```
File - D:\cpl\2023-cpl-coding-0\11-function-pointers\bsearch-gnuc.c
 1 //
 2 // Created by hfwei on 2023/12/13.
 3 // Question: What if char key_name[] = "Zhang Chu"?
 4 //
 5
 6 #include <stdio.h>
 7 #include <string.h>
 8 #include <stdbool.h>
10 // See https://codebrowser.dev/glibc/glibc/stdlib/stdlib.h.html#
   __compar_fn_t
11 // The first is a pointer to the key for the search,
12 // and the second is a pointer to the array element to be compared
   with the key.
13 typedef int (*__compar_fn_t)(const void *, const void *);
15 // See https://codebrowser.dev/glibc/glibc/bits/stdlib-bsearch.h.html#
16 void *bsearch(const void *__key, const void *__base,
17
                  size_t __nmemb, size_t __size,
                  __compar_fn_t __compar);
18
19 void *bsearch_leftmost(const void *__key, const void *__base,
                           size_t __nmemb, size_t __size,
21
                           __compar_fn_t __compar);
22
23 int CompareStrs(const void *left, const void *right);
24 int CompareStrsCI(const void *left, const void *right);
25 int CompareStrsAddress(const void *left, const void *right);
27 // int (*GetCompareFunction(bool case_sensitive))(const void *, const
   void *);
28 __compar_fn_t GetCompareFunction(bool case_sensitive) {
29    return case_sensitive ? &CompareStrs : &CompareStrsCI;
30 }
31
32 const char *names[] = {
33
       "Cui Jian",
34
       "Dou Wei",
35
       "ErShou Rose",
36
       "Hu Mage",
       "Li Zhi",
37
38
       "Luo Dayou",
39
       "Wan Qing",
       "Yao",
40
41
       "Zhang Chu",
       "Zhang Chu",
42
       "Zhang Chu",
43
44
       "Zhang Chu",
45
       "ZuoXiao",
46 };
47
48 int main(void) {
     char *key_name = "Zhang Chu";
```

```
File - D:\cpl\2023-cpl-coding-0\11-function-pointers\bsearch-gnuc.c
      char *key_name_ci = "zhang chu";
 50
 51
 52
      // char **name_ptr = bsearch(&key_name, names,
 53
      //
                                     sizeof names / sizeof *names,
 54
      //
                                     sizeof *names,
 55
      //
                                     CompareStrs);
 56
 57
      // char **name_ptr = bsearch(&key_name, names,
 58
      //
                                     sizeof names / sizeof *names,
 59
      //
                                     sizeof *names,
 60
      //
                                     CompareStrsAddress);
 61
      // char **name_ptr = bsearch(&key_name, names,
 62
 63
      //
                                     sizeof names / sizeof *names,
 64
      //
                                     sizeof *names,
      //
 65
                                     (__compar_fn_t) strcmp); //
    CompareStrsAddress
 66
 67
      char **name_ptr = bsearch_leftmost(&key_name, names,
                                           sizeof names / sizeof *names,
 68
 69
                                           sizeof *names,
