Sorting strings using qSort

Asked 13 years, 2 months ago Modified 2 years, 5 months ago Viewed 16k times



According to this site, I have done the following program which sorts strings.

5





```
#include <cstdlib>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char list[5][4]={"dat","mai","lik","mar","ana"};
int main(int argc, char *argv[])
{
    int x;
    puts("sortirebamde:");
    for (x=0;x>sizeof(list)/sizeof(char);x++)
        printf("%s\n",list[x]);
        qsort(&list,(sizeof(list)/sizeof(char)),sizeof(list[0]),strcmp);
        system("PAUSE");
        return EXIT_SUCCESS;
}
```

Here is the error I get

```
13 C:\Documents and Settings\LIBRARY\Desktop\string_sortireba.cpp invalid conversion from `int (*)(const char*, const char*)' to `int (*)(const void*, const void*)'
13 C:\Documents and Settings\LIBRARY\Desktop\string_sortireba.cpp initializing argument 4 of `void qsort(void*, size_t, size_t, int (*)(const void*, const void*))'
```

Please help

```
c++ c qsort
```

Share Edit Follow Flag



asked Sep 21, 2010 at 6:47



This is not a good way to sort strings in C, you should probably sort an array of char *, not a two-dimensional array with fixed length strings. – Sam Watkins Feb 6, 2015 at 7:43

it should be: for (x=0;x<sizeof(list)/sizeof(char);x++) – kuszi Nov 20, 2015 at 11:56



13

Please note: It is unusual to store C strings in two dimensional char arrays. It's more normal to have char *ary[], such as argy. That type cannot be sorted directly using qsort and strcmp, because gsort will pass char ** not char * to the comparison function. This is good for efficiency, the pointers can be swapped instead of the whole strings. The Linux manpage for gsort has some good example code with a correct comparison function.



You can't pass strcmp directly to quert as its comparison function because quert expects to pass pointers to void where strcmp expects pointers to const char. Given the required similarity between pointers to void and pointers to char, you could probably do it with a cast (for your code), but the cleaner way would be to write a function that takes the right types:

```
int cmpstr(void const *a, void const *b) {
    char const *aa = (char const *)a;
    char const *bb = (char const *)b;
   return strcmp(aa, bb);
}
```

Note, however, that in C++ you'd normally want to use std::sort instead of qsort, and probably use std::string instead of char *, which case the sorting gets a lot simpler (and generally faster as well).

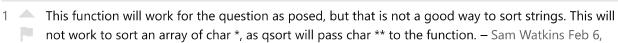
Share Edit Follow Flag

edited Jul 10, 2021 at 3:16

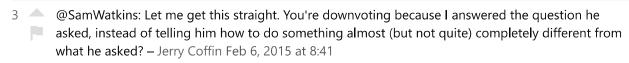
answered Sep 21, 2010 at 6:52

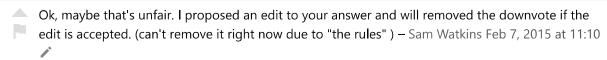


Jerry Coffin **480k** 81 634 1121



not work to sort an array of char *, as qsort will pass char ** to the function. – Sam Watkins Feb 6, 2015 at 7:41





1 for completeness, the cast version version would be: (int ()(const void, const void *))strcmp), see the working example – kuszi Nov 20, 2015 at 11:57 🧪



The fourth argument of qsort takes 2 void* pointers as args. So you need to define a compare function for yours. refer to this <u>link</u> for more details.



Share Edit Follow Flag



3,947 2 19 25

answered Sep 21, 2010 at 6:51





You can pass strcmp directly to gsort

```
#include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 char list[5][4]={"dat","mai","lik","mar","ana"};
 int main(int argc, char *argv[]) {
     int x;
     puts("unsorted:");
     for (x=0;x<sizeof(list)/sizeof(list[0]);x++)</pre>
          printf("%s\n",list[x]);
     qsort(list,sizeof(list)/sizeof(list[0]),sizeof(list[0]),strcmp);
     puts("sorted:");
     for (x=0;x<sizeof(list)/sizeof(list[0]);x++)</pre>
          printf("%s\n",list[x]);
 // system("PAUSE");
     return EXIT_SUCCESS;
 }
use C, not C++
```

answered Apr 19, 2014 at 8:58



3 — Do not pass strcmp directly to qsort, in the normal case (sorting an array of char *) it will not work, and it's always a type violation. – Sam Watkins Feb 6, 2015 at 7:42 🧪



Beyond why gsort fails, don't use it in C++.

```
#include <algorithm>
#include <iostream>
#include <iterator>
#include <string>
```

Share Edit Follow Flag

```
char const* const raw_data[5] = {"dat", "mai", "lik", "mar", "ana"};
```

```
std::vector<std::string> data (raw_data, raw_data + 5);
// would rarely be a global
// see below for code that needs to go here
int main() {
  using namespace std;
  cout << "before: " << data << "\n";</pre>
  sort(data.begin(), data.end());
  cout << "after: " << data << "\n";</pre>
  return 0;
}
```

Boost has stream inserter overloads to output a vector directly, but here's one <u>simple version</u>. This goes into a header, rather than being copied and pasted continually:

```
template<class Stream, class Iter, class Ch>
 void write_sequence(Stream& s, Iter begin, Iter end, Ch const* initial, Ch const*
 sep, Ch const* final) {
   if (initial) {
     s << initial;</pre>
   if (begin != end) {
     s << *begin;
     ++begin;
     for (; begin != end; ++begin) {
        if (sep) {
          s << sep;
        s << *begin;
     }
   }
   if (final) {
     s << final;
 }
 template<class T, class A>
 std::ostream& operator<<(std::ostream& s, std::vector<T,A> const& value) {
   write_sequence(s, value.begin(), value.end(), "[", ", ", "]");
   return s;
 }
Share Edit Follow Flag
                                                                answered Sep 21, 2010 at 7:24
                                  edited May 23, 2017 at 11:45
                                        Community Bot
                                                                      Roger Pate
                                        1
```

More C++-Style - nowadays - with static_cast:

```
•
```

```
int scmp(const void * s1, const void * s2)
{
    const char* _s1 = *static_cast<const char* const*>(s1);
    const char* _s2 = *static_cast<const char* const*>(s2);
    return strcmp(_s1, _s2);
}
```

And in main():

```
char *str_arr[] = { "one", "two", "three" };
qsort(str_arr, sizeof(str_array)/sizeof(char*), sizeof(char*), scmp);
```

Much easier is it with vector and std::sort.

Share Edit Follow Flag

