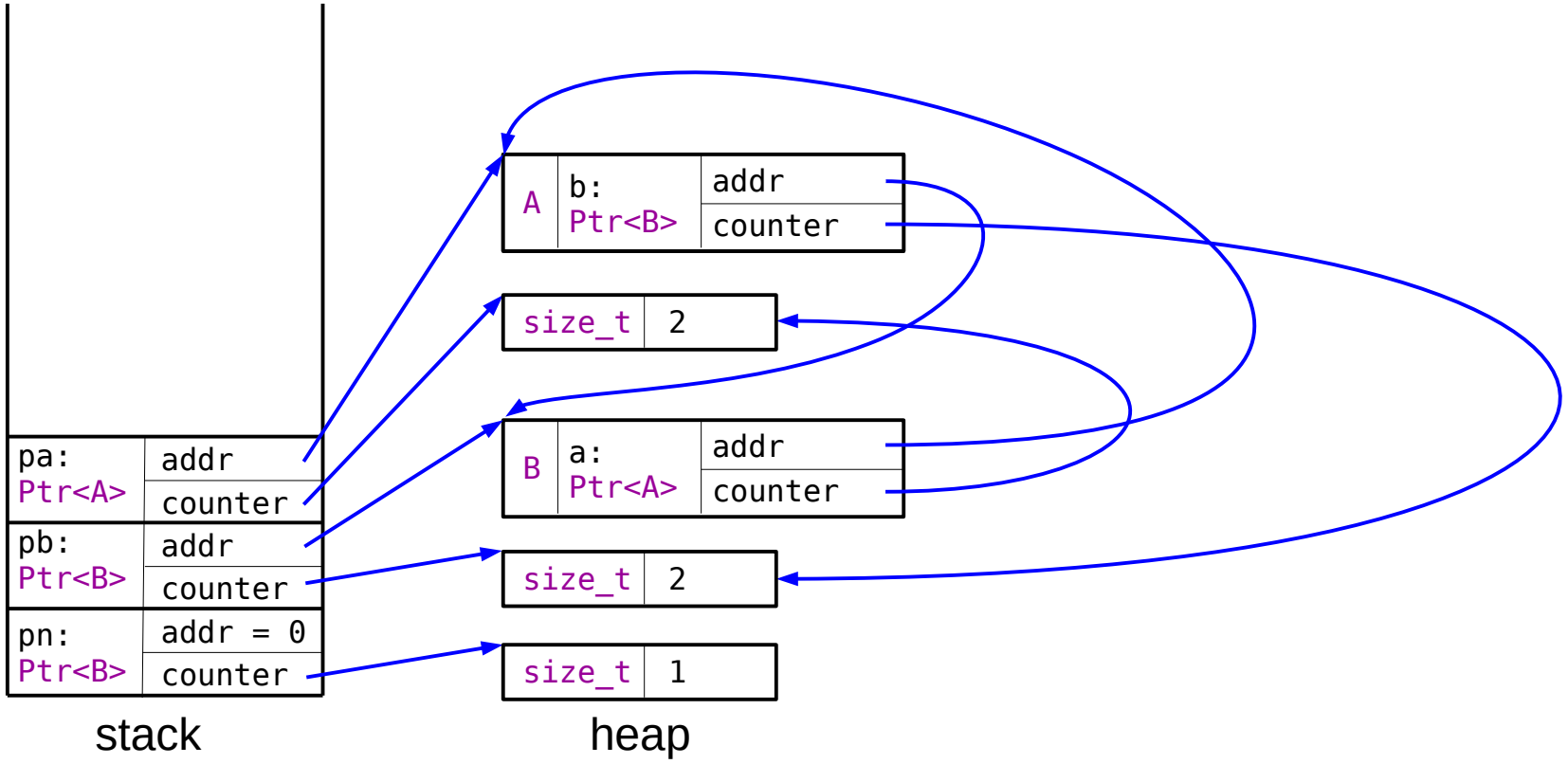
```
line 21: Ptr<B> pa = new A(pb);
```

calls: A(Ptr<B>&)
Ptr<B>(const Ptr<B>&)
Ptr<A>(A*)