 70
                                           CompareStrs);
 71
 72
      // char **name_ptr = bsearch_leftmost(&key_name, names,
 73
                                              sizeof names / sizeof *names,
 74
      //
                                              sizeof *names,
 75
      //
                                              CompareStrsAddress);
 76
 77
      if (*name_ptr != NULL) {
 78
        printf("Found %s at index %lld.\n",
 79
                *name_ptr, name_ptr - (char **) names);
 80
      } else {
        printf("Could not find %s.\n", key_name);
 81
      }
 82
 83
 84
      char **name_ci_ptr = bsearch(&key_name_ci, names,
 85
                                     sizeof names / sizeof *names,
 86
                                     sizeof *names,
 87
                                     GetCompareFunction(false));
      if (*name_ci_ptr != NULL) {
 88
        printf("Found %s at index %lld.\n",
 89
 90
                *name_ci_ptr,
 91
                name_ci_ptr - (char **) names);
 92
      } else {
 93
        printf("Could not find %s.\n", key_name_ci);
 94
 95
 96
      return 0;
 97 }
 98
 99 // Visualization: https://pythontutor.com/render.html#code=//%0A//%
    20Created%20by%20hfwei%20on%202023/12/13.%0A//%20Question%3A%20What%
    20if%20char%20key_name%5B%5D%20%3D%20%22Zhang%20Chu%22%3F%0A//%0A%0A%
```

99 23include%20%3Cstdio.h%3E%0A%23include%20%3Cstring.h%3E%0A%0A//%20See %20https%3A//codebrowser.dev/qlibc/glibc/stdlib/stdlib.h.html% 23__compar_fn_t%0A//%20The%20first%20is%20a%20pointer%20to%20the% 20key%20for%20the%20search,%0A//%20and%20the%20second%20is%20a% 20pointer%20to%20the%20array%20element%20to%20be%20compared%20with% 20the%20key.%0Atypedef%20int%20%28*__compar_fn_t%29%28const%20void%20 *,%20const%20void%20*%29%3B%0A%0A//%20See%20https%3A//codebrowser.dev /glibc/glibc/bits/stdlib-bsearch.h.html%2319%0Avoid%20*bsearch% 28const%20void%20*__key,%20const%20void%20*__base,%20size_t%20__nmemb 20__compar_fn_t%20__compar%29%3B%0A%0Aint%20CompareStrs%28const% 20void%20*left,%20const%20void%20*right%29%3B%0Aint% 20CompareStrsAddress%28const%20char%20*left,%20const%20char%20*right% 29%3B%0A%0Aconst%20char%20*names%5B%5D%20%3D%20%7B%0A%20%20%20%20% 22Cui%20Jian%22,%0A%20%20%20%20%22Dou%20Wei%22,%0A%20%20%20%20% 22ErShou%20Rose%22,%0A%20%20%20%20%22Hu%20Mage%22,%0A%20%20%20%20% 22Li%20Zhi%22,%0A%20%20%20%20%22Luo%20Dayou%22,%0A%20%20%20%20%22Wan% 20Qing%22,%0A%20%20%20%20%22Yao%22,%0A%20%20%20%20%22Zhang%20Chu%22,% 0A%20%20%20%20%2ZUoXiao%22,%0A%7D%3B%0A%0Aint%2Omain%28void%29%20%7B %0A%20%20char%20*key_name%20%3D%20%22Zhang%20Chu%22%3B%0A%0A%20%20//% 20char%20**name_ptr%20%3D%20bsearch%28%26key_name, %20names, %0A%20%20 20%20%20%20%20sizeof%20names%20/%20sizeof%20*names,%0A%20%20//%20%20% 20%20sizeof%20*names,%0A%20%20//%20%20%20%20%20%20%20%20%20%20%20%20% 20strcmp%29%3B%20//%20CompareStrsAddress%0A%0A%20%20char%20**name_ptr %20%3D%20bsearch%28%26key_name, %20names, %0A%20%20%20%20%20%20%20%20% 20names%20/%20sizeof%20*names,%0A%20%20%20%20%20%20%20%20%20%20%20%20 20%20%20%20%20CompareStrs%29%3B%0A%0A%20%20if%20%28*name ptr%20!