qsort: sorting array of strings, integers and structs

abstract

qsort() is standard C function for sorting arrays. It is defined by ISO C standard, and implemented in most C/C++ standard libraries(stdlib.h). This article contains an example of using qsort() for sorting integers, strings and structs.

compatible

- Any ANSI C compiler
- Should work with most C++ compilers

qsort() takes four arguments:

> void **qsort**(void *base, size_t nel, size_t width, int (*compar)(const void *, const void *));

base — is a pointer to the beginning of data array
nel — is a number of elements
width — is a size of each element (in bytes)
compar — is a callback function (pointer to function), which does comparison and returns positive or negative integer depending on result.

This example contains three separate functions sort_integers_example(), sort_cstrings_example() and sort_structs_example().

sort_integers_example() uses int_cmp() as *compar* callback. Additional function print_int_array is used for printing integer array contents.
sort_cstrings_example() uses cstring_cmp() as *compar* callback. Additional function print_cstring_array is used for printing string array contents.
sort_structs_example() uses struct_cmp_by_price() and struct_cmp_by_product() as *compar* callbacks. Additional function print_struct_array is used for printing array of struct.

source code: C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* qsort int comparison function */
int int_cmp(const void *a, const void *b)
{
    const int *ia = (const int *)a; // casting pointer
    const int *ib = (const int *)b;
    return *ia  - *ib;
        /* integer comparison: returns negative if b > a
        and positive if a > b */
}
```

```c
/* integer array printing function */
void print_int_array(const int *array, size_t len)
{
    size_t i;

    for(i=0; i<len; i++)
        printf("%d | ", array[i]);

    putchar('\n');
}

/* sorting integers using qsort() example */
void sort_integers_example()
{
    int numbers[] = { 7, 3, 4, 1, -1, 23, 12, 43, 2, -4, 5 };
    size_t numbers_len = sizeof(numbers)/sizeof(int);

    puts("*** Integer sorting...");

    /* print original integer array */
    print_int_array(numbers, numbers_len);

    /* sort array using qsort functions */
    qsort(numbers, numbers_len, sizeof(int), int_cmp);

    /* print sorted integer array */
    print_int_array(numbers, numbers_len);
}

/* qsort C-string comparison function */
int cstring_cmp(const void *a, const void *b)
{
    const char **ia = (const char **)a;
    const char **ib = (const char **)b;
    return strcmp(*ia, *ib);
       /* strcmp functions works exactly as expected from
       comparison function */
}

/* C-string array printing function */
void print_cstring_array(char **array, size_t len)
{
    size_t i;

    for(i=0; i<len; i++)
        printf("%s | ", array[i]);

    putchar('\n');
}

/* sorting C-strings array using qsort() example */
void sort_cstrings_example()
{
    char *strings[] = { "Zorro", "Alex", "Celine", "Bill", "F
    size_t strings_len = sizeof(strings) / sizeof(char *);

    /** STRING */
    puts("*** String sorting...");

    /* print original string array */
    print_cstring_array(strings, strings_len);
```

```
    /* sort array using qsort functions */
    qsort(strings, strings_len, sizeof(char *), cstring_cmp);

    /* print sorted string array */
    print_cstring_array(strings, strings_len);
}



/* an example of struct */
struct st_ex {
    char product[16];
    float price;
};

/* qsort struct comparision function (price float field) */
int struct_cmp_by_price(const void *a, const void *b)
{
    struct st_ex *ia = (struct st_ex *)a;
    struct st_ex *ib = (struct st_ex *)b;
    return (int)(100.f*ia->price - 100.f*ib->price);
       /* float comparison: returns negative if b > a
       and positive if a > b. We multiplied result by 100.0
       to preserve decimal fraction */

}

/* qsort struct comparision function (product C-string field)
int struct_cmp_by_product(const void *a, const void *b)
{
    struct st_ex *ia = (struct st_ex *)a;
    struct st_ex *ib = (struct st_ex *)b;
    return strcmp(ia->product, ib->product);
       /* strcmp functions works exactly as expected from
       comparison function */
}

/* Example struct array printing function */
void print_struct_array(struct st_ex *array, size_t len)
{
    size_t i;

    for(i=0; i<len; i++)
        printf("[ product: %s \t price: $%.2f ]\n", array[i].

    puts("--");
}

/* sorting structs using qsort() example */
void sort_structs_example(void)
{
    struct st_ex structs[] = {{"mp3 player", 299.0f}, {"plasm
                              {"notebook", 1300.0f}, {"smartp
                              {"dvd player", 150.0f}, {"match

    size_t structs_len = sizeof(structs) / sizeof(struct st_e

    puts("*** Struct sorting (price)...");

    /* print original struct array */
    print_struct_array(structs, structs_len);
```

```
    /* sort array using qsort functions */
    qsort(structs, structs_len, sizeof(struct st_ex), struct_

    /* print sorted struct array */
    print_struct_array(structs, structs_len);

    puts("*** Struct sorting (product)...");

    /* resort using other comparision function */
    qsort(structs, structs_len, sizeof(struct st_ex), struct_

    /* print sorted struct array */
    print_struct_array(structs, structs_len);
}


/* MAIN program (calls all other examples) */
int main()
{
    /* run all example functions */
    sort_integers_example();
    sort_cstrings_example();
    sort_structs_example();
    return 0;
}
?>
```

on result:

```
teger sorting...
| 4 | 1 | -1 | 23 | 12 | 43 | 2 | -4 | 5 |
1 | 1 | 2 | 3 | 4 | 5 | 7 | 12 | 23 | 43 |
ring sorting...
| Alex | Celine | Bill | Forest | Dexter |
 Bill | Celine | Dexter | Forest | Zorro |
ruct sorting (price)...
uct: mp3 player    price: $299.00 ]
uct: plasma tv     price: $2200.00 ]
uct: notebook      price: $1300.00 ]
uct: smartphone    price: $499.99 ]
uct: dvd player    price: $150.00 ]
uct: matches       price: $0.20 ]

uct: matches       price: $0.20 ]
uct: dvd player    price: $150.00 ]
uct: mp3 player    price: $299.00 ]
uct: smartphone    price: $499.99 ]
uct: notebook      price: $1300.00 ]
uct: plasma tv     price: $2200.00 ]

ruct sorting (product)...
uct: dvd player    price: $150.00 ]
uct: matches       price: $0.20 ]
uct: mp3 player    price: $299.00 ]
uct: notebook      price: $1300.00 ]
uct: plasma tv     price: $2200.00 ]
uct: smartphone    price: $499.99 ]
```

warning

- Never use strcmp() function in real-world applications for untrusted data. Use strncmp() or strlcmp() instead.

tested by AnyExample.com on 2007-06-15

- Windows XP :: Microsoft Visual C++ 2003
- Free BSD 5.2 :: gcc 3.3.3