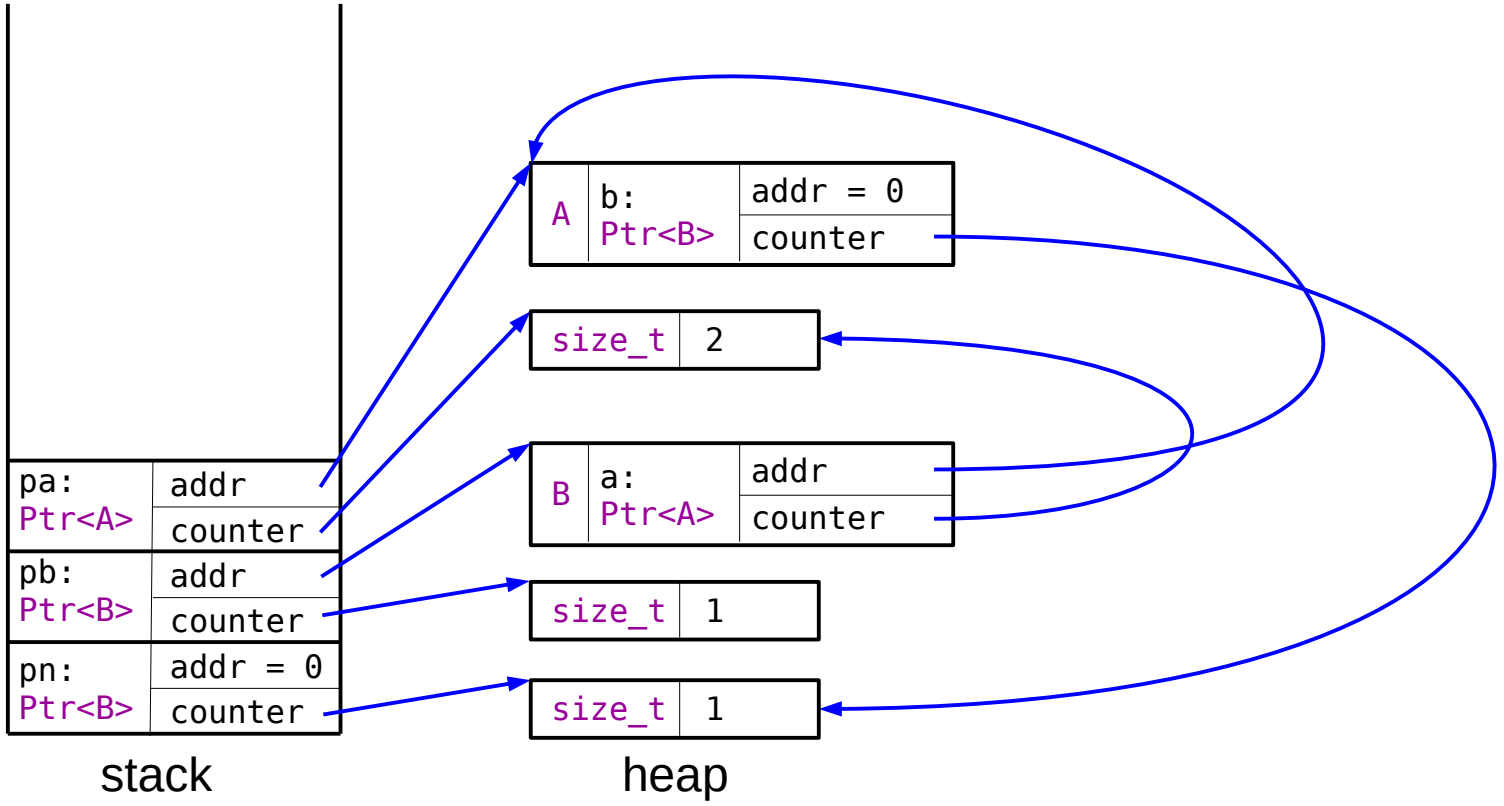
line 23: pb->a = pa;

calls: Ptr<B>::operator->()
Ptr<A>::operator=(..)



