10.18 — Member selection with pointers and references

ALEX JULY 11, 2021

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 (.):

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

Person person{};

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

This syntax also works for references:

```cpp
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 use the arrow operator:

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

// Member selection using pointer to struct
Person* ptr{ &person };
ptr->age = 5;
```

The arrow operator does the same as an indirection followed by the . member selection operator

```cpp
(*ptr).age = 5;
// same as
ptr->age = 5;
```

Note that indirection through the pointer must be enclosed in parentheses, because the member selection operator has a higher precedence than the indirection operator.

The arrow operator is not only easier to type, but is also much less prone to error because the indirection 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.

Best practice

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

The member selection operator is always applied to the currently selected variable. If you have a mix of pointer- and normal member variables, you can see member selections where . and -> are mixed.

```cpp
#include <iostream>
#include <string>

struct Paw
{
    int claws{};
};

struct Animal
{
    std::string name{};
    Paw paw{};
};

int main()
{
    Animal puma{ "Puma", { 5 } };

    Animal* pointy{ &puma };

    // pointy is a pointer, use ->
    // paw is not a pointer, use .
    std::cout << pointy->paw.claws << '\n';

    return 0;
}
```

Next lesson
10.19 For-each loops

Back to table of contents

Previous lesson
10.17 References and const

X

X