2013年度 卒業論文

プログラミング教育支援システムの改良

大阪産業大学 工学部 情報システム工学科
情報教育システム研究室

10H071 豊田太郎

目次

1 はじめに

ゲームを題材とした学習は、広い分野で利用されている。本研究の先行研究 [1,2] では、レーダー作戦ゲーム [3] を改変した Battle Ship ゲームを題材としてアルゴリズムという概念の理解を補助するためのシステム開発が行われた。

先行研究の Battle Ship ゲームでは、学生が単独で学習する形式となっており、自分の成績を他人と比較したい場合には、プログラムの出力結果を相互に教えあう事でしか比較ができず、その範囲も友人等のごく狭い範囲に限られる。そのため、自身の考案したアルゴリズムが例えば受講生全体の内でどの程度優れているのかといった、広範囲での客観的な評価が難しい。

そこで、この問題を解決するため、より多くの学生とアルゴリズムを比較し、自身のアルゴリズムに対する客観的な評価を得られるシステムを開発した。

第 2 章では本研究の目的について説明する。第 3 章では本研究に用いた Battle Ship ゲームについて説明する。第 4 章では Battle Ship に追加した Compare モードについて説明する、第 5 章では新しく作成したプログラムの機能と動作について説明する。第 6 章では、結果と考察を記述する。第 7 章には研究のまとめ、第 8 章には今後の課題を記述する。

2 研究の目的

本研究の目的は、Battle Ship ゲームを学生の教育に利用する際、全ての学生が共通の基準で各自のアルゴリズムを評価できるようにする事である。それにより、これまで自己評価ないし限られた範囲 (例えば隣席の知り合い同士) との比較でのみ得られた評価から、より客観的な評価が得られるようになると考えられる。

次に、Battle Ship ゲームを利用するのに必要となるが、ゲームの目的と直接関係の無い要素を自動化する事で手間を排し、より簡単にゲームを遊べるようにするシステムの開発を試みた。

2.1 評価基準

アルゴリズムの評価基準には、ゲームの成績を数段階に区切った、度数分布を用いる。偏差値等の相対的な評価基準は、点数記録の蓄積量が少ない時、妥当な評価を得られないと考え、今回は用いなかった。

2.2 コンパイル作業の省略

これまでの Battle Ship ゲームは、アルゴリズムを書き換えた時など、繰り替えしてゲームを利用する際、再度のコンパイルが必要である。様々なアルゴリズムを試行し書き換えが頻繁になると、その手間が煩雑になると考えられるので、その作業を自動化することで手間を無くした。

3 Battle Ship ゲーム

本章では、本研究に用いた Battle Ship ゲーム、ゲームの各モードと、ゲームに使用するアルゴリズムを説明する。

3.1 基本的なルール

元来の Battle Ship ゲーム [4](日本名:レーダー作戦ゲーム [3]、軍艦ゲーム 等) は 2 人用ゲームであり、各自の持つ 10 × 10 マスの盤上に複数の艦船を相手に隠して配置する。1 マスずつ交互に攻撃し、先に相手の艦船が存在するマス全てに命中させることで勝利となる。

先行研究では、敵艦船の配置が自動化され、それに対しての攻撃のみを行う一人用のゲームに改変された。また、攻撃はプログラムにより自動で行われるようにし、プレイヤーはプログラムの中身となるアルゴリズムを記述する。ゲーム終了までに必要とした手数が少なければ少ないほど、より効率的に攻撃を行うアルゴリズムが記述できた事になる。

