

```
1. /**
2.  * copy-2.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Copies a string.
8.  *
9.  * Demonstrates pointer arithmetic.
10. */
11.
12. #include <cs50.h>
13. #include <ctype.h>
14. #include <stdio.h>
15. #include <string.h>
16.
17. int main(void)
18. {
19.     // get line of text
20.     printf("Say something: ");
21.     char* s = GetString();
22.     if (s == NULL)
23.     {
24.         return 1;
25.     }
26.
27.     // allocate enough space for copy
28.     char* t = malloc((strlen(s) + 1) * sizeof(char));
29.     if (t == NULL)
30.     {
31.         return 1;
32.     }
33.
34.     // copy string, including '\0' at end
35.     for (int i = 0, n = strlen(s); i <= n; i++)
36.     {
37.         *(t + i) = *(s + i);
38.     }
39.
40.     // change copy
41.     printf("Capitalizing copy...\n");
42.     if (strlen(t) > 0)
43.     {
44.         *t = toupper(*t);
45.     }
46.
47.     // print original and copy
48.     printf("Original: %s\n", s);
```

```
49.     printf("Copy:   %s\n", t);
50.
51.     // success
52.     return 0;
53. }
```

```
1. /*****
2.  * CS50 Library 5
3.  * https://manual.cs50.net/library/
4.  *
5.  * Based on Eric Roberts' genlib.c and simpio.c.
6.  *
7.  * Copyright (c) 2013,
8.  * Glenn Holloway <holloway@eecs.harvard.edu>
9.  * David J. Malan <malan@harvard.edu>
10. * All rights reserved.
11. *
12. * BSD 3-Clause License
13. * http://www.opensource.org/licenses/BSD-3-Clause
14. *
15. * Redistribution and use in source and binary forms, with or without
16. * modification, are permitted provided that the following conditions are
17. * met:
18. *
19. * * Redistributions of source code must retain the above copyright notice,
20. * * this list of conditions and the following disclaimer.
21. * * Redistributions in binary form must reproduce the above copyright
22. * * notice, this list of conditions and the following disclaimer in the
23. * * documentation and/or other materials provided with the distribution.
24. * * Neither the name of CS50 nor the names of its contributors may be used
25. * * to endorse or promote products derived from this software without
26. * * specific prior written permission.
27. *
28. * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
29. * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
30. * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
31. * PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
32. * HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
33. * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
34. * TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
35. * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
36. * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
37. * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
38. * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
39. *****/
40.
41. #include <stdio.h>
42. #include <stdlib.h>
43. #include <string.h>
44.
45. #include "cs50.h"
46.
47. /**
48.  * Reads a line of text from standard input and returns the equivalent
```

```
49.  * char; if text does not represent a char, user is prompted to retry.
50.  * Leading and trailing whitespace is ignored. If line can't be read,
51.  * returns CHAR_MAX.
52.  */
53.  char GetChar(void)
54.  {
55.      // try to get a char from user
56.      while (true)
57.      {
58.          // get line of text, returning CHAR_MAX on failure
59.          string line = GetString();
60.          if (line == NULL)
61.          {
62.              return CHAR_MAX;
63.          }
64.
65.          // return a char if only a char (possibly with
66.          // leading and/or trailing whitespace) was provided
67.          char c1, c2;
68.          if (sscanf(line, " %c %c", &c1, &c2) == 1)
69.          {
70.              free(line);
71.              return c1;
72.          }
73.          else
74.          {
75.              free(line);
76.              printf("Retry: ");
77.          }
78.      }
79.  }
80.
81.  /**
82.  * Reads a line of text from standard input and returns the equivalent
83.  * double as precisely as possible; if text does not represent a
84.  * double, user is prompted to retry. Leading and trailing whitespace
85.  * is ignored. For simplicity, overflow and underflow are not detected.
86.  * If line can't be read, returns DBL_MAX.
87.  */
88.  double GetDouble(void)
89.  {
90.      // try to get a double from user
91.      while (true)
92.      {
93.          // get line of text, returning DBL_MAX on failure
94.          string line = GetString();
95.          if (line == NULL)
96.          {
```

```
97.         return DBL_MAX;
98.     }
99.
100.    // return a double if only a double (possibly with
101.    // leading and/or trailing whitespace) was provided
102.    double d; char c;
103.    if (sscanf(line, " %lf %c", &d, &c) == 1)
104.    {
105.        free(line);
106.        return d;
107.    }
108.    else
109.    {
110.        free(line);
111.        printf("Retry: ");
112.    }
113. }
114. }
115.
116. /**
117.  * Reads a line of text from standard input and returns the equivalent
118.  * float as precisely as possible; if text does not represent a float,
119.  * user is prompted to retry. Leading and trailing whitespace is ignored.
