Description

The C library function **void \*bsearch(const void \*key, const void \*base, size\_t nitems, size\_t size, int (\*compar)(const void \*, const void \*))** function searches an array of **nitems** objects, the initial member of which is pointed to by **base**, for a member that matches the object pointed to by **key**. The size of each member of the array is specified by **size**.

The contents of the array should be in ascending sorted order according to the comparison function referenced by **compar**.

Declaration

Following is the declaration for bsearch() function.

void \*bsearch(const void \*key, const void \*base, size\_t nitems, size\_t size, int (\*compar)(const void \*, const void \*))

Parameters

* **key**-- This is the pointer to the object that serves as key for the search, type-casted as a void\*.
* **base**-- This is the pointer to the first object of the array where the search is performed, type-casted as a void\*.
* **nitems**-- This is the number of elements in the array pointed by base.
* **size**-- This is the size in bytes of each element in the array.
* **compar**-- This is the function that compares two elements.

Return Value

This function returns a pointer to an entry in the array that matches the search key. If key is not found, a NULL pointer is returned.

Example

The following example shows the usage of bsearch() function.

#include <stdio.h>

#include <stdlib.h>

int cmpfunc(const void \* a, const void \* b)

{

return ( \*(int\*)a - \*(int\*)b );

}

int values[] = { 5, 20, 29, 32, 63 };

int main ()

{

int \*item;

int key = 32;

/\* using bsearch() to find value 32 in the array \*/

item = (int\*) bsearch (&key, values, 5, sizeof (int), cmpfunc);

if( item != NULL )

{

printf("Found item = %d\n", \*item);

}

else

{

printf("Item = %d could not be found\n", \*item);

}

return(0);

}

Let us compile and run the above program, this will produce the following result:

Found item = 32