Функции сортировки и поиска

```
#include <stdlib.h>
```

- **qsort()** – сортировка массива в возрастающем порядке
- **bsearch()** – бинарный поиск в отсортированном массиве

Функция qsort

Сортировка массива, порядок сортировки задается функцией compar

Прототип функции:

```
void qsort(void * base,
           size_t nmemb,
           size_t size,
           int(*compar)(const void *, const void *));
```

Аргументы qsort():

- `void * base` – указатель на сортируемый массив
- `size_t nmemb` – число элементов в массиве
- `size_t size` – размер одного элемента (в битах)
- `compar` – указатель на функцию сравнения двух элементов:
  `int compar(const void *arg1, const void* arg2)`

Пример

```
int main() {
  int numbers[]={43,76,23,1,100,56,23,99,33,654};
  int how_many=sizeof(numbers)/sizeof(numbers[0]);
  printf("\n These are the unsorted numbers\n");
  display_nums(numbers, how_many);

  qsort(
        numbers,              /* Pointer to elements */
        how_many,             /* Number of elements  */
        sizeof(numbers[0]),   /* size of one element */
        &comp_nums            /* comparison function */
       );

  printf("\n These are the sorted numbers\n");
  display_nums(numbers, how_many);
}
```

Функция сравнения

`int compar(const void *arg1, const void* arg2)`

☞ сортировка в возрастающем порядке определяется значением возвращаемым функцией `compar`:

сравнение элементов	возвращаемое значение
$arg1 < arg2$	меньше нуля
$arg1 == arg2$	нуль
$arg1 > arg2$	больше нуля

Продолжение примера

```
int comp_nums(const void *n1, const void *n2) {
  int* num1 = (int*)n1;
  int* num2 = (int*)n2;
  if (*num1 <  *num2) return -1;
  if (*num1 == *num2) return  0;
  if (*num1 >  *num2) return  1;
}
```

Печать элементов массива

```
void display_nums(int *array, int count) {
  while ( count-- ) {
    printf("%d ",*(array++));
  }
  printf("\n");
}
```

Output:

```
These are the unsorted numbers
43 76 23 1 100 56 23 99 33 654

These are the sorted numbers
1 23 23 33 43 56 76 99 100 654
```

Функция bsearch

Двоичный поиск в отсортированном массиве

Прототип функции:

```
void *bsearch(const void *key,
              const void *buf,
              size_t nmemb, size_t size,
              int (*compare) (const void *, const void *));
```

Комментарии:

- `void *key` – указатель на искомый ключ-значение
- `void *buf` – указатель на массив, отсортированный функцией `compar`
- `size_t nmemb, size_t size, compar` – смотри `qsort`
- `bsearch()` возвращает указатель на первый элемент массива совпадающий с искомым или `NULL`

Поиск в отсортированном массиве (продолжение примера)

```
int num = 0;
printf("\n Input the number:");
scanf("%i",&num);

int* pnt = (int *) bsearch(
      &num,                /* that we are looking */
      numbers,             /* where we are looking for */
      how_many,            /* number of elements  */
      sizeof(numbers[0]),  /* size of one element */
      &comp_nums           /* comparison function */
     );

if( pnt ) {
  printf("\n The number %d is %d element of array\n",num,pnt-numbers);
} else {
  printf("\n The number %d is not in array\n",num);
}
```

Тест №1

```
These are the sorted numbers
1 23 23 33 43 56 76 99 100 654

Input the number:56

The number 56 is 5 element of array
```

Тест №2

```
These are the sorted numbers
1 23 23 33 43 56 76 99 100 654

Input the number:55

The number 55 is not in array
```