```
    1: #include <stdlib.h>
    2: #include <stdio.h>
    3: #include <ctype.h>
    4: #include <string.h>
    5: #include "util.h"
    6:
    7:
    8: //strip trailing whitespace and ctrl ch
aracters
    9: char * CleanStringTR(char * sz)
   10: {
   11:   //Easy. Just start at the end, and wo
rk back to first printing character.
   12:     char * p = sz;
   13:     //find the null
   14:     while(*p)++p;
   15:     //work backwards
   16:     while(p > sz)
   17:     {
   18:         if(!isgraph(*p)) *(p--) = 0;
   19:         else break;
   20:     }
   21:     return sz;
   22: }
   23:
   24:
   25: //Remove padding, CR, DESTRUCTIVE
   26: char * CleanStringCR(char * sz)
   27: {
   28:     //This modifies the input string by
 removing any
   29:     //leading or trailing whitespace, C
R
   30:     char * p = sz; //Pointer to first c
har
   31:     char * q = sz; //Insertion point
   32:     int leadspace = 1; // Am I in leadi
ng whitepace?
   33:     while(*p)// While I haven't reached
 the null
   34:     {
   35:       if(!isgraph(*p)) //non-printing
   36:       {
   37:           if(isspace(*p))
   38:           {
   39:               if(leadspace)
   40:                   p++; //drop leading w
hitespace
   41:               else
   42:               {
   43:                   //Replace all whiteps
ace with spaces
   44:                   *q=' ';
   45:                   ++q;
   46:                   ++p;
   47:               }
   48:           }
   49:       }
   50:       else
   51:       {
   52:           //Done with any lead space
   53:           leadspace = 0;
   54:           *q = *p;
   55:           ++p;
   56:           ++q;
   57:       }
   58:     }
   59:     *q = *p;//Copy final null.
   60:
   61:     //Get rid of trailing whitespace
   62:     sz = CleanStringTR(sz);
   63:     return sz; // Return modified strin
g
   64: }
   65:
   66: //Remove padding, convert to lower case
 DESTRUCTIVE
   67: char * CleanStringLC(char * sz)
   68: {
   69:     //This modifies the input string by
 removing any
   70:     //leading or trailing whitespace an
d converting
   71:     //all uppercase characters to lower
case
   72:     sz = CleanStringCR(sz);
   73:     char * p = sz; //Pointer to first c
har
   74:     char * q = sz; //Insertion point
   75:     while(*p)// While I haven't reached
 the null
   76:     {
   77:         if(isspace(*p) || iscntrl(*p))
//whitespace
   78:         {
   79:             p++; //Ignore
   80:         }
   81:         else
   82:         {//tolower doesn't change non-u
ppercase
   83:             *q = tolower(*p);
   84:             ++p;
   85:             ++q;
   86:         }
   87:     }
   88:     *q = *p;//Copy final null.
   89:     return sz; // Return modified strin
g
   90: }
   91:
   92: //Get an int from user
   93: int GetInt(char * prompt)
   94: {
   95:     int gotanint = 0;
   96:     int value = 0;
   97:     char * pos = NULL; //Pointer to loc
ation in buffer
   98:     char * buffer = (char *)malloc(INPU
TBUFFSIZE);
   99:     if (!buffer)// DMA Failure
  100:     {
  101:         fprintf(stderr, "Error: DMA Fai
```

```
lure.\n");
  102: #ifdef _DEBUG
  103:          fflush(stdin); //Make sure ther
e's nothing lurking in the buffer.
  104:          printf("Press Enter to Exit");
  105:          fgetc(stdin);
  106: #endif
  107:          exit(EXIT_FAILURE);
  108:      }
  109:
  110:      do
  111:      {
  112:          printf("%s ",prompt);
  113:          fflush(stdout);
  114:          fflush(stdin);
  115:          if (!fgets(buffer, INPUTBUFFSIZ
E, stdin)) // Error talking to user
  116:          {
  117:              fprintf(stderr, "Error: Una
ble to get value from user.\n");
  118:              exit(EXIT_FAILURE);
  119: #ifdef _DEBUG
  120:              fflush(stdin); //Make sure
there's nothing lurking in the buffer.
