

Re: an string array problem

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2004-11/0384.html>

From: Filip Konvicka (*filip.konvicka.remove_this_antispam_token_please_at_logis.cz*)

Date: 11/09/04

Date: Tue, 9 Nov 2004 14:22:24 +0100

Hi Li,

it seems that you want to do lookup. This is best done using a "hashset", which is partly supported by the C++ standard. If you want to stick to the current standard, you may still use the "set" container. The code works as follows:

```
#include <set>
using namespace std;
typedef set<string> StringSet;
int main(int argc, char* argv[]) {
    StringSet mySet;
    // fill in the set
    mySet.insert(string("dot"));
    mySet.insert(string("polygon"));
    ...
    // look up a string
    string test("polygon");
    StringSet::iterator i, e=mySet.end();
    i=mySet.find(test);
    if ( i==e )
        cout << "Not found" << endl;
    else
        cout << "Found!" << end;
}
```

I haven't compiled this, so sorry for any typos... You may wrap the set into a global class (and fill in the set in the constructor), or just create a global instance of the set and fill it in the main() function before anything other takes place.

Searching in a set is faster than iterative comparisons (i.e. linear search) in an array. This is because the set is sorted internally (so for N items you only test at most log(N) items to find the minimum, compared to at most N items using an unsorted array). A hashset (stdext::hashset in recent VC and std::hashset or __gnu_cxx::hashset in g++) is even better: usually it only takes "constant time" to check for presence of a string in a hashset (i.e. for N items in the set, it only takes about 1 comparison to tell

whether the string is in the set). In other words: for big number of words, sets might be much more efficient than C arrays!! (For hashset, this even holds true for small numbers of words.)

If you need to store an element with your "key" (e.g. "polygon" = "3" and for "polygon" you want the "3"), use a "map" (resp. "hashmap").

Remember, the standard containers (std::vector, std::list, std::deque, std::set, std::map, stdext::hashset, stdext::hashmap and others) are your big friends in C++! Understanding lists, sets, queues and maps is very important if you want efficient programs.

Best Regards and Good Luck,
Filip

```
"david li" <davidli@augmentum.com.cn> píle v diskusním pøíspěvku
news:ucGh7yKxEHA.3212@TK2MSFTNGP09.phx.gbl...
> I want to use == function of string. because my string will be compared to
> that set. do I need to use strcmp instead of ==? sometime it's hard to
make
> decision. I want to do it in C++ way, remove all C feature. but C has more
> efficient.
> Thanks so much for your mail
>
> "josh" <smileyfaceswillruletheworld@yahoo.com.NOSPAM>
> ??????:#qKBGrixEHA.908@TK2MSFTNGP11.phx.gbl...
> > david li wrote:
> > > Hi, all:
> > >
> > > I have a question. In C we can use following way to get the string
> number in
> > > an array:
> > > const char* commandSet[] =
> > > {
> > > "dot",
> > > "polygon"
> > > "rectangle",
> > > "ellipse",
> > > "remove",
> > > "display",
> > > "help",
> > > "quit"
> > > };
> > >
> > > nStrNum = sizeof(commandSet) / sizeof(char*);
> > >
> > > but If we write it in C++:
> >
> > The above code is perfectly good C++. (given the right context, of
> course)
> >
```

```
>>> const string commandSet[] =
>>> {
>>> "dot",
>>> "polygon"
>>> "rectangle",
>>> "ellipse",
>>> "remove",
>>> "display",
>>> "help",
>>> "quit"
>>> };
>>>
>>> We can not use the way what we used in C. is there any ways to
implement
>>> that? or just put in a vector or list?
>>>
>>> Note that this version involves a number of heap allocations and string
>>> copies at runtime. Why would you want that? Don't use std::string for
>>> constants unless you have a very good reason.
>>>
>>> In any case, sizeof(commandSet)/sizeof(*commandSet) will work for either
>>> of these arrays.
>>>
>>> -josh
>>>
>>>
>>>
```