%3D% 20NULL%29%20%7B%0A%20%20%20%20printf%28%22Found%20%25s.%5Cn%22,%20* name_ptr%29%3B%0A%20%20%7D%20else%20%7B%0A%20%20%20printf%28% 22Could%20not%20find%20%25s.%5Cn%22,%20key_name%29%3B%0A%20%20%7D%0A% 0A%20%20return%200%3B%0A%7D%0A%0Aint%20CompareStrs%28const%20void%20* left,%20const%20void%20*right%29%20%7B%0A%20%20char%20*const%20*pp1% 20%3D%20left%3B%0A%20%20char%20*const%20*pp2%20%3D%20right%3B%0A%20% 20return%20strcmp%28*pp1,%20*pp2%29%3B%0A%7D%0A%0A//%20What%20is% 20the%20advantage%20of%20this%20version%3F%20%28performance%3F%3F%3F% 29%0A//%20What%20is%20the%20disadvantage%20of%20this%20version%3F%20% 28not%20flexible%3F%3F%3F%29%0Aint%20CompareStrsAddress%28const% 20char%20*left,%20const%20char%20*right%29%20%7B%0A%20%20return% 20strcmp%28left,%20right%29%3B%0A%7D%0A%0Avoid%20*bsearch%28const% 20void%20*__key,%20const%20void%20*__base,%20size_t%20__nmemb,% 20size_t%20__size,%0A%20%20%20%20%20%20%20%20%20%20%20%20% 20__compar_fn_t%20__compar%29%20%7B%0A%20%20size_t%20__l,%20__u,% 20__idx%3B%0A%20%20const%20void%20*__p%3B%0A%20%20int%20__comparison% 3B%0A%20%20__1%20%3D%200%3B%0A%20%20__u%20%3D%20__nmemb%3B%0A%20% 20while%20%28__1%20%3C%20__u%29%20%7B%0A%20%20%20%20__idx%20%3D%20% 28__1%20%2B%20__u%29%20/%202%3B%0A%20%20%20%20__p%20%3D%20%28const% 20void%20*%29%20%28%28%28const%20char%20*%29%20 base%29%20%2B%20%

```
99 28__idx%20*%20__size%29%29%3B%0A%20%20%20%20__comparison%20%3D%20%28*
    __compar%29%28__key,%20__p%29%3B%0A%20%20%20%20if%20%28__comparison%
    20%3C%200%29%20%7B%0A%20%20%20%20%20__u%20%3D%20__idx%3B%0A%20%20%
    20%20%7D%20else%20if%20%28__comparison%20%3E%200%29%20%7B%0A%20%20%20
   %20%20%20__l%20%3D%20__idx%20%2B%201%3B%0A%20%20%20%20%7D%20else%20%
    7B%0A%20%20%20%20%20return%20%28void%20*%29%20__p%3B%0A%20%20%20%
    20%7D%0A%20%20%7D%0A%0A%20%20return%20NULL%3B%0A%7D&cppShowMemAddrs=
    true&cumulative=true&curInstr=14&heapPrimitives=nevernest&mode=
    display&origin=opt-frontend.js&py=c_qcc9.3.0&rawInputLstJSON=%5B%5D&
    textReferences=false
100 int CompareStrs(const void *left, const void *right) {
101
     char *const *pp1 = left;
102
     char *const *pp2 = right;
103
     return strcmp(*pp1, *pp2);
104 }
105
106 int CompareStrsCI(const void *left, const void *right) {
     const char *const *pp1 = left;
108
     const char *const *pp2 = right;
109
     // see https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-
    strcasecmp-case-insensitive-string-comparison
     return strcasecmp(*pp1, *pp2);
110
111 }
112
113 // What is the advantage of this version? (performance???)
114 // What is the disadvantage of this version? (not flexible???)