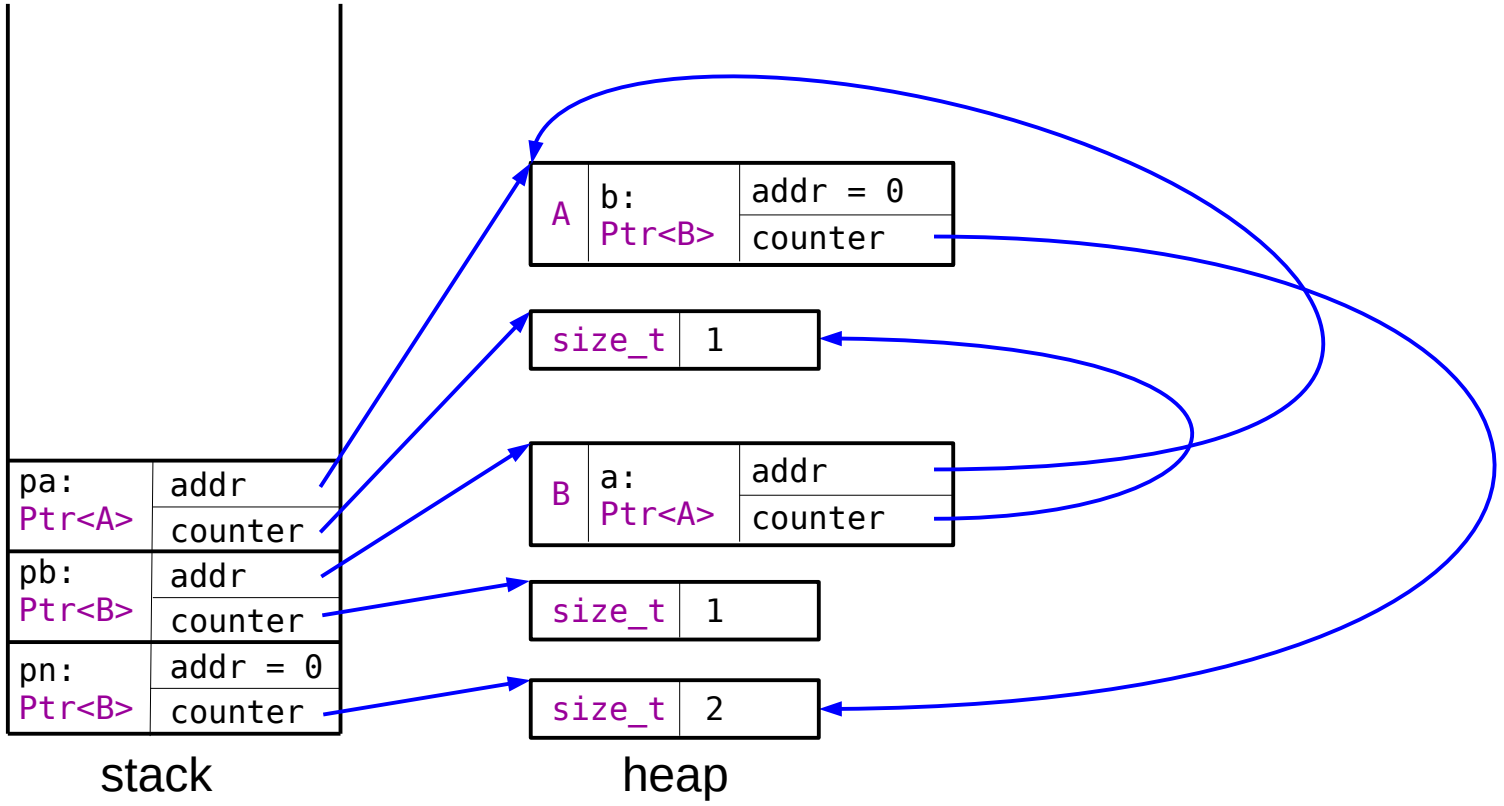
line 25: pa->b = pn;

calls: Ptr<A>::operator->()
Ptr<B>::operator=(..)



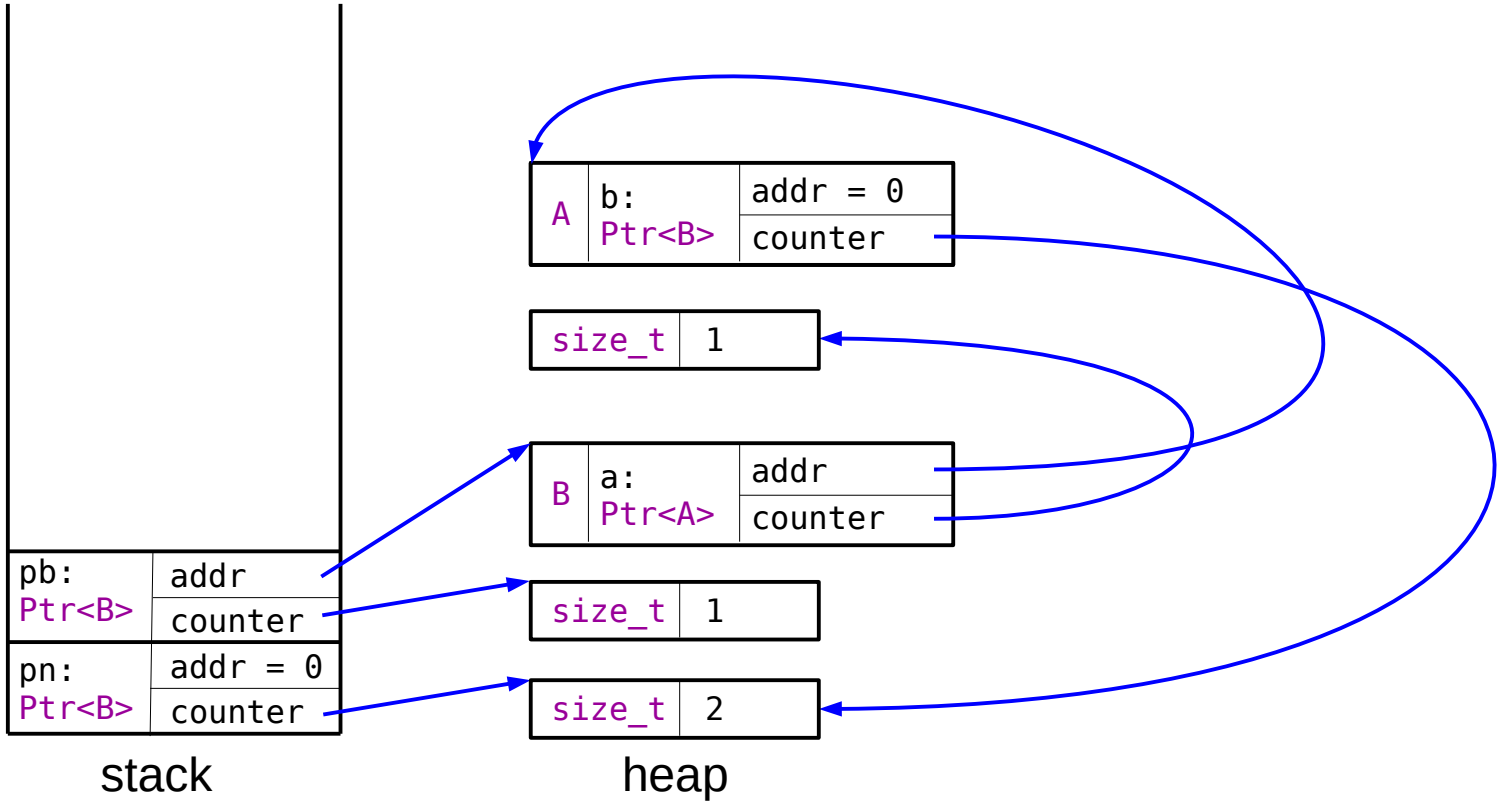
line 27: return 0;

