Scoped Enumerations

In this lesson, we will learn about Enumerations or Enums.

WE'LL COVER THE FOLLOWING

Enumerations conveniently define integer constants with names. These integer constants are called **enumerators**. Let's go over some of the drawbacks of using classic enums.

Drawbacks of Enumerations in Classical C++

A short reminder. Three drawbacks of enumerations:

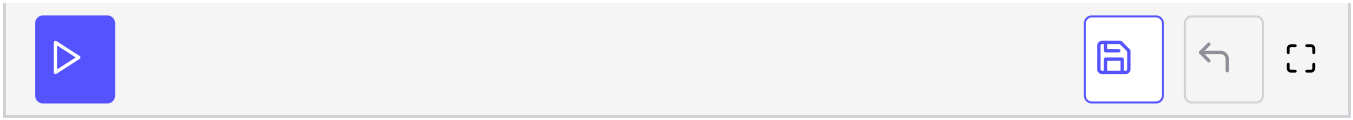
1. The enumerators implicitly convert to `int`.
2. They introduce the enumerators in the enclosing scope.
3. The type of the enumeration cannot be specified.

Regarding point 3, enumerations cannot be forward declared since their type unknown. We can only the guarantee for the enumerators in classical C++. The type must be integral and large enough to hold the enumerators.

Points 1 and 2 are more surprising.

```
// enumClassic.cpp

#include <iostream>

int main(){

  std::cout << std::endl;

  enum Colour{red= 0,green= 2,blue};
```

```
  enum Colour{red= 0,green= 2,blue};

  std::cout << "red: " << red << std::endl;
  std::cout << "green: " << green << std::endl;
  std::cout << "blue: " << blue << std::endl;

  int red2= red;

  std::cout << "red2: " << red2 << std::endl;

  // int red= 5;  ERROR

}
```



On the one hand, the enumerators `red`, `green`, and `blue` are known in the enclosing scope, meaning that the definition of the variable `red` in line 19 is not possible. On the other hand, red can be implicitly converted to `int`.

If we do not use a name for an enumeration like `enum{red, green, blue}`, the enumerators will be introduced in the enclosing scope.

But that surprise ends with C++11. Now that we have gone over the drawbacks, let's move on to strongly-typed enumerations.

Strongly-typed Enumerations

Scoped enumerations are often called **strongly-typed enumerations**. The strongly-typed enumerations have to follow stronger rules:

1. The enumerators can only be accessed in the scope of the enumeration.
2. The enumerators don't implicitly convert to `int`.
3. The enumerators aren't imported in the enclosing scope.
4. The type of the enumerators is by default `int`. Therefore, we can forward the enumeration.

The syntactical difference between the classic enumerations and the strongly-typed enumerations is minimal. The strongly-typed enumerations additionally get the keyword `class` or `struct`.

If we want to use an enumerator as an `int`, we have to explicitly convert it with `static_cast`.

```
// enumCast.cpp

#include <iostream>

enum OldEnum{
  one= 1,
  ten=10,
  hundred=100,
  thousand= 1000
};

enum struct NewEnum{
  one= 1,
  ten=10,
  hundred=100,
  thousand= 1000
};

int main(){

  std::cout << std::endl;

  std::cout << "C++11= " << 2*thousand + 0*hundred + 1*ten + 1*one << std::endl;
  std::cout << "C++11= " << 2*static_cast<int>(NewEnum::thousand) +
                            0*static_cast<int>(NewEnum::hundred) +
                            1*static_cast<int>(NewEnum::ten) +
                            1*static_cast<int>(NewEnum::one) << std::endl;

}
```

In order to calculate or output the enumerators, we must convert the enumerators into integral types. Neither the addition nor the output of strongly-typed enumerations is defined.

In this lesson, we have discussed classical versus strongly-typed enumerations. Commonly, these are known as scoped and unscoped enumerations.

Let’s look at an example of this topic in the next lesson.