3.2 各モードの説明

以下に示すのは、先行研究で開発された Battle Ship ゲームに存在する 3 つのモードと、ゲームに使用するアルゴリズムの説明である。

3.2.1 Practice モード

ゲームをプレイしてルールを把握するためのモード。プレイヤーが自分で、攻撃する座標を選択する。

3.2.2 Verification モード

実装したアルゴリズムの挙動を、一手ずつ確認していくモード。

3.2.3 Score モード

実装したアルゴリズムを使ってゲームを 100 回自動実行し、各ゲーム終了までに掛った手数と、その平均攻撃回数を表示するモード。

平均攻撃回数が少ない程、効率的なアルゴリズムを実装できたということになる。

3.3 Battle Ship 用アルゴリズム

初学者向けに動作が抽象化されたマクロが用意されている。プレイヤーはマクロを組み合わせる事でアルゴリズムを作成し、攻撃関数ファイルとして保存する。

4　Compare モード

今回、Battle Ship ゲームの新たなモードとして、Compare モードを追加した。

このモードを実行すると、一度 Score モードを実行した後、その時の平均攻撃回数にログイン名を併記して、外部ファイルに書き込む。

このとき、既に自分の記録が存在する場合、最新の記録と既存の記録を比較し、より優秀な方を書き残す。

最後に、全ての記録の分布を表示する事で、自身の記録が全体の中でどの程度優秀なのかを把握することができる。

Compare モード実行時のプログラムのフローチャートを図 1 に示す。

また、Compare モード実行時に表示される、度数分布の表示を図 2 に示す。この場合の点数とは、平均攻撃回数の事である。

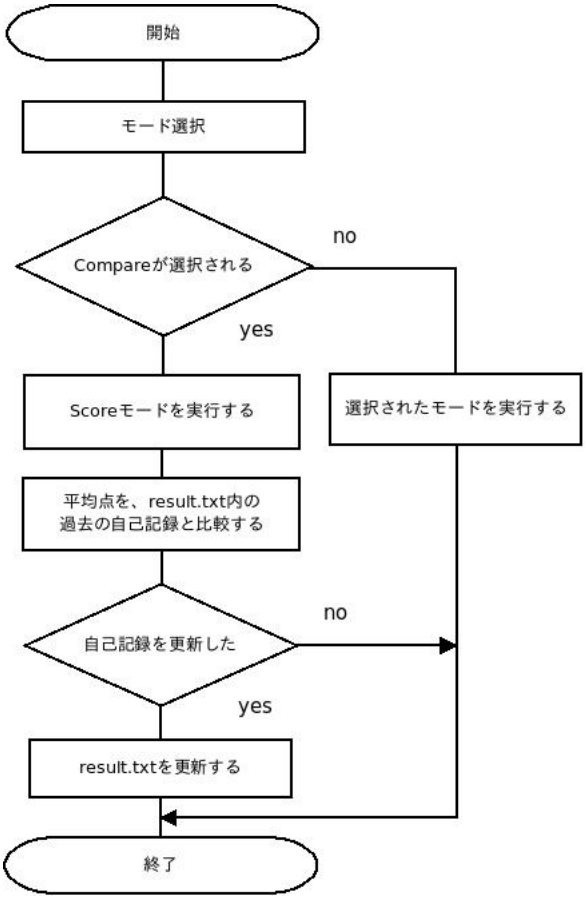

図 1　Compare.c を実行した際の動作を表すフローチャート

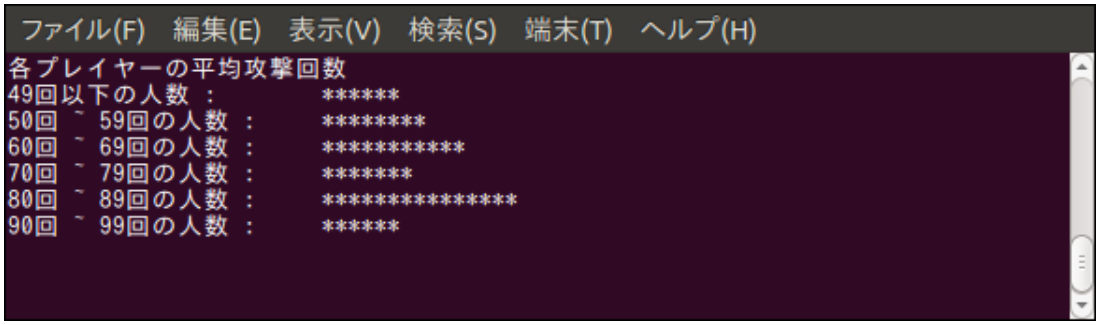

図 2　Compare モード実行時の度数分布の表示

5　システム

本章では、新しく作成した Battle Ship ゲームの自動コンパイルシステムの説明をする。

5.1　概要

Battle Ship ゲームの各モードを利用する際、プログラムによって、手動でコンパイルする作業を省き、モード選択をするだけで利用できるようにする。

5.2　動作

プログラムを起動すると、Battle Ship ゲームの各モードを選択する表示が出る。選んだモードに合わせて、自身の作ったアルゴリズムファイルを含んだ Battle Ship ゲームの実行ファイルが生成され、自動で実行される。