  121:              printf("Press Enter to Exit
");
  122:              fgetc(stdin);
  123: #endif
  124:          }
  125:          value = strtol(buffer, &pos, 10
); // Expect base 10.  pos at final
  126:          //Was what we got an integer?
  127:          if (buffer[0] != '\n' && (*pos
== '\n' || *pos == '\0'))
  128:              gotanint = 1; //looks good
  129:      } while (!gotanint);
  130:      //Free the memory
  131:      free(buffer);
  132:      return value;
  133: }
  134:
  135:
  136: //Get a y or n response from user.  Y r
eturns 1, else 0
  137: //Will keep asking until it gets someth
ing intelligible.
  138: int GetYorN(char * prompt)
  139: {
  140:      int gotResponse = 0;
  141:      int retval = 0;
  142:      char * buffer = (char *)malloc(INPU
TBUFFSIZE);
  143:      if (!buffer)// DMA Failure
  144:      {
  145:          fprintf(stderr, "Error: DMA Fai
lure.\n");
  146: #ifdef _DEBUG
  147:          fflush(stdin); //Make sure ther
e's nothing lurking in the buffer.
  148:          printf("Press Enter to Exit");
  149:          fgetc(stdin);
  150:          exit(EXIT_FAILURE);
  151: #endif
  152:      }
  153:
  154:      do
  155:      {
  156:          printf("%s ",prompt);
  157:          fflush(stdout);
  158:          fflush(stdin);
  159:          if (!fgets(buffer, INPUTBUFFSIZ
E, stdin)) // Error talking to user
  160:          {
  161:              fprintf(stderr, "Error: Una
ble to get value from user.\n");
  162: #ifdef _DEBUG
  163:              fflush(stdin); //Make sure
there's nothing lurking in the buffer.
  164:              printf("Press Enter to Exit
");
  165:              fgetc(stdin);
  166:              exit(EXIT_FAILURE);
  167: #endif
  168:          }
  169:          //We now have a response from t
he user.  Is it what we're looking for?
  170:          buffer = CleanStringLC(buffer);
 //Convert to lcase, strip whitespace
  171:
  172:          if(!strcmp(buffer,"y") || !strc
mp(buffer,"yes"))
  173:          {
  174:              gotResponse = 1;
  175:              retval = 1;
  176:          }
  177:          else if(!strcmp(buffer,"n") ||
!strcmp(buffer,"no"))
  178:          {
  179:              gotResponse = 1;
  180:              retval = 0;
  181:          }
  182:          else gotResponse = 0;
  183:      } while (!gotResponse);
  184:      //Free the memory
  185:      free(buffer);
  186:      return retval;
  187: }
  188:
  189:
  190: //Get a string from the user after a pr
ompt
  191: char * GetString(char * prompt, char *
buffer, int buffersize)
  192: {
  193:      //In this case, we will not acquire
 our own temporary buffer
  194:      //but use the one passed to us by t
he calling scope.
  195:      if (!buffer)// bad buffer location
  196:      {
  197:          fprintf(stderr, "Error: Buffer
is a Null Pointer.\n");
```

```
  198: #ifdef _DEBUG
  199:         fflush(stdin); //Make sure ther
e's nothing lurking in the buffer.
  200:         printf("Press Enter to Exit");
  201:         fgetc(stdin);
  202:         exit(EXIT_FAILURE);
  203: #endif
  204:     }
  205:
  206:     printf("%s ",prompt);
  207:     fflush(stdout);
  208:     fflush(stdin);
  209:     if (!fgets(buffer, buffersize, stdi
n)) // Error talking to user
  210:     {
  211:         fprintf(stderr, "Error: Unable
to get value from user.\n");
  212: #ifdef _DEBUG
  213:         fflush(stdin); //Make sure ther
e's nothing lurking in the buffer.
  214:         printf("Press Enter to Exit");
  215:         fgetc(stdin);
  216:         exit(EXIT_FAILURE);
  217: #endif
  218:     }
  219:     return CleanStringCR(buffer);
  220: }
```