calls: ~Ptr<A>



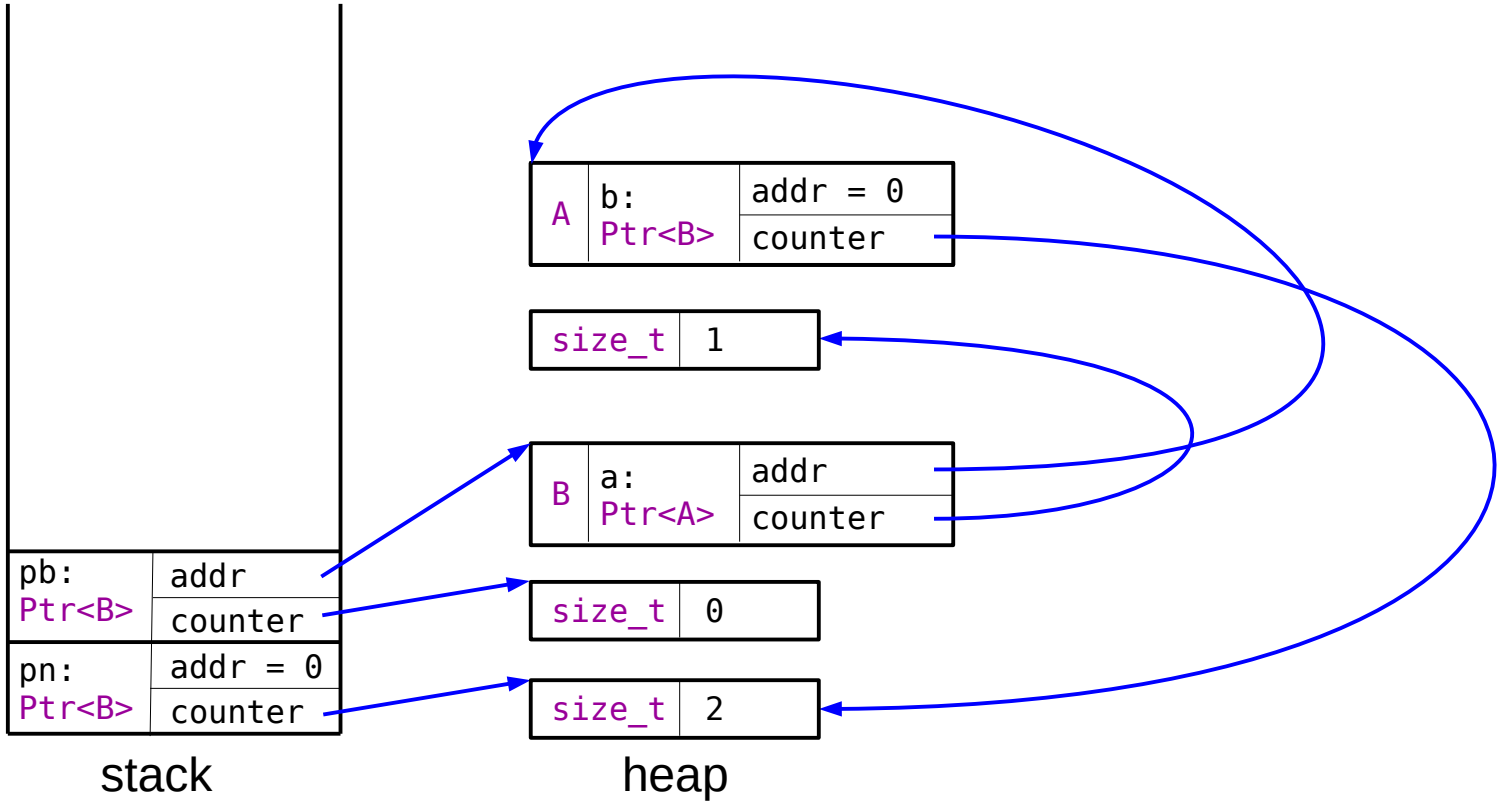
line 27: return 0;

