

Re: Iterating through an enum???

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2004-04/1281.html>

From: muchan (*usenet_at_usenet.usenet*)

Date: 04/28/04

Date: Wed, 28 Apr 2004 13:37:21 +0200

Simon Trew wrote:

> "muchan" <*usenet@usenet.usenet*> wrote in message
> *news:Jzajc.1608\$37.234736@news.siol.net...*
>
>>> *The names of the enums don't exist at runtime, and no checking is done that*
>>> *the value is one of the defined values. So you can just use the enum as a*
>>> *number. The above example, using enums as a set of flags, is quite common.*
>
>> *Interesting. And in this case, what ++ operator should do with enum? 8)*
>
>
> *I think that is rather the point. Schobi defined a ++ operator which skipped*
> *through the *defined* values of the enum. I did the same, only in a more*
> *generic manner using a set to hold the values. (It would be better as a*
> *template class and I have in fact templated a version but I doubt it is*
> *worth posting here.) A specialized iterator onto an underlying vector would*
> *probably be a better choice; I coded up a set implementation as it avoided*
> *the use of an iterator (in the style of STL iterators, anyway) so the*
> *control variable could be of the enum type itself.*
>
> S.
>

Whatever the specification is, and whatever is theoretically possible, I stand that "enum" stands for "enumeration", so I'd recommend (myself) to enumerate all the possible value in the definition, either used or not used in a concrete program.

That is, if to values are combined (that is, |ed) to make combined value, it should be in enumeration list, like:

```
enum { a = 1, b, c, d, e = 10, ae = (a|e), be, ce, de };
```

ha! I don't know if such use of a and e is allowed in enum declaration...

If combination is infinite (well, always finite within the range of int...) and/or too many that it isn't practical to enumerate all, I think I should use integer instead of enum from the starting. :)

microsoft.public.vc.language: Re: Iterating through an enum???

Or, another (self) guidance would be all the explicit values in enum declaration should be defined as constant, either with #define or with const int.

(...and I don't know if const int declaration is treated with compiler without allocation in the runtime program...)

so above case

```
const int A = 1;
const int E = 10;
const int AE = (A|E);
enum abc { a=A, b, c, d, e=E, ae=AE, be, ce, de };
```

```
// this implicitly saying that other combination like b|c, a|d etc
// are not used in this program as a value for the type abc
```

Then, if I were to implement the iteration through declared enum values, the redundancy of duplicating the enumeration in enum declaration and array initialisation may introduce bug for careless editing, I would define entire definition string in macro:

```
#define ABC_ENUM { a = A, b, c, d, e = E, ae = AE, be, ce, de }
enum abc ABC_ENUM;
abc abcarray[] = ABC_ENUM;
std::vector<abc> v_abc(abcarray);
```

Then I'd like to reduce the duplicate memory, so the array declaration should be inside the constructor of a class, which wraps the enum from global namespace,

```
class ABC { // a wrapper class, or a real "user" of that enum
public:
    ABC(); // constructor

    //... functions

public: // or private, depends on how this class is used
    enum abc ABC_ENUM;

private:
    static std::vector<abc> v_abc; // vector to hold all the possible value, for iteration
};

std::vector<abc> ABC::v_abc; // static member must be declared somewhere

ABC::ABC()
{
    abc abcarray[] = ABC_ENUM; // temporary hold the value in array
    v_abc = std::vector(abcarray); // this is redundant, elements are copied twice!
    // but I don't know the better syntax...
// good if we could simply say
// v_abc = abcarray;
```

Re: Iterating through an enum???

microsoft.public.vc.language: Re: Iterating through an enum???

```
// with implicate conversion of array to std::vector...  
}
```

and now any function of class ABC can use

```
std::vector<abc>::const_iterator vi;  
for (vi = v_abc.begin(), v_abc.end(); ++vi) {  
    abc a = *vi;  
    // ...do something with a  
}
```

for the redundant correctness, the first const int declarations should be also defined inside some namespace or within a class...

Whether I ever need such code or not, that is the question. ;)

muchan