BOMBARDMENT

The rules are simple. You have 4 bases that you can put in 25 spaces, the computer has 4 bases they can put in 25 other spaces. Then players alternate turns trying to shoot at their enemies bases. The first one to find the bases of their enemy wins.

Bombardment was inspired by a BASIC program of the same name by Martin Burdash as found in 'BASIC Computer Games' edited by David H. Ahl © 1978.

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

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

float rand_const25;
int num[2], b[2][25];

void showb () {
  int y, p, c;
  char token[2][5] = {{".PX*"}, {"..X*"}};

  printf ("\n\n%17s%27s\n", "Yours", "Mine");
  for (y = 0; y < 5; y++) {
    for (p = 0; p < 2; p++) {
      for (c = y * 5; c < y * 5 + 5; c++)
        if (token[p][b[p][c]] == '.')
          printf (" %3d", c + 1);
        else printf (" %3c", token[p][b[p][c]]);
      printf ("\t\t");
    }
    printf ("\n\n");
  }
}

void intro() {
  printf("Bombardment\n");
  printf("-----------\n");
  printf("In this game you hide 4 platoons on your our 5×5 grid. Then you try\n"
  "to find where I have hid my 4 by dropping bombs on them as I rain bombs\n"
  "yours.\n\n"
  "The first one to bomb all 4 of their enemy's platoons is the winner.\n"
  "the playfield is numbered like so:\n");
}

void init() {

  rand_const25 = (RAND_MAX + 1) / 25;
  srand(time(NULL));
}

void setcbases() {
  int c, d;

  for (c = 1; c < 5; c++) {
    do {d = (int)(rand() / rand_const25);} while (b[1][d]);
    b[1][d] = 1;
  }
  printf ("My bases are all in place.\n");
}

void sethbases() {
```

Listing continued on page 2...

```
  int c, d;

  for (c = 1; c < 5; c++) {
    showb ();
    printf("Place your platoon #%d : ",c);
    scanf("%d", &d);
    if ((d < 1) || (d > 25)) {
      printf("\nThat is not a valid loaction.\n");
      printf("Choose a location between 1 and 25.\n");
      c--;
    } else if (b[0][d - 1]) {
      printf("\nYou already have a platoon there, choose another location.\n");
      c--;
    } else b[0][d - 1] = 1;
  }
}

void humanmove() {
  int input;

  do {
    printf("\nChoose where you want to fire : ");
    scanf("%d", &input);
    if ((input < 1) || (input > 25))
      printf("\nChoose a location between 1 and 25.");
  } while ((input < 1) || (input > 25));
  if (b[1][input - 1] == 1) {
    b[1][input - 1] = 2;
    switch (--num[1]) {
      case 3 : printf("\nYou found my first platoon.\n"
               "A lucky shot. You'll never find my other three.");
               break;
      case 2 : printf("\nTwo down, two to go. I'm not out yet!");
               break;
      case 1 : printf("\nI'll never give up. You still have one more"
               "platoon to find!");
               break;
      case 0 : showb ();
               printf("\nYou got my last platoon.\n"
               "I've lost, but I'll be back!");
               break;
    }
  } else {b[1][input - 1] = 3; printf("Ha, you missed me.\n");}
}

void computermove() {
  int mark;

  do mark = (int) (rand() / rand_const25); while (b[0][mark] > 1);
  if (b[0][mark]) {
    b[0][mark] = 2;
    printf("\nBoom! I got your platoon at %d.\n"
      "That platoon was number %d.\n", mark + 1, num[0]--);
      } else {
    printf("\nDrat, I missed you. My shot was %d", mark + 1);
    b[0][mark] = 3;
  }
}

void playgame() {
```

Listing continued on page 3...

BOMBARDMENT.C	Listing Continued from page 1….

```
  int c, d;

  for (c = 0; c < 2; c++) for (d = 0; d < 25; d++) b[c][d] = 0;
  for (c = 0; c < 2; c++) num[c] = 4;
  sethbases(); setcbases(); c = 0;
  do {
    showb (); humanmove();
    if (num[1]) computermove();
  } while ((num[0]) && (num[1]));
    showb ();
  if (!num[0]) {
    printf("You Lose! My remaining bases were at :\n");
    for (c = 1; c < 25; c++) if (b[1][c] == 1) printf("%d\n",c + 1);
  } else
    printf("\nYou Win!");
}

int playagain() {
  char input[25];

  printf("\nDo you want to play again? (y\\n)");
  scanf("%s", input);
  if ((input[0] == 'y') || (input[0] == 'Y')) return 1;
  else return 0;
}

int main() {
  intro();
  init();
  do playgame(); while (playagain());
  return (0);
}
```

Author's Notes:

When I first saw this game in BASIC for some reason it struck me as much cooler than it actually was. So I set my mind to the process or rewriting it in C. However, once I started disassembling it I became disappointed with it. The 5x5 table displayed had nothing to do with the game. In fact the data stored in the game was in an 1 dimensional array. It could have just as easily been displayed as a straight line of numbers. I felt deceived. At this I was disillusioned with the game and tho I committed to the recode I never got over that.

Once disenchanted with the game it was easy to find fault. It was not like battleship where one hit could lead to others. Nor was there any strategy more elegant than sequentially hitting numbers, mathematically speaking. In the end winning boils down to pure luck, out guessing the random number generator.

Bur it is salvageable. Combine the two player nature with one of the many hunting game plays like Mug-wump or Hurkle and it would be a considerably more enjoyable experience. The hard part would be making a computer opponent that was enjoyable to play against.