KIT205 Data Structures and Algorithms

Week 6 Tutorial

In this tutorial you will be implementing quicksort for strings. In addition, you will practice some string functions and input redirection, which can help with testing.

Strings

1. Create a new empty project
2. Add a file called *main.c* with a *main* method
3. Add the following typedef

```
typedef char *String;
```

4. Add a variable called *strings* that is a *String* pointer that can be used to store an array of strings
5. Add a variable called *buffer* that is large enough to store a 100 character string
6. Add some code to prompt the user for the number or strings and store the result in a variable, *n*
7. Initialise the *strings* array so that it has size *n*
8. Add a for loop that reads *n* strings into *buffer*
9. Now add the following code (within the for loop) to copy from the buffer into the array

```
strings[i] = (String)malloc(strlen(buffer) + 1);
strcpy(strings[i], buffer);
```

10. Finally, add another loop to print the array.

> *This typedef is not required, we could just continue using char**
>
> *However, when we make an array of strings, this would become char***, *which can get confusing.*
>
> *Instead of char**, we can now write String*, which is more readable*

Make sure that you test and understand this string code before continuing.

11. Add a loop to free memory for all of the strings and the string array

Input Redirection

We want to test our sorting code with a large number of strings. Like these 100 for example:

sausage
blubber
pencil
cloud
moon
water
computer
school
network
hammer
walking

violently
mediocre
literature
chair
two
window
cords
musical
zebra
xylophone
penguin
home
dog
final
ink
teacher
fun
website
banana
uncle
softly
mega
ten
awesome
attatch
blue
internet
bottle
tight
zone
tomato
prison
hydro
cleaning
telivision
send
frog
cup
book
zooming
falling
evily
gamer
lid
juice

moniter
captain
bonding
loudly
thudding
guitar
shaving
hair
soccer
water
racket
table
late
media
desktop
flipper
club
flying
smooth
monster
purple
guardian
bold
hyperlink
presentation
world
national
comment
element
magic
lion
sand
crust
toast
jam
hunter
forest
foraging
silently
tawesomated
joshing
pong
sponge
rubber

12. Choose *Add New Item* from the *Project* menu (or right click in the *Solution Explorer* pan). Select *C++|Utility|Text File (.txt)* as the type and change the name to *input.txt*. Click *Add*.
13. Copy the words above into the file and on a new line at the beginning of the file add the number of words: 100. Save the file.
14. Open the *Project Properties* dialog (accessed from the *Project* menu). Go to Configuration *Properties|Debugging* and add *<"$(ProjectDir)input.txt"* to the *Command Arguments* field. Click *Apply*.
15. Now run your program again.

You should see that the program now runs without any input from you. The < redirects standard input so that it comes from the file, instead of from the keyboard.

Quicksort

Next we are going to write a quicksort function for strings. The function prototype will be:

```
void quicksort(String *a, int first, int last);
```

a is the array of strings, *first* is the first index to be sorted and *last* is the last index to be sorted. Initially the indices will be 0 and n-1 for an n element array.

16. Implement the quicksort function using the following *pseudo-code* as a guide (remember that you will need to use the *strcmp* function in your implementation):

```
quicksort(a,first,last):
      if (first<last):
            // use last element as a pivot
            i=first; j=last-1
            while i<j:
                  while i<last and a[i]<a[last]:
                        i++
                  while j>=first and a[j]>a[last]:
                        j--
                  if i<j:
                        swap(a[i],a[j])
            swap(a[i],a[last])
            quicksort(a,first,j)
            quicksort(a,i+1,last)
```

17. Modify you main method so that the quicksort function is called after reading the strings and before printing them. Check that the result is correct.
18. Now modify your code so that it uses the median-of-three method for choosing a pivot.
19. If you get time, modify your code so that it switches to insertion sort when the number of elements is less than some threshold.