プログラムの動作を表したフローチャートを図 3 に示す。実際にプログラムを使用し、そこから Score モードを実行した様子を図 4 に示す。

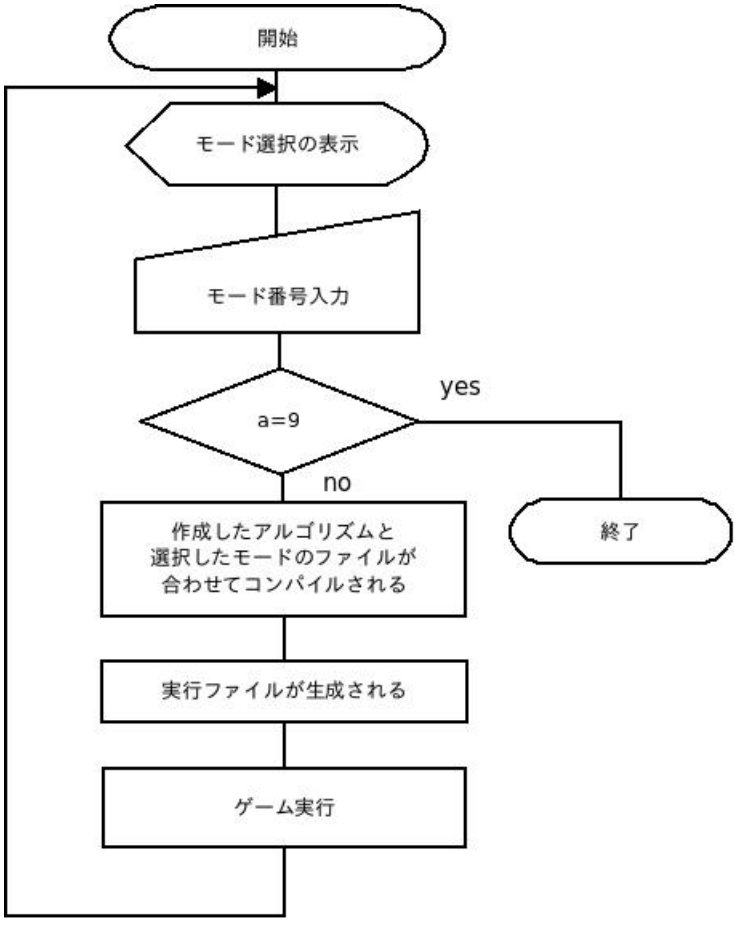

図 3　自動コンパイルシステムを実行した際の動作を表すフローチャート

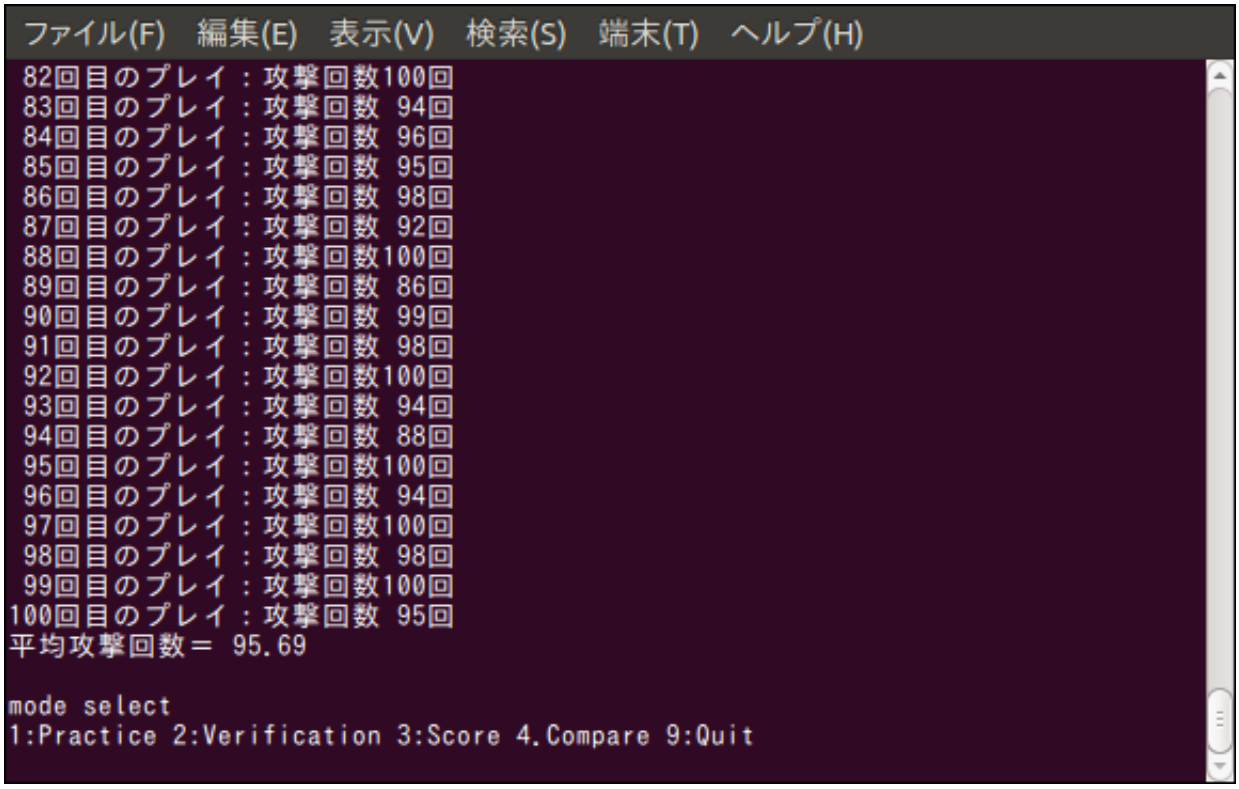

図 4　Score モードを実行し、再度モード選択に移った様子

6 結果と考察

- Battle Ship ゲームの利用者が、共通の基準で各自のアルゴリズムを評価できるようにするという目的について、収集した利用者の成績を度数分布で評価し、その結果を表示することで、利用者はより客観的な評価を得ることができるようになる。度数分布の表示について、各度数に属する人数を単に数値で表すのではなく、グラフ化して表すことで、分布の様子が視覚的に分かりやすくなった。しかし、度数をどの程度の間隔で区切るのが最も良いか、度数分布以外の適した評価方法が存在するのではないか、といった事については考察がなされていないため、その余地が有ると考える。
- 従来の Battle Ship ゲームから比較してコンパイルの手間が省かれることで、利用者はより少ない操作でゲームを利用する事が可能となり、アルゴリズムを編集してからその挙動をチェックするまでの手順を省略することができた。それにより、より多くの試行回数を重ねることができるようになった。しかし、ゲーム本体を構成する複数のファイルのダウンロードが必要であったり、ゲームを利用するためには多少の準備が必要となるため、その問題を解決すれば、新規利用者が更にこのゲームを利用しやすくなると考える。
- Battle Ship ゲームを実行し、その結果を自動で保存し蓄積するプログラムは作成したが、ネットワークを介して動作する機能は実装できず、ローカルな端末での動作に留まっている。そのため、実際に複数の利用者が互いの実行結果を共有し、成績の分布を表示することはできていない。ゲームの記録の蓄積をネットワーク上で行い、そのデータにアクセスするシステムを製作すれば、それが実現できる。また、ゲーム本体をダウンロードして利用する現在の形式は、仮にゲームの仕様に変更が有る場合などに、利用者が再度ゲーム本体をダウンロードする必要があるが、本体をネットワーク上の 1 箇所に設置し、それを利用する形式にすれば、再度ダウンロードさせるという負担を強いずに済む。

7　まとめ

今回作成した度数分布によりゲームの成績を評価するシステムによって、学習者が記述したアルゴリズムの良し悪しが、各自の点数そのものだけでなく、他の利用者の点数を見ることで客観的な評価を得られるようになった。点数データの収集さえ行われていれば、単独で Battle Ship ゲームを利用しても、利用者全体との比較ができる。しかし、収集したデータから分布を表示するプログラムは作成できたが、データを自動で収集し、収集したデータにアクセスするという処理については、ネットワーク上で動作するものは完成していない。