120.  * For simplicity, overflow and underflow are not detected. If line can't
121.  * be read, returns FLT_MAX.
122.  */
123. float GetFloat(void)
124. {
125.     // try to get a float from user
126.     while (true)
127.     {
128.         // get line of text, returning FLT_MAX on failure
129.         string line = GetString();
130.         if (line == NULL)
131.         {
132.             return FLT_MAX;
133.         }
134.
135.         // return a float if only a float (possibly with
136.         // leading and/or trailing whitespace) was provided
137.         char c; float f;
138.         if (sscanf(line, " %f %c", &f, &c) == 1)
139.         {
140.             free(line);
141.             return f;
142.         }
143.         else
144.         {
```

```
145.         free(line);
146.         printf("Retry: ");
147.     }
148. }
149. }
150.
151. /**
152.  * Reads a line of text from standard input and returns it as an
153.  * int in the range of  $[-2^{31} + 1, 2^{31} - 2]$ , if possible; if text
154.  * does not represent such an int, user is prompted to retry. Leading
155.  * and trailing whitespace is ignored. For simplicity, overflow is not
156.  * detected. If line can't be read, returns INT_MAX.
157.  */
158. int GetInt(void)
159. {
160.     // try to get an int from user
161.     while (true)
162.     {
163.         // get line of text, returning INT_MAX on failure
164.         string line = GetString();
165.         if (line == NULL)
166.         {
167.             return INT_MAX;
168.         }
169.
170.         // return an int if only an int (possibly with
171.         // leading and/or trailing whitespace) was provided
172.         int n; char c;
173.         if (sscanf(line, " %d %c", &n, &c) == 1)
174.         {
175.             free(line);
176.             return n;
177.         }
178.         else
179.         {
180.             free(line);
181.             printf("Retry: ");
182.         }
183.     }
184. }
185.
186. /**
187.  * Reads a line of text from standard input and returns an equivalent
188.  * long long in the range  $[-2^{63} + 1, 2^{63} - 2]$ , if possible; if text
189.  * does not represent such a long long, user is prompted to retry.
190.  * Leading and trailing whitespace is ignored. For simplicity, overflow
191.  * is not detected. If line can't be read, returns LLONG_MAX.
192.  */
```

```
193.
194. long long GetLongLong(void)
195. {
196.     // try to get a long long from user
197.     while (true)
198.     {
199.         // get line of text, returning LLONG_MAX on failure
200.         string line = GetString();
201.         if (line == NULL)
202.         {
203.             return LLONG_MAX;
204.         }
205.
206.         // return a long long if only a long long (possibly with
207.         // leading and/or trailing whitespace) was provided
208.         long long n; char c;
209.         if (sscanf(line, " %lld %c", &n, &c) == 1)
210.         {
211.             free(line);
212.             return n;
213.         }
214.         else
215.         {
216.             free(line);
217.             printf("Retry: ");
218.         }
219.     }
220. }
221.
222. /**
223.  * Reads a line of text from standard input and returns it as a
224.  * string (char*), sans trailing newline character. (Ergo, if
225.  * user inputs only "\n", returns "" not NULL.) Returns NULL
226.  * upon error or no input whatsoever (i.e., just EOF). Leading
227.  * and trailing whitespace is not ignored. Stores string on heap
228.  * (via malloc); memory must be freed by caller to avoid leak.
229.  */
230. string GetString(void)
231. {
232.     // growable buffer for chars
233.     string buffer = NULL;
234.
235.     // capacity of buffer
236.     unsigned int capacity = 0;
237.
238.     // number of chars actually in buffer
239.     unsigned int n = 0;
240.
```

```
241. // character read or EOF
242. int c;
243.
244. // iteratively get chars from standard input
245. while ((c = fgetc(stdin)) != '\n' && c != EOF)
246. {
247.     // grow buffer if necessary
248.     if (n + 1 > capacity)
249.     {
250.         // determine new capacity: start at 32 then double
251.         if (capacity == 0)
252.         {
253.             capacity = 32;
254.         }
255.         else if (capacity <= (UINT_MAX / 2))
256.         {
257.             capacity *= 2;
258.         }
259.         else
260.         {
261.             free(buffer);
262.             return NULL;
263.         }
264.
265.         // extend buffer's capacity
266.         string temp = realloc(buffer, capacity * sizeof(char));
267.         if (temp == NULL)
268.         {
269.             free(buffer);
270.             return NULL;
271.         }
272.         buffer = temp;
273.     }
274.
275.     // append current character to buffer
276.     buffer[n++] = c;
277. }
278.
279. // return NULL if user provided no input
280. if (n == 0 && c == EOF)
281. {
282.     return NULL;
283. }
284.