calls: ~Ptr<A>



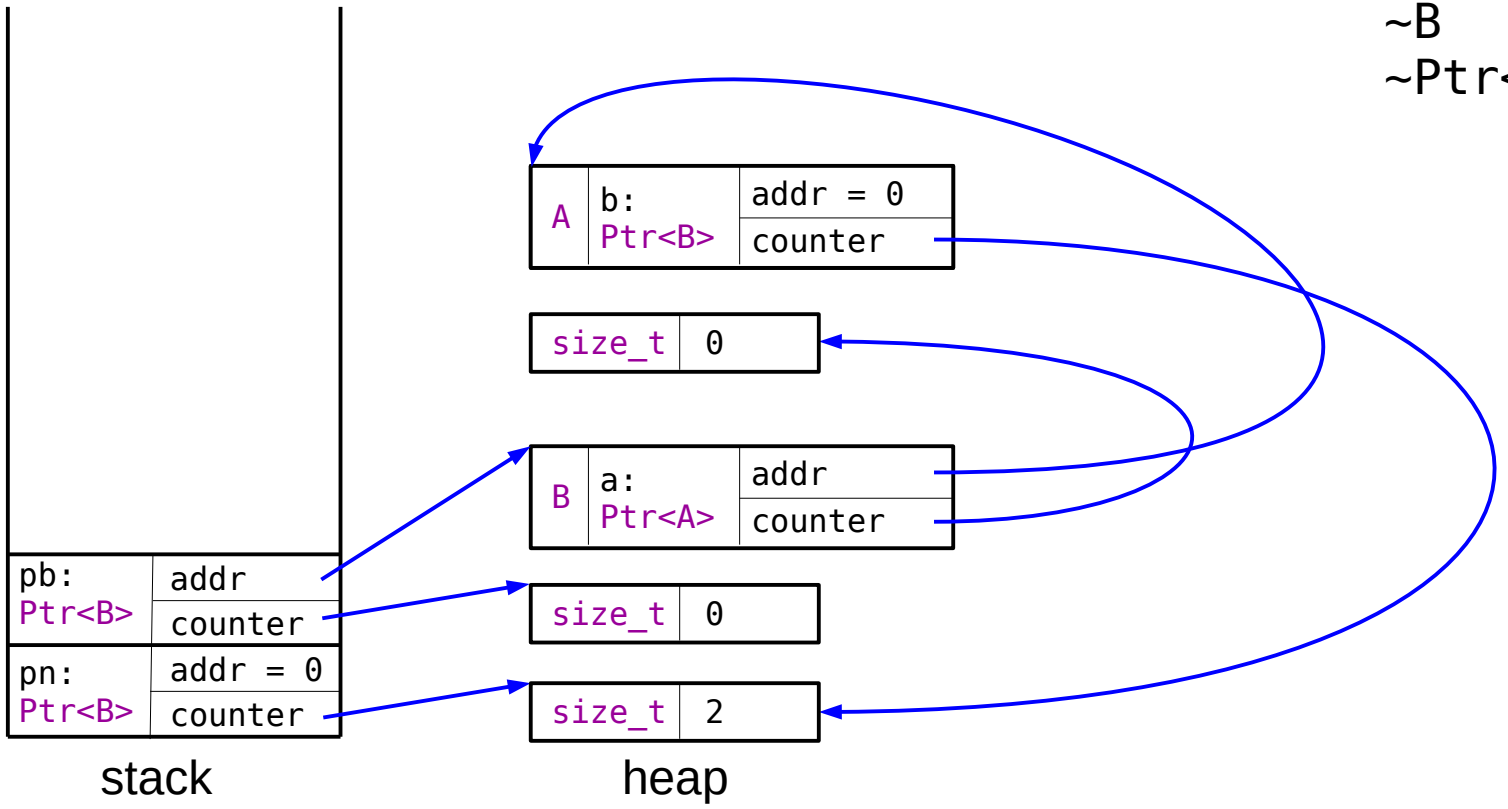
line 27: return 0;