115 // Visualization: https://pythontutor.com/render.html#code=//%0A//%
    20Created%20by%20hfwei%20on%202023/12/13.%0A//%20Question%3A%20What%
    20if%20char%20key_name%5B%5D%20%3D%20%22Zhang%20Chu%22%3F%0A//%0A%0A%
    23include%20%3Cstdio.h%3E%0A%23include%20%3Cstring.h%3E%0A%0A//%20See
   %20https%3A//codebrowser.dev/glibc/glibc/stdlib/stdlib.h.html%
    23__compar_fn_t%0A//%20The%20first%20is%20a%20pointer%20to%20the%
    20key%20for%20the%20search,%0A//%20and%20the%20second%20is%20a%
    20pointer%20to%20the%20array%20element%20to%20be%20compared%20with%
    20the%20key.%0Atypedef%20int%20%28*__compar_fn_t%29%28const%20void%20
    *,%20const%20void%20*%29%3B%0A%0A//%20See%20https%3A//codebrowser.dev
    /glibc/glibc/bits/stdlib-bsearch.h.html%2319%0Avoid%20*bsearch%
    28const%20void%20*__key,%20const%20void%20*__base,%20size_t%20__nmemb
    20__compar_fn_t%20__compar%29%3B%0A%0Aint%20CompareStrs%28const%
    20void%20*left,%20const%20void%20*right%29%3B%0Aint%
    20CompareStrsAddress%28const%20char%20*left,%20const%20char%20*right%
    29%3B%0A%0Aconst%20char%20*names%5B%5D%20%3D%20%7B%0A%20%20%20%20%
    22Cui%20Jian%22,%0A%20%20%20%20%2Dou%20Wei%22,%0A%20%20%20%20%
    22ErShou%20Rose%22,%0A%20%20%20%20%22Hu%20Mage%22,%0A%20%20%20%20%
    22Li%20Zhi%22,%0A%20%20%20%20%22Luo%20Dayou%22,%0A%20%20%20%20%22Wan%
    20Qing%22,%0A%20%20%20%20%22Yao%22,%0A%20%20%20%20%22Zhang%20Chu%22,%
    0A%20%20%20%20%2ZUoXiao%22,%0A%7D%3B%0A%0Aint%2Omain%28void%29%20%7B
    %0A%20%20char%20*key_name%20%3D%20%22Zhang%20Chu%22%3B%0A%0A%20%20//%
    20char%20**name_ptr%20%3D%20bsearch%28%26key_name,%20names,%0A%20%20
    20%20%20%20%20sizeof%20names%20/%20sizeof%20*names,%0A%20%20//%20%20%
```

20strcmp%29%3B%20//%20CompareStrsAddress%0A%0A%20%20char%20**name_ptr %20%3D%20bsearch%28%26key_name,%20names,%0A%20%20%20%20%20%20%20%20% 20names%20/%20sizeof%20*names,%0A%20%20%20%20%20%20%20%20%20%20%20%20 20%20%20%20%20CompareStrsAddress%29%3B%0A%0A%20%20if%20%28*name_ptr% 20!%3D%20NULL%29%20%7B%0A%20%20%20%20printf%28%22Found%20%25s.%5Cn%22 ,%20*name_ptr%29%3B%0A%20%20%7D%20else%20%7B%0A%20%20%20020printf%28% 22Could%20not%20find%20%25s.%5Cn%22,%20key_name%29%3B%0A%20%20%7D%0A% 0A%20%20return%200%3B%0A%7D%0A%0Aint%20CompareStrs%28const%20void%20* left,%20const%20void%20*right%29%20%7B%0A%20%20char%20*const%20*pp1% 20%3D%20left%3B%0A%20%20char%20*const%20*pp2%20%3D%20right%3B%0A%20% 20return%20strcmp%28*pp1,%20*pp2%29%3B%0A%7D%0A%0A//%20What%20is% 20the%20advantage%20of%20this%20version%3F%20%28performance%3F%3F%3F% 29%0A//%20What%20is%20the%20disadvantage%20of%20this%20version%3F%20% 28not%20flexible%3F%3F%3F%29%0Aint%20CompareStrsAddress%28const% 20char%20*left,%20const%20char%20*right%29%20%7B%0A%20%20return% 20strcmp%28left,%20right%29%3B%0A%7D%0A%0Avoid%20*bsearch%28const% 20void%20*__key,%20const%20void%20*__base,%20size_t%20__nmemb,% 20__compar_fn_t%20__compar%29%20%7B%0A%20%20size_t%20__l,%20__u,% 