Programming Project

Terry Marris January 2010

4 Utility

We look at some utility functions that are used throughout the project.

4.1 utility.h

Perhaps the three most useful functions are *getStr()*, *getInt()* and *intToString()*.

getStr() displays a prompt and returns a line of text entered at the keyboard. The line is constrained by the given size.

getInt() displays a prompt and returns an integer.

intToString() returns the given integer as a string.

```
/* utility.h: specifies some useful function prototypes */
/* ver 1.0 1Jan2011 */

#ifndef UTILITY_H
#define UTILITY_H

/* fgetLine: reads a line of text up to the given size from the file,
   returns the line */
char *fgetLine(FILE *file, int size);

/* getStr: returns a string up to the given size from the keyboard */
char *getStr(char *prompt, int size);

/* getInt: returns the integer entered at the keyboard */
int getInt(char *prompt);

/* getDouble: returns the number of type double entered
   at the keyboard */
double getDouble(char *prompt);

/* stringDuplicate: returns a duplicate copy of string */
char *stringDuplicate(const char *string);

/* reverseString: reverses the contents of string */
char *reverseString(const char *string);

/* intToString: converts an integer to a string */
char *intToString(int n);

/* equalStringsIgnoreCase: returns 1 if str1 and str2 have identical
   elements, otherwise returns 0 irrespective of case */
int equalStringsIgnoreCase(const char *str1, const char* str2);
```

```
/* compareStringsIgnoreCase: returns -1 if str1 < str2,
   0 if str1 == str2, +1 if str1 > str2 in alphabetical order */
int compareStringsIgnoreCase(const char *str1, const char *str2);

/* stringToLowerCase: returns lower case version of string parameter
*/
char *stringToLowerCase(const char *string);

/* stringToUpperCase: returns upper case version of string parameter
*/
char *stringToUpperCase(const char *string);

/* doubleEquals: returns 1 if a and b are close enough to equal,
    (as determined by tolerance) otherwise returns 0 */
int doubleEquals(double a, double b, double tolerance);

#endif
```

4.2 utility.c

The basis of the functions that return keyboard input is *fgetLine()*. *fgetLine()* returns a line of text, of length up to the given size, from the given file.

```
/* utility.c: specifies some useful functions */
/* ver 1.0 1Jan2011 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include "utility.h"

/* fgetLine: reads a line of text from the given file, returns its
length */
char *fgetLine(FILE *file, int size)
{
  char *line = malloc(size + 1);
  int c, i;

  i = 0;
  while ((c = getc(file)) != EOF) {
    if (c == '\n')
      break;
    if (i < size - 1) {
      line[i] = c;
      i++;
    }
  }
  line[i] = '\0';
  return line;
}

/* getStr: returns a string up to the given size from the keyboard */
char *getStr(char *prompt, int size)
{
  printf("%s ", prompt);
  return fgetLine(stdin, size);
}
```

```
/* getInt: returns the integer entered at the keyboard */
int getInt(char *prompt)
{
  char *s;

  s = getStr(prompt, 10);
  return atoi(s);
}

/* getDouble: returns the number of type double entered
   at the keyboard */
double getDouble(char *prompt)
{
  char *s;

  s = getStr(prompt, 308);
  return atof(s);
}

/* stringDuplicate: returns a duplicate copy of string */
char *stringDuplicate(const char *string)
{
  char *copy;
  int i;

  copy = malloc(strlen(string) + 1);  /* +1 for '\0' */
  for (i = 0; string[i] != '\0'; i++)
    copy[i] = string[i];
  copy[i] = '\0';
  return copy;
}

/* reverseString: reverses the contents of string */
char *reverseString(const char *string)
{
  char *str = malloc(strlen(string) + 1);
  int i, j;

  j = strlen(string) - 1;
  for (i = 0; string[i] != '\0'; i++) {
    str[j] = string[i];
    j--;
  }
  str[i] = '\0';
  return str;
}

/* intToString: converts an integer to a string */
char *intToString(int n)
{
  int i, sign;
  char *string = malloc(12);

  sign = n;
  if (n < 0)
    n = -n;
  i = 0;
```

