

```
1: // $Id: numlist.c,v 1.6 2014-02-07 17:13:33-08 - - $
2:
3: //
4: // Demo of how to use malloc and free.
5: //
6:
7: #include <assert.h>
8: #include <libgen.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11:
12: //
13: // Declare the type of the handle, or pointer, to the struct.
14: // In Java, the same name is used for both the handle and the
15: // struct.
16: //
17: // Declare the type of the node. This is much like Java, except
18: // that the word "struct" is used. C does not allow functions
19: // to be declared inside structs, as does Java.
20: //
21: typedef struct node node;
22: struct node {
23:     double item;
24:     node *link;
25: };
26:
27: //
28: // The main program allocates some nodes, pushes them onto a list,
29: // prints them out, and then frees up the nodes.
30: //
31: int main (int argc, char **argv) {
32:     char *programe = basename (argv[0]);
33:
34:     //
35:     // Declare and set the head of the list to NULL.
36:     //
37:
38:     node *head = NULL;
39:
40:     //
41:     // Loop, pushing some random numbers onto the list. Note that
42:     // '->' in C means '.' in Java. Malloc(3c) is used to allocate
43:     // storage, like 'new' in Java. Always check with 'assert' that
44:     // malloc has actually returned the address of new memory.
45:     // 'sizeof' returns the number of bytes necessary for its
46:     // argument.
47:     //
48:     int max = argc < 2 ? 10 : atoi (argv[1]);
49:     printf ("%s: looping %d times\n", programe, max);
50:     for (int count = 0; count < max; ++count) {
51:         node *tmp = malloc (sizeof (struct node));
52:         assert (tmp != NULL);
53:         tmp->item = drand48() * 1e6;
54:         tmp->link = head;
55:         head = tmp;
56:     }
```

```
57:
58:  //
59:  // Loop down the list, printing out each entry in debug mode.
60:  //
61:  printf ("%p\n", &head);
62:  printf ("head= %p\n", head);
63:  for (node *curr = head; curr != NULL; curr = curr->link) {
64:      printf ("%p -> struct node {item= %.15g, link= %p}\n",
65:              curr, curr->item, curr->link);
66:  }
67:  printf ("NULL= %p\n", NULL);
68:
69:  //
70:  // Free up all of the nodes.
71:  //
72:  while (head != NULL) {
73:      node *old = head;
74:      head = head->link;
75:      free (old);
76:  }
77:
78:  //
79:  // Deliberately cause some memory leaks and throw away result.
80:  //
81:  for (int leaks = 0; leaks < 4; ++leaks) malloc (256);
82:  malloc (4096);
83:
84:  return EXIT_SUCCESS;
85: }
86:
87: /*
88: //TEST// valgrind --leak-check=full --log-file=numlist.lisval \
89: //TEST//      ./numlist >numlist.lisout 2>&1
90: //TEST// mkpspdf numlist.ps numlist.c* numlist.lis*
91: */
92:
```

[illegible]

```
1: numlist: looping 10 times
2: &head= 0xffefff500
3: head= 0x9c9a360
4: 0x9c9a360 -> struct node {item= 454433.423738244, link= 0x9c9a310}
5: 0x9c9a310 -> struct node {item= 526750.279762108, link= 0x9c9a2c0}
6: 0x9c9a2c0 -> struct node {item= 487217.223946828, link= 0x9c9a270}
7: 0x9c9a270 -> struct node {item= 92297.6476986754, link= 0x9c9a220}
8: 0x9c9a220 -> struct node {item= 91330.6121122943, link= 0x9c9a1d0}
9: 0x9c9a1d0 -> struct node {item= 364602.248390607, link= 0x9c9a180}
10: 0x9c9a180 -> struct node {item= 176642.642542916, link= 0x9c9a130}
11: 0x9c9a130 -> struct node {item= 41631.0015946131, link= 0x9c9a0e0}
12: 0x9c9a0e0 -> struct node {item= 985.394674650308, link= 0x9c9a090}
13: 0x9c9a090 -> struct node {item= 3.90798504668055e-08, link= (nil)}
14: NULL= (nil)
```

```
1: ==26755== Memcheck, a memory error detector
2: ==26755== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
3: ==26755== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright
info
4: ==26755== Command: ./numlist
5: ==26755== Parent PID: 26754
6: ==26755==
7: ==26755==
8: ==26755== HEAP SUMMARY:
9: ==26755==      in use at exit: 5,120 bytes in 5 blocks
10: ==26755==    total heap usage: 16 allocs, 11 frees, 5,296 bytes allocated
11: ==26755==
12: ==26755== 1,024 bytes in 4 blocks are definitely lost in loss record 1 o
f 2
13: ==26755==    at 0x4C29BBD: malloc (in /opt/rh/devtoolset-3/root/usr/lib6
4/valgrind/vgpreload_memcheck-amd64-linux.so)
14: ==26755==    by 0x400B4C: main (numlist.c:81)
15: ==26755==
16: ==26755== 4,096 bytes in 1 blocks are definitely lost in loss record 2 o
f 2
17: ==26755==    at 0x4C29BBD: malloc (in /opt/rh/devtoolset-3/root/usr/lib6
4/valgrind/vgpreload_memcheck-amd64-linux.so)
18: ==26755==    by 0x400B60: main (numlist.c:82)
19: ==26755==
20: ==26755== LEAK SUMMARY:
21: ==26755==    definitely lost: 5,120 bytes in 5 blocks
22: ==26755==    indirectly lost: 0 bytes in 0 blocks
23: ==26755==    possibly lost: 0 bytes in 0 blocks
24: ==26755==    still reachable: 0 bytes in 0 blocks
25: ==26755==    suppressed: 0 bytes in 0 blocks
26: ==26755==
27: ==26755== For counts of detected and suppressed errors, rerun with: -v
28: ==26755== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 1 from 1)
```