Why can't I access and use a member type aliases from an object?

Asked 1 year, 1 month ago Modified 1 year, 1 month ago Viewed 66 times

0

Is it impossible to access a type alias (or `typedef`) of a templated class, from an object of that class? For example, why isn't the following possible:

```
template <typename TKey, typename TData>
class MyClass
{
public:
    using key_t = TKey;
    using data_t = TData;
    TKey key;
    TData data;
    MyClass(TKey _key, TData _data) : key(_key), data(_data) {  }
};

int main() {
    MyClass<int, float> mc{1, 1.0f};
    using DataType = typename mc.data_t; //error: expected ';' after alias declaration
    mc.data_t newData = 2.0f; //error: Cannot refer to type member in
'MyClass<int,float> with '.'

    return 0;
}
```

Is there another way to do something like this?

c++ c++17 type-alias

Share Edit Follow Flag

edited Apr 28, 2021 at 13:32
Nicol Bolas
413k ● 61 ● 709 ● 903

asked Apr 28, 2021 at 12:02
Verwirrt
366 ● 1 ● 12

`using DataType = MyClass<int, float>::data_t;` works for the first line. – Jeffrey Apr 28, 2021 at 12:07

1 Answer

Sorted by: Highest score (default)

6

You got the syntax wrong. Change it to:

```
int main() {
    MyClass<int, float> mc{1, 1.0f};
    using DataType = typename MyClass<int,float>::data_t;
    MyClass<int,float>::data_t newData = 2.0f;
    return 0;
}
```

If you want to get the aliases from `mc` directly, you can `using DataType = decltype(mc)::data_t;`. You can actually avoid any repetition of `MyClass<int,float>` :

```
int main() {
    MyClass<int, float> mc{1, 1.0f};

    // suppose we cannot or do not want to spell out MyClass<int,float> again...
    using my_class_t = decltype(mc);
    using data_t = my_class_t::data_t;
    data_t newData = 2.0f;
}
```

Share Edit Follow Flag

edited Apr 28, 2021 at 12:22

answered Apr 28, 2021 at 12:06
463035818_is_not_a_number
88.3k ● 9 ● 75 ● 150

Thanks, decltype is exactly what I was looking for! – Verwirrt Apr 30, 2021 at 9:39