calls: ~Ptr<A>
~Ptr<B>



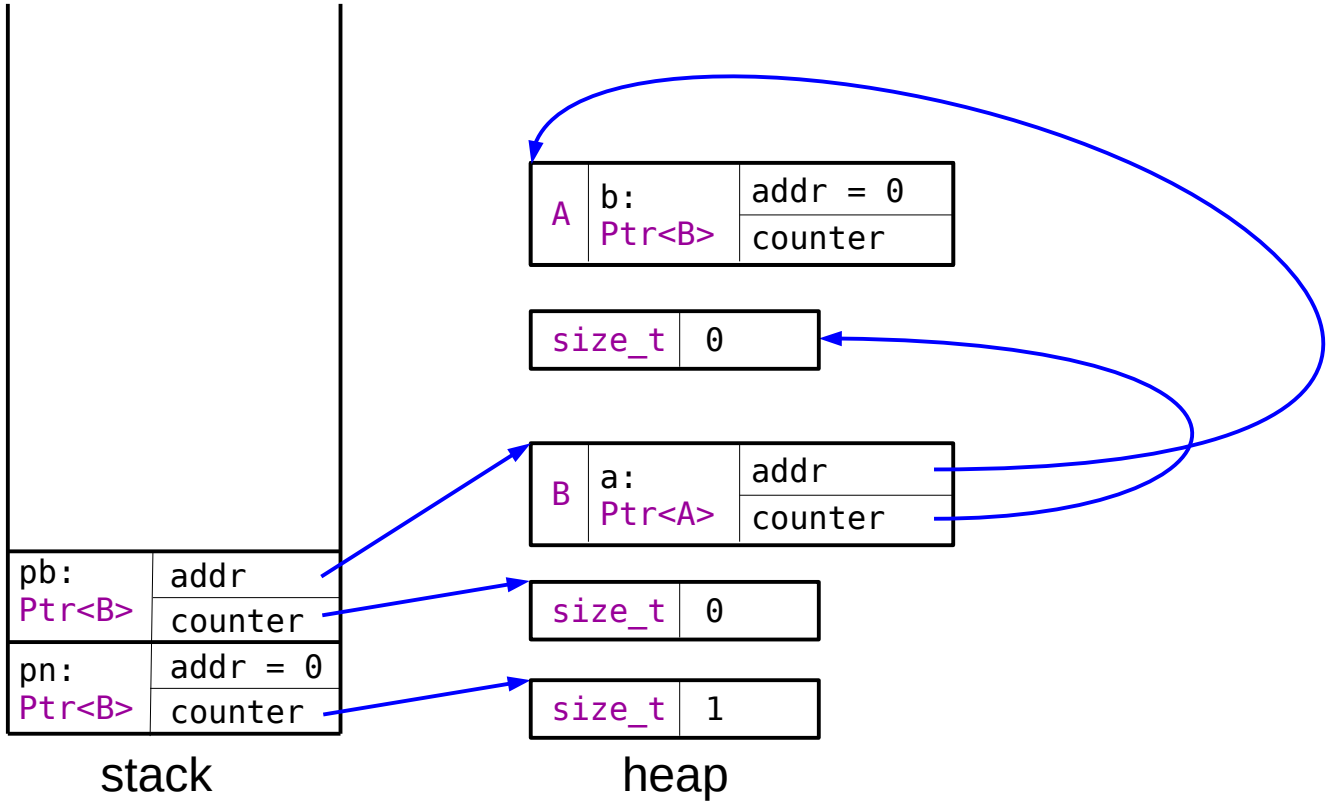
line 27: return 0;

calls: ~Ptr<A>
~Ptr<B>
~B
~Ptr<A>



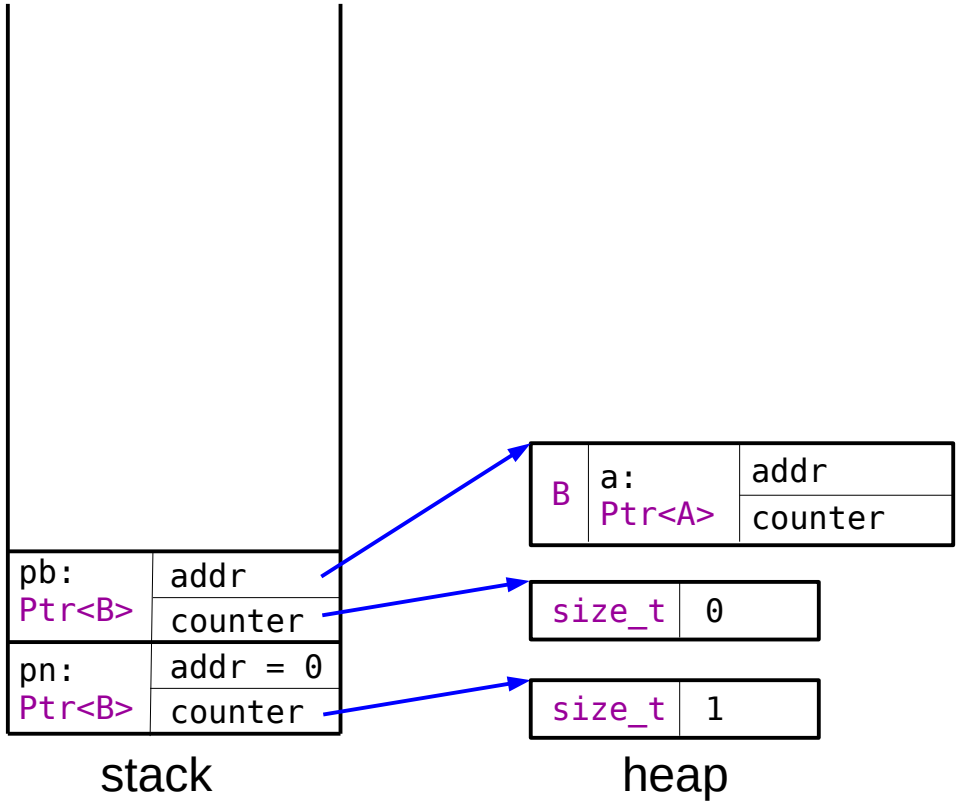
line 27: return 0;

