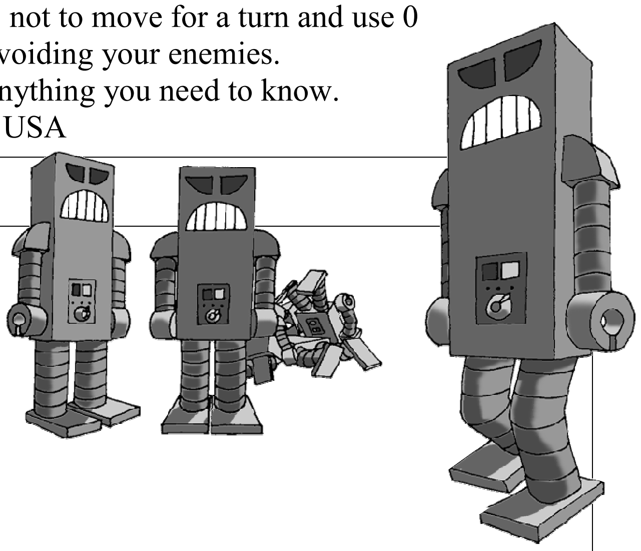
ROBOTESCAPE.C

Robots are chasing you and you have nothing but your wits to avoid death with. You are the 'O', you are being pursued by the '+' and if two of them collide they form a pile of debris that looks like '#'. The game is designed to be controlled with a standard number pad. Press 5 not to move for a turn and use 0 (zero) to activate your teleport-a-matic, your one weapon in avoiding your enemies.

The listing contains instructions that pretty much covers anything you need to know.

Robot Escape is written by Joseph Larson of Aurora, CO, USA



ROBOTESCAPE.C	You will need: a C/C++ complier.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <ctype.h>
#define W 77
#define H 21
#define MAX_LOC  W * H
#define SQ(a) (a) * (a)
typedef struct {int loc, type;} STUFF;
STUFF **p;
int numobj;
char msg[W], icons[4] = {' ', '#', 'O', '+'};

void intro (void) {
  printf ("Robot Escape\n------------\n"
  "You are being chased by robots. Fortunately you have two advantages. First,\n"
  "your pursuers aren't very smart. Second, you have a teleport-a-matic at\n"
  "your command should you need it.\n\n"
  "The robots will take the most direct route to you, even if it means running\n"
  "into another robot. When two robots crash they make a debris heap that you\n"
  "can crash other robots into. When all robots are destroyed you win and go\n"
  "on to the next level. Try to make it to level 10!\n\n"
  "If you have to use your teleport-a-matic remember that it might teleport\n"
  "you into your enemies, so use it sparingly.\n\n"
  "You move using the number keys. (Be sure num-lock is on) Use the 5 key if\n"
  "you don't want to move for a turn. Use the 0 key to turn on your\n"
  "teleport-a-matic. And if you don't need to move for the rest of the level\n"
  "type a period ('.') for your move. Otherwise 8 is up, 2 is down, 4 is left\n"
  "and 6 is right and the corner keys are diagonal.\n\n"
  "\nGood luck! Press any the ENTER key start.\n\n");
  getchar ();
}

int setup (int lv) {
  int n, numleft;
  numobj = numleft = 3 + lv * 4;
  p = (STUFF**) malloc ((numobj + 1) * sizeof (STUFF**));
  n = rand () % numleft;
  while (numleft >= 0) {
    p[numleft] = (STUFF*) malloc (sizeof (STUFF));
    p[numleft]->loc = (numleft - numobj) ?
      numleft * (MAX_LOC / numobj) + rand() % (MAX_LOC / numobj) : MAX_LOC + W;
    p[numleft]->type = (n != numleft--) + 2;
  }
  return n;
}
```

Continued on page 2...