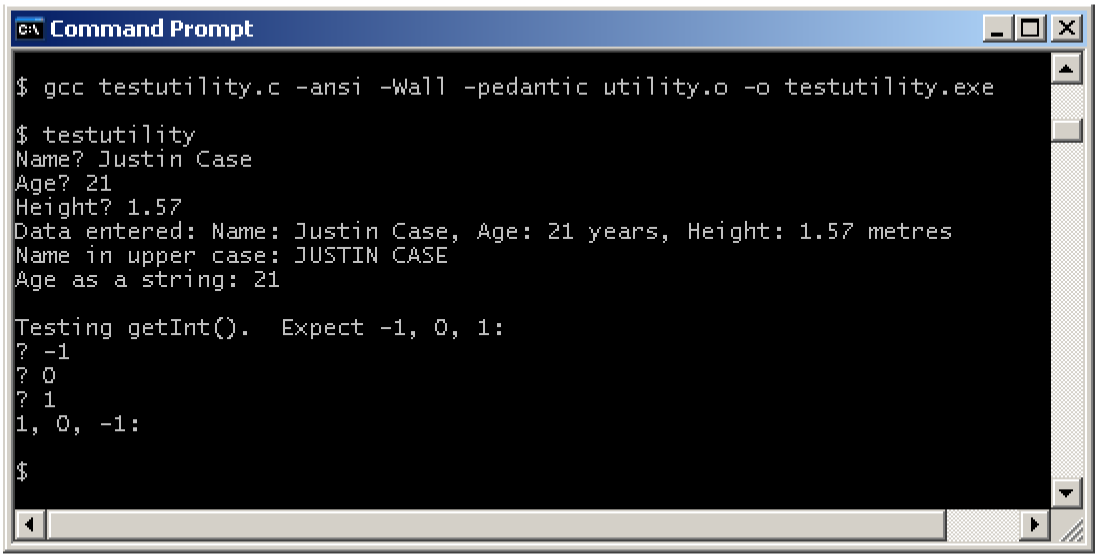
```
  do {
    string[i] = n % 10 + '0';
    i++;
    n = n / 10;
  } while (n > 0);
  if (sign < 0) {
    string[i] = '-';
    i++;
  }
  string[i] = '\0';
  return reverseString(string);
}

/* stringToLowerCase: returns lower case version of string parameter
*/
char *stringToLowerCase(const char *string)
{
  char *lcString = malloc(strlen(string) + 1); /* +1 for '\0' */
  int i;

  for (i = 0; string[i] != '\0'; i++)
    lcString[i] = string[i];
  lcString[i] = '\0';
  for (i = 0; lcString[i] != '\0'; i++)
    lcString[i] = tolower(lcString[i]);
   return lcString;
}

/* stringToUpperCase: returns upper case version of string parameter
*/
char *stringToUpperCase(const char *string)
{
  char *ucString = malloc(strlen(string) + 1); /* +1 for '\0' */
  int i;

  for (i = 0; string[i] != '\0'; i++)
    ucString[i] = string[i];
  ucString[i] = '\0';
  for (i = 0; ucString[i] != '\0'; i++)
    ucString[i] = toupper(ucString[i]);

  return ucString;
}

/* equalStringsIgnoreCase: returns 1 if str1 and str2 have identical
elements, otherwise returns 0 irrespective of case */
int equalStringsIgnoreCase(const char *str1, const char* str2)
{
  int i;
  char *s1 = stringToLowerCase(str1);
  char *s2 = stringToLowerCase(str2);

  for (i = 0; s1[i] != '\0'; i++)
    if (s1[i] != s2[i])
      return 0;
  return s1[i] == s2[i];
}
```

```
/* compareStringsIgnoreCase: returns -1 if str1 < str2,
   0 if str1 == str2, +1 if str1 > str2 in alphabetical order */
int compareStringsIgnoreCase(const char *str1, const char *str2)
{
  int i;
  char *s1 = stringToLowerCase(str1);
  char *s2 = stringToLowerCase(str2);

  for (i = 0; s1[i] == s2[i]; i++)
    if (s1[i] == '\0')
      return 0;
  if (s1[i] < s2[i])
    return -1;
  else
    return 1;
}

/* doubleEquals: returns 1 if a and b are close enough to equal,
    (as determined by tolerance) otherwise returns 0 */
int doubleEquals(double a, double b, double tolerance)
{
  double difference = fabs(a - b);
  return difference < tolerance;
}
```

4.3 testutility.c

Testing is limited. Should really test every function exhaustively, but ...

```
/* testutility.c */
/* ver 1.0 1Jan2011 */

#include <stdio.h>
#include "utility.h"

int main()
{
  char *name;
  int age;
  double height;

  name = getStr("Name?", 15);
  age = getInt("Age?");
  height = getDouble("Height?");
  printf("Data entered: Name: %s, Age: %d years, Height: %0.2f metres\n",
          name, age, height);
  printf("Name in upper case: %s\n", stringToUpperCase(name));
  printf("Age as a string: %s\n", intToString(age));
  printf("\n");

  printf("Testing getInt().  Expect -1, 0, 1: \n");
  printf("%d, %d, %d:\n", getInt("?"), getInt("?"), getInt("?"));

  return 0;
}
```



Bibliography

Kernighan B & Ritchie D *The C Programming Language* Prentice Hall 1988 pp 64, 69