calls: ~Ptr<A>
~Ptr<B>
~B
~Ptr<A>
~A
~Ptr<B>



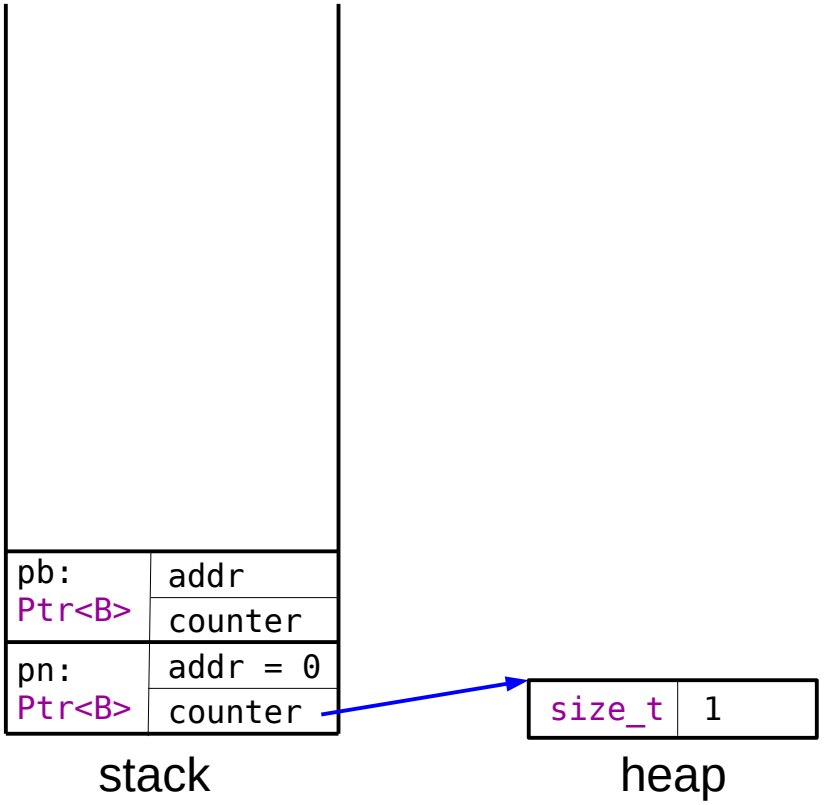
line 27: return 0;

calls: ~Ptr<A>
~Ptr<B>
~B
~Ptr<A>
~A
~Ptr<B>



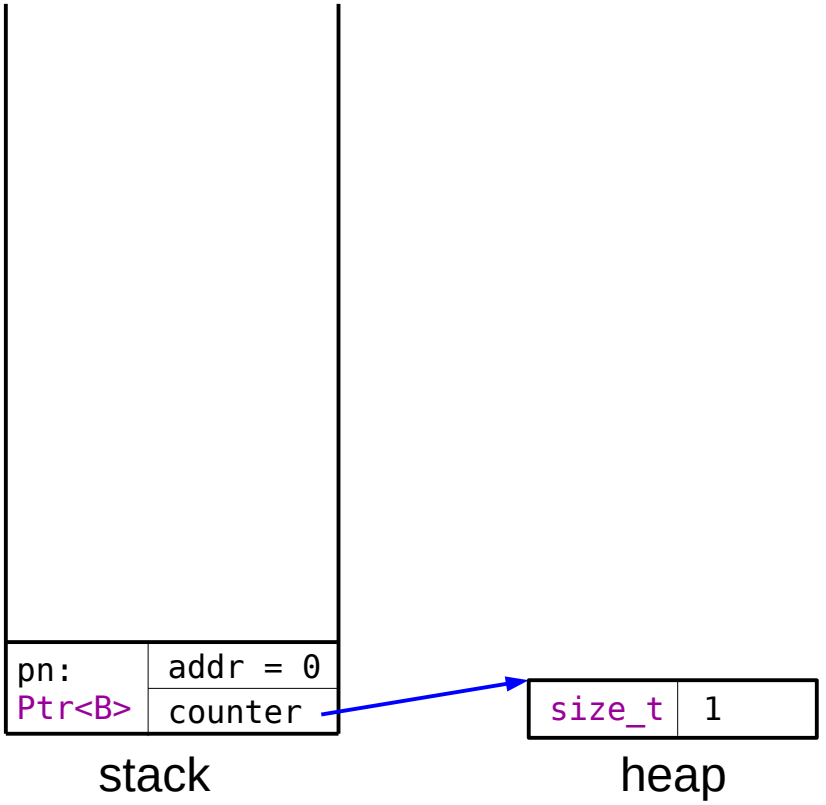
line 27: return 0;