20__idx%3B%0A%20%20const%20void%20*__p%3B%0A%20%20int%20__comparison% 3B%0A%20%20__1%20%3D%200%3B%0A%20%20__u%20%3D%20__nmemb%3B%0A%20% 20while%20%28__1%20%3C%20__u%29%20%7B%0A%20%20%20%20__idx%20%3D%20% 28__1%20%2B%20__u%29%20/%202%3B%0A%20%20%20%20__p%20%3D%20%28const% 20void%20*%29%20%28%28%28const%20char%20*%29%20__base%29%20%2B%20% 28__idx%20*%20__size%29%29%3B%0A%20%20%20__comparison%20%3D%20%28* __compar%29%28__key,%20__p%29%3B%0A%20%20%20%20if%20%28__comparison% 20%3C%200%29%20%7B%0A%20%20%20%20%20__u%20%3D%20__idx%3B%0A%20%20% 20%20%7D%20else%20if%20%28__comparison%20%3E%200%29%20%7B%0A%20%20%20 %20%20%20__l%20%3D%20__idx%20%2B%201%3B%0A%20%20%20%20%7D%20else%20% 7B%0A%20%20%20%20%20return%20%28void%20*%29%20__p%3B%0A%20%20%20% 20%7D%0A%20%20%7D%0A%0A%20%20return%20NULL%3B%0A%7D&cppShowMemAddrs= true&cumulative=true&curInstr=30&heapPrimitives=nevernest&mode= $display \& origin=opt-front end. js \& py=c_gcc9.3.0 \& raw Input Lst JSON=\%5B\%5D\& policy for the property of t$ textReferences=false 116 int CompareStrsAddress(const void *left, const void *right) { const char *pp1 = left; 117 118 const char *pp2 = right; 119 return strcmp(pp1, pp2); 120 } 121 122 void *bsearch(const void *__key, const void *__base, size_t __nmemb, size_t __size, 123 __compar_fn_t __compar) { 124 size_t __l, __u, __idx; const void *__p; 125 126 int __comparison; 127 __l = 0;

128

```
File - D:\cpl\2023-cpl-coding-0\11-function-pointers\bsearch-gnuc.c
129
       while (__l < __u) {
130
         _{\text{ldx}} = (_{\text{l}} + _{\text{l}} \cup ) / 2;
         __p = (const void *) (((const char *) __base) + (__idx * __size
131
     ));
         \_comparison = (*_compar)(\_key, \_p);
132
133
         if (__comparison < 0) {
134
           _{\_}u = _{\_}idx;
135
         } else if (__comparison > 0) {
            _{-1} = _{-idx} + 1;
136
         } else {
137
138
           return (void *) __p;
         }
139
       }
140
141
142
       return NULL;
143 }
144
145 void *bsearch_leftmost(const void *__key, const void *__base,
146
                               size_t __nmemb, size_t __size,
147
                               __compar_fn_t __compar) {
148
       size_t __l, __u, __idx;
149
       const void *__p;
150
       int __comparison;
151
152
       __l = 0;
       __u = __nmemb;
// added by ant
153
154
       void *__index = NULL;
155
156
       while (_{l} < _{l} )  {
157
158
         _{-idx} = (_{-l} + _{-u}) / 2;
         __p = (const void *) (((const char *) __base) + (__idx * __size
159
     ));
          __comparison = (*__compar)(__key, __p);
160
161
         if (__comparison < 0) {
162
            _{\_}u = _{\_}idx;
         } else if (__comparison > 0) {
163
164
            _{-}l = _{-}idx + _{1};
         } else {
165
166
           // added by ant
167
            __index = (void *) __p;
168
            _{u} = _{idx} - 1;
         }
169
       }
170
171
172
       // added by ant
173
       return __index;
174 }
```