Enum type in C++11

Originally enum types were created to be a simple type that allowed you to name a small number of named constants as a type. In the early use of enum in C++ the type was an integral type.

Here is a simple example:

`enum Answers{ Yes, No, Maybe};` // Yes will equal 0; No will equal 1; Maybe will equal 2.

They could also be unnamed :

`enum { FahrenheitBoiling = 100, CelsiusBoiling =  212};` //explicitly assigned

Here we see that specific integer values can be given to enumerators.

These plain enums do not provide scope and therefore enumerators (the named constants) can be ambiguous as in :

`enum Permissions{ Yes,  No};` // conflicting with enum Answers; if both in the same scope.

So C++11 introduces enum class.

```
    enum class  Permission { Yes, No};

    enum class AnswerClassEnum{Yes, No, Maybe};
```

provide distinct enumerators as each is strongly typed and scoped.

We pay a price in convenience as these enum class types do not automatically convert to int, as was the case for plain enums. So,

```
    int i = Answers::Yes; // automatically converts

    int j =  AnswerClassEnum::Yes;  //error  but can use a
static_cast<int>
```

Finally, with enum class we can specify the underlying integral type for enumerators as in:

```
    enum class CarType:short{ Sedan, Coupe, SUV,
Convertible};
```

These types can be char, short, int, long, long long, or their unsigned forms

Ira Pohl, January 18, 2016