ゲーム利用時に発生するコンパイル等の手間を削減するシステムによって、アルゴリズムの編集と挙動のチェックの間の手順が無くなり、利用者は時間あたりのゲームの試行回数を増やすことができるようになった。

これらのシステムを実際にプログラミング初学者に利用してもらいその効果を検証する事は行っていないため、実際に教育に導入する際には、その検証を行う必要があると考える。

8　今後の課題

現在の Battle Ship ゲームは、情報を学ぶ学生をターゲットとして開発されている。また、コンパイラが用意された環境でのみ利用できる。今後、対象年齢を引き下げ、あらゆる環境で Battle Ship ゲームを利用した学習を実施したい場合、以下の課題が考えられる。

8.1　蓄積する得点記録の書式

今回作成したシステムは、ログイン名をプレイヤー名として使用し、テキストファイルに得点とログイン名を併記して得点記録を作っている。あらゆる環境でゲームを実施できるようにする場合、ログイン名が同一のプレイヤー同士が現れる可能性が有るので、この場合の対策を講じなければならない。

8.2　ゲームをネットワークに対応させる

全ての利用者の成績等のデータを 1 箇所に集約し、利用者同士が競い合えるようにしたい場合、そのデータを集約するサーバーと、サーバーに接続するシステムをゲーム本体に実装る必要がある。

8.3　誰でも簡単に利用できるシステムにする

現在の Battle Ship ゲームは terminal 上で実行する形になっているが、terminal に馴染みの無い利用者には、それが一つの壁となる。あらゆる人に利用してもらうためには、terminal を使用せず、更に簡単な形で利用できるシステムを開発する必要がある。そうしたシステムの例として、GUI アプリケーションを挙げる。そのアプリケーションのイメージ図を図 5 に示す。

図 5　Battle Ship 用アプリケーションのイメージ図。 アルゴリズムが記述されたファイルを指定し、play ボタンを押すことで実行結果が表示される。

謝辞

本研究を行うにあたり、ご指導頂いた大垣斉 准教授、ご助言を頂いた東川諒央助手、そして研究へのモチベーション維持に一役買ってくれた同輩・後輩達に、深く感謝の意を表します。

参考文献

[1] 東川諒央. ゲームを題材としたプログラミング教育支援システムの開発. 2009.

[2] 東川諒央. ゲームを題材としたプログラミング教育支援システムの開発と評価. 2011.

[3] レーダー作戦ゲーム - wikipedia. http://ja.wikipedia.org/wiki/海戦ゲーム#.E3.83.AC.E3.83.BC.E3.83.80.E3.83.BC.E4.BD.9C.E6.88.A6.E3.82.B2.E3.83.BC.E3.83.A0.

[4] Battleship (game). http://en.wikipedia.org/wiki/Battleship_(game).

付録 A　ソースコード

A.1　モード選択プログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main(void) {

  int a=0;
  char buff[64],number[16];
  char *login = getlogin();

  while(a != 9){

    printf("\nmode select\n1:Practice 2:Verification 3:Score 4.Compare 9:Quit\n");
    scanf("%d",&a);

    switch(a){

    case 1:{
      system("./practice");
      break;
    }

    case 2:{
      sprintf(buff,"gcc -o %s function.o verification.o %s.c",login,login);
      system(buff);

      sprintf(buff,"./%s",login);
      system(buff);

      break;
    }

    case 3:{
      sprintf(buff,"gcc -o %s function.o score.o %s.c",login,login);
      system(buff);

      sprintf(buff,"./%s",login);
```

```
      system(buff);
      break;
    }

    case 4:{
      sprintf(buff,"gcc -o %s function.o compare.o name_entry.o %s.c",login,login);
      system(buff);

