# `bsearch`

Article • 12/02/2022

Performs a binary search of a sorted array. A more secure version of this function is available; see bsearch_s.

## Syntax

```C
void *bsearch(
   const void *key,
   const void *base,
   size_t num,
   size_t width,
   int ( __cdecl *compare ) (const void *key, const void *datum)
);
```

### Parameters

*`key`*
Pointer to the key to search for.

*`base`*
Pointer to the base of the search data.

*`number`*
Number of elements.

*`width`*
Width of elements.

*`compare`*
Callback function that compares two elements. The first is a pointer to the key for the search, and the second is a pointer to the array element to be compared with the key.

## Return value

**`bsearch`** returns a pointer to an occurrence of *`key`* in the array pointed to by *`base`*. If *`key`* isn't found, the function returns `NULL`. If the array isn't in ascending sort order or contains duplicate records with identical keys, the result is unpredictable.

## Remarks

The `bsearch` function performs a binary search of a sorted array of *`number`* elements, each of *`width`* bytes in size. The *`base`* value is a pointer to the base of the array to be searched, and *`key`* is the value being sought. The *`compare`* parameter is a pointer to a user-supplied routine that compares the requested key to an array element. It returns one of the following values that specify their relationship:

Expand table

| Value returned by *`compare`* routine | Description |
|---|---|
| `< 0` | Key is less than array element. |
| `0` | Key is equal to array element. |
| `> 0` | Key is greater than array element. |

This function validates its parameters. If *`compare`*, *`key`* or *`number`* is `NULL`, or if *`base`* is `NULL` and *`number`* is nonzero, or if *`width`* is zero, the function invokes the invalid parameter handler, as described in Parameter validation. If execution is allowed to continue, `errno` is set to `EINVAL` and the function returns `NULL`.

By default, this function's global state is scoped to the application. To change this behavior, see Global state in the CRT.

## Requirements

Expand table

| Routine | Required header |
|---|---|
| `bsearch` | <stdlib.h> and <search.h> |

For more compatibility information, see Compatibility.

## Example

This program sorts a string array with qsort, and then uses bsearch to find the word "cat".

C

```
// crt_bsearch.c
#include <search.h>
#include <string.h>
#include <stdio.h>

int compare( char **arg1, char **arg2 )
{
   /* Compare all of both strings: */
   return _strcmpi( *arg1, *arg2 );
}

int main( void )
{
   char *arr[] = {"dog", "pig", "horse", "cat", "human", "rat", "cow",
"goat"};
   char **result;
   char *key = "cat";
   int i;

   /* Sort using Quicksort algorithm: */
   qsort( (void *)arr, sizeof(arr)/sizeof(arr[0]), sizeof( char * ), (int
(*)(const
   void*, const void*))compare );

   for( i = 0; i < sizeof(arr)/sizeof(arr[0]); ++i )    /* Output sorted
list */
      printf( "%s ", arr[i] );

   /* Find the word "cat" using a binary search algorithm: */
   result = (char **)bsearch( (char *) &key, (char *)arr,
sizeof(arr)/sizeof(arr[0]),
                              sizeof( char * ), (int (*)(const void*, const
void*))compare );
   if( result )
      printf( "\n%s found at %Fp\n", *result, result );
   else
      printf( "\nCat not found!\n" );
}
```

Output

```
cat cow dog goat horse human pig rat
cat found at 002F0F04
```

## See also

Searching and sorting
_lfind

_lsearch

qsort