6.12 — Member selection with pointers and references

BY ALEX ON JULY 17TH, 2007 | LAST MODIFIED BY ALEX ON APRIL 7TH, 2016

It is common to have either a pointer or a reference to a struct (or class). As you learned previously, you can select the member of a struct using the **member selection operator (.)**:

```
struct Person
{
    int age;
    double weight;
};
Person person;

// Member selection using actual struct variable
person.age = 5;
```

This syntax also works for references:

```
struct Person
{
    int age;
    double weight;
};
Person person; // define a person

// Member selection using reference to struct
Person &ref = person;
ref.age = 5;
```

However, with a pointer, you need to dereference the pointer first:

```
struct Person
{
    int age;
    double weight;
};
Person person;

// Member selection using pointer to struct
Person *ptr = &person;
(*ptr).age= 5;
```

Note that the pointer dereference must be enclosed in parenthesis, because the member selection operator has a higher precedence than the dereference operator.

Because the syntax for access to structs and class members through a pointer is awkward, C++ offers a second member selection operator (->) for doing member selection from pointers. The following two lines are equivalent:

```
(*ptr).age  = 5;
ptr->age = 5;
```

This is not only easier to type, but is also much less prone to error because the dereference is implicitly done for you, so there are no precedence issues to worry about. Consequently, when doing member access through a pointer, always use the -> operator instead of the . operator.

Rule: When using a pointer to access the value of a member, use operator-> instead of operator. (the . operator)

6.12a -- For-each loops