```
void takedown (void) {
  int n;
  for (n = 0; n < numobj; n++) free (p[n]);
  free (p);
}
int draw_floor (int l) {
  int c, d, n, s = 1;
  STUFF *temp;
  putchar ('\n'); puts (msg);
  putchar ('-'); for (c = W; c; c--) putchar ('-');
  for (d = 0; d <= numobj; d++) {
    for (n = numobj; d < numobj && n > d; n--)
      if (p[n]->loc < p[n - 1]->loc) {
        temp = p[n]; p[n] = p[n - 1]; p[n - 1] = temp;
      }
    if (d) {
      if (p[d]->loc == p[d - 1]->loc) {p[d]->type = p[d - 1]->type = 1;}
      else do {
        if (!(c % W)) printf ("|\n|");
        putchar (icons[(p[d - 1]->loc == c) ? p[d - 1]->type : 0]);
      } while (++c < p[d]->loc && c < MAX_LOC);
      if (p[d - 1]->type == 3) s = 0;
    }
  }
  printf ("|\n|"); for (c = W; c; c--) putchar ('-'); printf ("-\n");
  sprintf (msg, "Level %d", l);
  return s;
}

void gorobo (STUFF *pl) {
  int n;
  for (n = 0; n < numobj; n++) if (p[n]->type == 3) {
    if ((p[n]->loc % W) > (pl->loc % W)) p[n]->loc -= 1;
  else if ((p[n]->loc % W) < (pl->loc % W)) p[n]->loc += 1;
    if ((p[n]->loc / W) > (pl->loc / W)) p[n]->loc -= W;
  else if ((p[n]->loc / W) < (pl->loc / W)) p[n]->loc += W;
  }
}

int main (int argc, char *argv[]) {
  int n, m, w, level, d[9];
  char in;
  STUFF *pl;
  srand (time (NULL));
  intro ();
  for (n = 0; n < 9; n++) d[n] = W * (1 - n / 3) + (n % 3 - 1);
  level = (--argc) ? (atoi (argv[1]) - 1) : 0;
  sprintf (msg, "Game Start!");
  do {
    pl = p[setup (++level)]; m = 1;
    draw_floor (level);
    do {
      printf ("Move : ");
      if (m) {
        do in = getchar (); while (iscntrl (in));
        if (in == '.') m = 0;
        else if (!(in - '0')) {
          gorobo (pl); pl->loc = rand () % MAX_LOC;
```

Continued on page 3...

ROBOTESCAPE.C Listing continued from page 2...

```
     sprintf (msg, "Stunned... Press ENTER"); draw_floor (level);
     getchar (); getchar ();
     gorobo (pl);
    } else if (in > '0' && in <= '9') {
     n = pl->loc + d[in - '1'];
     if (SQ((n % W) - (pl->loc % W)) < 2 && n < MAX_LOC && n >= 0)
      pl->loc = n;
     else sprintf (msg, "You were stopped by the wall.");
     gorobo(pl);
    } else sprintf (msg, "Invalad entry.");
   } else gorobo(pl);
   w = draw_floor (level);
  } while (!w && pl->type == 2);
  if ((w = pl->type) == 2) sprintf (msg, "level %d complete! ", level);
  takedown ();
 } while (w == 2 && level < 10);
 if (w == 2) printf ("You Win!");
 else printf ("You were captured by the robots at level %d\n", level);
 exit (0);
}
```

Author's Notes:

In this program collision and screen drawing are done at the same time with a modified bubble sort. This works because with every itineration of the bubble sort you can assume a certain amount of the list is already sorted.

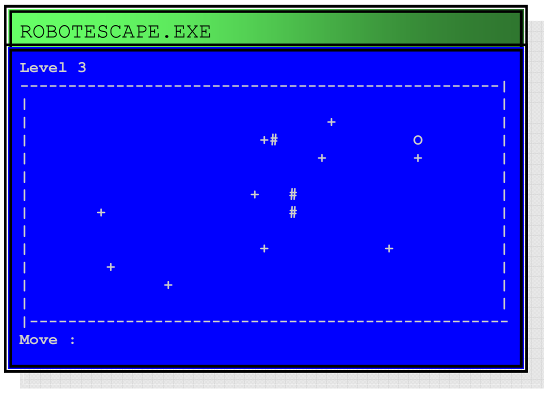
If level 10 becomes no challenge you can remove the winning condition that depends on the level and play until you're killed. This would mean changing the line near the end that reads

```
} while (w == 2 && level < 10);
```

...to

```
} while (w == 2);
```

After that you can play for as long as you don't die. Technically this eliminates any chance of winning, but it can be fun.



```
Level 3
--------------------------------------------------|
|                                                 |
|                               +                 |
|                        +#              o        |
|                              +         +        |
|                                                 |
|                       +   #                     |
|       +                   #                     |
|                                                 |
|                        +            +           |
|        +                                        |
|              +                                  |
|                                                 |
|--------------------------------------------------
Move :
```