Bubble sort (C)

From LiteratePrograms

Other implementations: C | CLIPS | C++ | C# | Eiffel | Erlang | Io | Java | Lisp | Smalltalk

A somewhat generic implementation

The support for functional programming is not very elaborated in C. However you can have function pointers and this come in very handy, especially for sorting algorithms.

I have mimiced the interface of the standard function qsort. So it looks like this:

<<bubble_sort function>>=
```
/* mimics the qsort interface */
void bubble_sort(void *base, size_t nmemb, size_t size,
                 int (*compar)(const void *, const void *)) {
  int i, j;
  char *pc =  base;
  char *pc_at_i;
  char *pc_at_j;

  for (i = nmemb -1; i > 0; --i){
    for (j = 0; j < i; ++j) {
         /* we have to calculate the offsets, by defintion the size of
         char is 1 in C, so we do not have to include  the size of the
         elements while doing this address calculations */
      pc_at_i = pc + (i * size);
      pc_at_j = pc + (j * size);
      if (compar (pc_at_i, pc_at_j) <  0) {
                swap_fun(base, size, i, j);
      }
    }
  }
}
```

I used two extra functions for comparison and swapping of elements. The swapping of elements was borrowed from the Quicksort page and looks like this:

<<swap_fun function>>=
```
/* Swapping of elements in an array. Because there is a void* we need to give
this function the element_size of the to be sorted elements, we then can exchange the
array elements character by character. */
static void swap_fun (void *base, size_t element_size,
                          int index1, int index2) {
        char *pc = base;
        char tmp;
        int i;
        for (i = 0; i < element_size; ++i) {
                tmp = pc[index1 * element_size + i];
        pc[index1 * element_size + i] = pc[index2*element_size + i];
        pc[index2 * element_size + i] = tmp;
        }
}
```

I just commented it a bit, because it's not fully clear to a C-outsider why one has to fall back to some character-wise operations.

Now the thing left is the comparison function. Here's the implementation:

```
<<int_cmp_fun function>>=
int int_cmp_fun (const void * v1, const void * v2) {
  const int * i1 = v1;
  const int * i2 = v2;
  int result;
  if (*i1 == *i2) {
    result = 0;
  } else if (*i1 < *i2) {
    result = -1;
  } else {
    result = 1;
  }
  return result;
}
```

We can use the functions as follows

Bubble-sort in action

Here a simple example for this sorting function:

```
<<bubble_sort.c>>=

#include <stdio.h>

swap_fun function

bubble_sort function

int_cmp_fun function

static void print_int_arr(int *arr, size_t size_of_arr) {
  int i;
  for (i = 0; i < size_of_arr; i++) {
    printf("%d ", arr[i]);
  }
  putchar('\n');
}


int main(void) {
  enum { T_SIZE = 7};
  int arr[T_SIZE] = {-1, 2, 1, 3, 5, -10, -11};

  printf("array before sorting: ");
  print_int_arr(arr, T_SIZE);
  printf("array after bubblesort: ");
  bubble_sort(arr, T_SIZE, sizeof(int), int_cmp_fun);
  print_int_arr(arr, T_SIZE);
  return 0;
}
```

We get the following output:

```
./a.out
```

```
array before sorting: -1 2 1 3 5 -10 -11
array after bubblesort: -11 -10 -1 1 2 3 5
```

So if you are using C, consider using function pointers, the are a really helpful utility.

Download code (http://en.literateprograms.org/index.php?title=Special:Downloadcode/Bubble_sort_(C)&oldid=15710)

Retrieved from "http://en.literateprograms.org/index.php?title=Bubble_sort_(C)&oldid=15710"
Categories: Programming language:C | Environment:Portable | Bubble sort