Pointers to Structs

/run/media/lijunjie/资料/Learning Video/pointers-arrays-recursion/01_pointers/04_pointers-to-sophisticated-types/01_pointers-to-structs_instructions.html

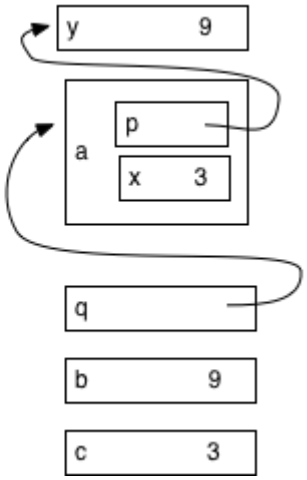
In this figure note that *a.b* dereferences the **b** field inside of struct *a* . Dereferencing *q* , then selecting field *x* requires either parenthesis, or the -> operator.

```
struct aStruct {
   int * p;
   int x;
};

int y = 9;
struct aStruct a;
struct aStruct * q = &a;

a.p = &y;
a.x = 3;

int b = *a.p;
int c = (*q).x;   //better: q->x
```



When we have pointers to structs, we *can* just use the * and . operators that we have seen so far, however, the order of operations means that . happens first. If we write * *a.b* , it means *(*a.b*)—a should be a struct, which we look inside to find a field named *b* (which should be a pointer), and we dereference that pointer. If we have a pointer to a struct (c, and we want to refer to the field d in the struct at the end of the arrow, we would need parenthesis, and write (* *c*). *d* (or the -> operator we will learn about momentarily).

In the figure above, we have a **struct** which has a field *p* (which is a pointer to an int), and a field *x* which is an int. We then declare *y* (an int), *a* (a struct), and *q* (a pointer to a struct), and initialize them. When we write **a.p* , the order of operations is to evaluate *a.p* (which is an arrow pointing at *y*), then dereference that arrow. If we wrote **q.x,* we would receive a compiler error, as *q* is not a struct, and the order of operations would say to do *q.x* first (which is not possible, since *q* is not a struct). We could write parenthesis, as in the figure ((**q*) *.x*).

However, pointers to structs are incredibly common, and the above syntax gets quite cumbersome, especially with pointers to structs which have pointers to structs, and so on. For (**q*) *.x,* it may not be so bad, but if we have *(*(*(*q).r).s).t* it becomes incredibly ugly, and confusing. Instead, we should use the -> operator, which is shorthand for dereferencing a pointer to a struct and selecting a field—that is, we could write *q->x* (which means exactly the same thing as (**q*) *.x*). For our more complex example, we could instead write *q->r->s->t* (which is easier to read and modify).