calls: ~Ptr<A>
~Ptr<B>
~B
~Ptr<A>
~A
~Ptr<B>



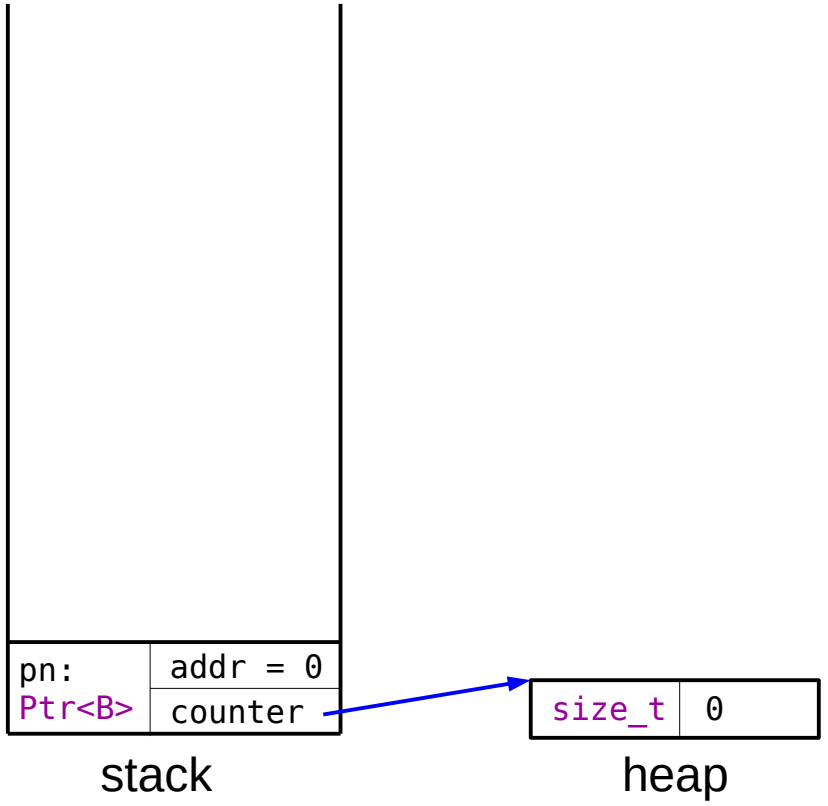
line 27: return 0;

calls: ~Ptr<A>
~Ptr<B>
~B
~Ptr<A>
~A
~Ptr<B>



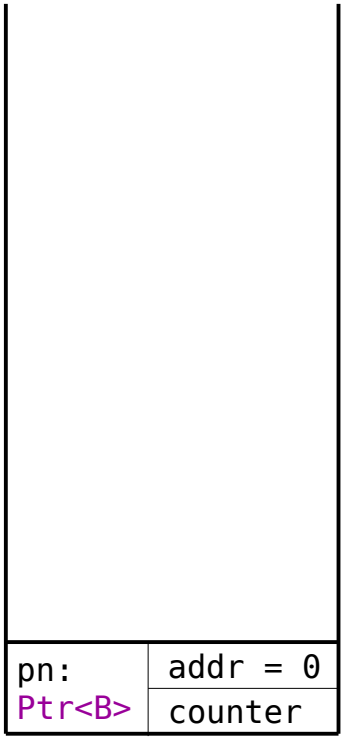
line 27: return 0;

calls: ~Ptr<B>



line 27: return 0;

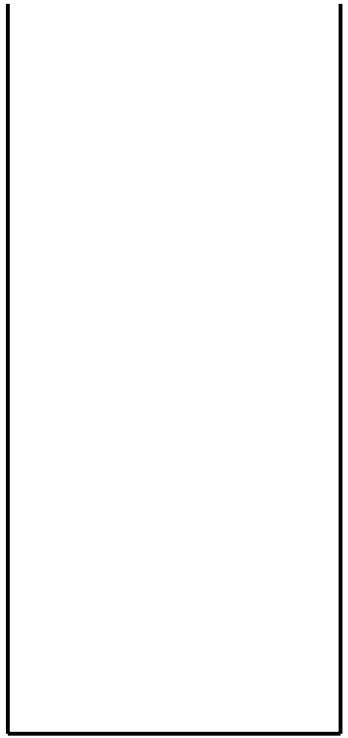
calls: ~Ptr<B>



stack

heap

line 28:



stack

heap