      sprintf(buff,"./%s",login);
      system(buff);
      break;
    }

      default:
break;
    }
  }

  return;
}
```

A.2 Compare モード

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

char line[100];

void haichi ();
void dispsb ();
int zanteki ();
void shoot ();
int meichup ();
int meichud ();

int tsb[10][10], sb[10][10];
int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
int gx=0, gy=0;
int i, kekka, k;
int bx, by;
int fun_c=0, at_flg=0;
int up, rig, dow, lef;
float average;

int main (void)
{
  int j, a, b, total = 0;

  for (j = 1; j <= 100; j++)
    {
      skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
      fun_c = 0, at_flg = 0;

      haichi ();
      for (a = 0; a < 10; a++)
{
  for (b = 0; b < 10; b++)
    {
      sb[a][b] = N;
    }
}
```

```
      zanteki ();
      i = 1;
      for (i = 2; i <= 101; i++)
{
  k = meichup ();
  kekka = meichud ();
  if (zanteki () || i == 101)
    {
      break;
    }
  shoot();
}
      i--;
      total = total + i;
      printf ("%3d 回目のプレイ：攻撃回数 %3d 回\n", j, i);

    }

  average = total / 100.0;
  printf ("平均攻撃回数＝ %6.2f\n", average);

  name_entry(average);

  return 0;
}

void
haichi ()
{
  int tate, yoko, cx, cy;
  static unsigned int seed = 1;

  srand ((unsigned int) time (NULL) * seed);

  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
{
  tsb[tate][yoko] = N;
}
    }
```

```
  if (rand () % 2 == 0)
    { /* 戦艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 6;
      tsb[cx][cy] = SK;
      tsb[cx + 1][cy] = SK;
      tsb[cx + 2][cy] = SK;
      tsb[cx + 3][cy] = SK;
      tsb[cx + 4][cy] = SK;
    }
  else
    { /* 戦艦横方向 */
      cx = rand () % 10;
      cy = rand () % 6;
      tsb[cx][cy] = SK;
      tsb[cx][cy + 1] = SK;
      tsb[cx][cy + 2] = SK;
      tsb[cx][cy + 3] = SK;
      tsb[cx][cy + 4] = SK;
    }

  if (rand () % 2 == 0)
    { /* 巡洋艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 7;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N
     || tsb[cx + 3][cy] != N)
{
  cy = rand () % 10;
  cx = rand () % 7;
}
      tsb[cx][cy] = JK;
      tsb[cx + 1][cy] = JK;
      tsb[cx + 2][cy] = JK;
      tsb[cx + 3][cy] = JK;
    }
  else
    { /* 巡洋艦横方向 */
      cx = rand () % 10;
      cy = rand () % 7;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N
     || tsb[cx][cy + 3] != N)
{
```

```
  cx = rand () % 10;
  cy = rand () % 7;
}
      tsb[cx][cy] = JK;
      tsb[cx][cy + 1] = JK;
      tsb[cx][cy + 2] = JK;
      tsb[cx][cy + 3] = JK;
    }

  if (rand () % 2 == 0)
    { /* 駆逐艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
{
  cy = rand () % 10;
  cx = rand () % 8;
}
      tsb[cx][cy] = KK;
      tsb[cx + 1][cy] = KK;
      tsb[cx + 2][cy] = KK;
    }
  else
    { /* 駆逐艦横方向 */
      cx = rand () % 10;
      cy = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
{
  cx = rand () % 10;
  cy = rand () % 8;
}
      tsb[cx][cy] = KK;
      tsb[cx][cy + 1] = KK;
      tsb[cx][cy + 2] = KK;
    }

  if (rand () % 2 == 0)
    { /* 潜水艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
{
  cy = rand () % 10;
```

```
  cx = rand () % 8;
}
      tsb[cx][cy] = SM;
      tsb[cx + 1][cy] = SM;
      tsb[cx + 2][cy] = SM;
    }
  else
    { /* 潜水艦横方向 */
      cx = rand () % 10;
      cy = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
{
  cx = rand () % 10;
  cy = rand () % 8;
}
      tsb[cx][cy] = SM;
      tsb[cx][cy + 1] = SM;
      tsb[cx][cy + 2] = SM;
    }

  if (rand () % 2 == 0)
    { /* 輸送船縦方向 */
      cy = rand () % 10;
      cx = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N)
{
  cy = rand () % 10;
  cx = rand () % 9;
}
      tsb[cx][cy] = YS;
      tsb[cx + 1][cy] = YS;
    }
  else
    { /* 輸送船横方向 */
      cx = rand () % 10;
      cy = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N)
{
  cx = rand () % 10;
  cy = rand () % 9;
}
      tsb[cx][cy] = YS;
      tsb[cx][cy + 1] = YS;
```

```
    }
  seed = rand ();
}

void
dispsb ()
{
  int i, j;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
  printf ("  ---------------------\n");

  for (i = 0; i < 10; i++)
    {
      printf ("%2d", i + 1);
      for (j = 0; j < 10; j++)
{
  putchar ('|');
  switch (sb[i][j])
    {
    case BM:
      putchar ('O');
      break;
    case BH:
      putchar ('X');
      break;
    default:
      putchar (' ');
      break;
    }
}
      printf ("|\n");
    }

  printf ("  ---------------------\n");
}

int
zanteki ()
{
  if (skp <= 0 && jkp <= 0 && kkp <= 0 && smp <= 0 && ysp <= 0)
    {
      /* printf("全艦撃沈！\n"); */
```

```
      return 1;
    }
/*
printf("残敵＝");
if(skp > 0) {
printf("戦艦 ");
}
if(jkp > 0) {
printf("巡洋艦 ");
}
if(kkp > 0) {
printf("駆逐艦 ");
}
if(smp > 0) {
printf("潜水艦 ");
}
if(ysp > 0) {
printf("輸送船");
}
putchar('\n');
*/
  return 0;
}

int
meichup () /* x->gx, y->gy change */
{
  int a, b;

  a = tsb[gx][gy];
  b = sb[gx][gy];

  if (b != N)
    {
      return BH;
    }
  if (a == SK)
    {
      sb[gx][gy] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
```

```
    {
      sb[gx][gy] = BM;
      jkp--;
      return JK;
    }
  if (a == KK)
    {
      sb[gx][gy] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[gx][gy] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[gx][gy] = BM;
      ysp--;
      return YS;
    }
  sb[gx][gy] = BH;
  return BH;
}

int
meichud ()
{
  if (k == BH)
    {
      return BH;
    }
  else if (k == SK && skp <= 0)
    {
      return SK;
    }
  else if (k == JK && jkp <= 0)
    {
      return JK;
    }
  else if (k == KK && kkp <= 0)
```

```
    {
      return KK;
    }
  else if (k == SM && smp <= 0)
    {
      return SM;
    }
  else if (k == YS && ysp <= 0)
    {
      return YS;
    }
  return BM;

}
```