285. // minimize buffer
286. string minimal = malloc((n + 1) * sizeof(char));
287. strncpy(minimal, buffer, n);
288. free(buffer);
```

```
289.  
290.     // terminate string  
291.     minimal[n] = '\0';  
292.  
293.     // return string  
294.     return minimal;  
295. }
```

```
1. /*****
2.  * CS50 Library 5
3.  * https://manual.cs50.net/library/
4.  *
5.  * Based on Eric Roberts' genlib.c and simpio.c.
6.  *
7.  * Copyright (c) 2013,
8.  * Glenn Holloway <holloway@eecs.harvard.edu>
9.  * David J. Malan <malan@harvard.edu>
10. * All rights reserved.
11. *
12. * BSD 3-Clause License
13. * http://www.opensource.org/licenses/BSD-3-Clause
14. *
15. * Redistribution and use in source and binary forms, with or without
16. * modification, are permitted provided that the following conditions are
17. * met:
18. *
19. * * Redistributions of source code must retain the above copyright notice,
20. *   this list of conditions and the following disclaimer.
21. * * Redistributions in binary form must reproduce the above copyright
22. *   notice, this list of conditions and the following disclaimer in the
23. *   documentation and/or other materials provided with the distribution.
24. * * Neither the name of CS50 nor the names of its contributors may be used
25. *   to endorse or promote products derived from this software without
26. *   specific prior written permission.
27. *
28. * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
29. * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
30. * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
31. * PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
32. * HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
33. * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
34. * TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
35. * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
36. * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
37. * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
38. * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
39. *****/
40.
41. #ifndef _CS50_H
42. #define _CS50_H
43.
44. #include <float.h>
45. #include <limits.h>
46. #include <stdbool.h>
47. #include <stdlib.h>
48.
```

```
49. /**
50.  * Our own data type for string variables.
51.  */
52. typedef char* string;
53.
54. /**
55.  * Reads a line of text from standard input and returns the equivalent
56.  * char; if text does not represent a char, user is prompted to retry.
57.  * Leading and trailing whitespace is ignored. If line can't be read,
58.  * returns CHAR_MAX.
59.  */
60. char GetChar(void);
61.
62. /**
63.  * Reads a line of text from standard input and returns the equivalent
64.  * double as precisely as possible; if text does not represent a
65.  * double, user is prompted to retry. Leading and trailing whitespace
66.  * is ignored. For simplicity, overflow and underflow are not detected.
67.  * If line can't be read, returns DBL_MAX.
68.  */
69. double GetDouble(void);
70.
71. /**
72.  * Reads a line of text from standard input and returns the equivalent
73.  * float as precisely as possible; if text does not represent a float,
74.  * user is prompted to retry. Leading and trailing whitespace is ignored.
75.  * For simplicity, overflow and underflow are not detected. If line can't
76.  * be read, returns FLT_MAX.
77.  */
78. float GetFloat(void);
79.
80. /**
81.  * Reads a line of text from standard input and returns it as an
82.  * int in the range of  $[-2^{31} + 1, 2^{31} - 2]$ , if possible; if text
83.  * does not represent such an int, user is prompted to retry. Leading
84.  * and trailing whitespace is ignored. For simplicity, overflow is not
85.  * detected. If line can't be read, returns INT_MAX.
86.  */
87. int GetInt(void);
88.
89. /**
90.  * Reads a line of text from standard input and returns an equivalent
91.  * long long in the range  $[-2^{63} + 1, 2^{63} - 2]$ , if possible; if text
92.  * does not represent such a long long, user is prompted to retry.
93.  * Leading and trailing whitespace is ignored. For simplicity, overflow
94.  * is not detected. If line can't be read, returns LLONG_MAX.
95.  */
96. long long GetLongLong(void);
```

```
97.  
98. /**  
99.  * Reads a line of text from standard input and returns it as a  
100. * string (char *), sans trailing newline character. (Ergo, if  
101. * user inputs only "\n", returns "" not NULL.) Returns NULL  
102. * upon error or no input whatsoever (i.e., just EOF). Leading  
103. * and trailing whitespace is not ignored. Stores string on heap  
104. * (via malloc); memory must be freed by caller to avoid leak.  
105. */  
106. string GetString(void);  
107.  
108. #endif
```

```
1. /**
2.  * hello-2.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Says hello to whomever.
8.  *
9.  * Demonstrates use of CS50's library and standard input.
10. */
11.
12. #include <cs50.h>
13. #include <stdio.h>
14.
15. int main(void)
16. {
17.     printf("State your name: ");
18.     string name = GetString();
19.     printf("hello, %s\n", name);
20. }
```