【C++】Day81

Class	C++
Date	@April 12, 2022
Material	
Series Number	
Summary	Definition of template class Blob

【Ch16】Templates and Generic Programming

Member Functions of Class Templates

A member function of a class template has the same template parameters as the class itself.

Therefore, a member function defined outside the class template body starts with the keyword `template` followed by the class' template parameter list.

To define a member function of `Blob`, we shall write:

```
template <typename T>
ret-type Blob<T>::member-name(param-list)
```

The check and Element Access Members

```
template <typename T>
void Blob<T>::check(const size_type i, const std::string &msg) const {
  if(i >= data->size())
    throw std::out_of_range(msg);
}
```

The subscript operator and `back` function use the template parameter to specify the return type but are otherwise unchanged.

`back()` function:

```
template <typename T>
T Blob<T>::back() const {
```

```
  check(0, "Back on empty Blob");
  return data->back();
}
```

subscript operator:

```
template <typename T>
T Blob<T>::operator[](const size_type i) const {
  check(i, "Subscript out of range");
  return (*data)[i];
}
```

The `pop_back()` function is basically the same:

```
template <typename T>
T Blob<T>::pop_back() const {
  check(0, "Cannot pop on empty Blob");
  return data->pop_back();
}
```

Blob Constructors

As with any other member defined outside a class template, a constructor starts by declaring the template parameters for the class template of which it is a member:

```
template <typename T>
Blob<T>::Blob() : data(std::make_shared<std::vector<T>>()) {}

template <typename T>
Blob<T>::Blob(std::initializer_list<T> il) : data(std::make_shared<std::vector<T>>(il)) {}
```

To use this constructor, we must pass an `initializer_list` in which the elements are compatible with the element type of the `Blob`:

```
Blob<string> articles = { "a", "an", "the" };
```

Instantiation of Class-Template Member Functions

By default, a member function of a class template is instantiated only if the program uses that member function. For example, this code

```
// instantiates Blob<int> and the initializer_list<int> constructor
Blob<int> squares = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
// instantiates Blob<int>::size() const
for(size_t i = 0; i != squares.size(); ++i)
  squares[i] = i * i;
```

instantiates the `Blob<int>` class and three of its member functions: `operator[]`, `size`, and the `initializer_list<int>` constructor.

Note: By default, a member of an instantiated class template is instantiated only if the member is used.