A.3 度数分布表示プログラム

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

FILE *file;
char *score,str[16] ;
int i,j,temp;
int dist[200] = {0};

int main (void){
  file = fopen("./result.txt", "r");

  for(i=1; i<150; i++){

    while(fgets(str, 16, file)){

      score = strtok(str, ",");

      temp = atoi(score);
      temp = temp / 10;
      dist[temp]++;

      break;

    }
  }

  printf("各プレイヤーの平均攻撃回数\n");

  for(i=0;i<10;i++){

    if(i<4){
      dist[4] += dist[i];
    }

    else if(i>4){
      printf("%d 回 ~ %d 回の人数 :\t",i*10,i*10+9);
      for(j=0;j<dist[i];j++){
      putchar('*');
      }
```

```
      printf("\n");
    }

    else{
      printf("49 回以下の人数 :\t");

      for(j=0;j<dist[i];j++){
putchar('*');
      }
      printf("\n");
    }
  }
  fclose(file);

  return 0;
}
```

A.4 点数記録書き込みプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

FILE *temp,*file;
int i;
float x;
char str[16];
char *pt,*student;

int name_entry (float average){

  file = fopen("./result.txt","r");
  temp = fopen("./temp.txt", "a");

  for(i=0; i<255; i++){
    while(fgets(str, 16, file)){

      if(strstr(str,student)){
fputs(str,temp);

      }
      else{
pt = strtok(str,",");
x = atof(pt);
if(average < x){
  printf("NEW RECORD!\tNEW:%2.2lf\tOLD:%2.2lf\n",average,x);
  fprintf(temp,"%2.2lf,%s\n",average,student);

}

      }
    }

    fclose(file);
    fclose(temp);

    system("cp temp.txt result.txt && rm temp.txt");
```

```
  }
  return 0;
}
```