16. ゲームと乱数

ここにあげる例は、画面中 10 行に乱数で文字を表示し、それぞれの文字の出現数を数えておきます。そして、画面を見て多そうだと思う文字をキー入力し、その文字の出現数がスコアとなります。これを 5 回繰り返して合計点を競うゲームです。

```
 1  /*
 2       alphabet game
 3                       written by Sugiyama (sugiyama@vgd.co.jp)
 4  */
 5  #include <stdio.h>
 6  #include <stdlib.h>
 7  #include <time.h>
 8  int main()
 9  {
10      int ch, i, inkey, score = 0, count = 0;
11      int a[26];                                      /* 文字カウンタ */
12
13      srand((unsigned)time(NULL));                    /* 乱数生成の初期化      */
14      while(count != 5){
15          for (i = 0; i < 26; i++) {                  /* 文字カウンタの初期化 */
16              a[i]=0;
17          }
18          for(i = 0; i < 790; i++) {
19              if (i%79 == 0) putchar('\n');
20              ch = rand() % 50;               /*   0 から 49 までの乱数発生     */
21              if (ch >= 26) {                 /*   26 以上は空白にする        */
22                  putchar(' ');
23              } else {
24                  putchar(ch + 'a');      /* 乱数で出した文字の表示            */
25                  a[ch]++;                /* 乱数で出した文字のカウントアップ */
26              }
27          }
28          printf("\n");
29          inkey = getchar();                              /*  文字入力    */
30          if ('a' <= inkey && inkey <= 'z') {
31              score = score + a[inkey-'a'];
32              printf("%c = %d \n", inkey, a[inkey-'a']);
33          }
34          count++;
35          while((inkey = getchar()) != '\n');  /*   改行までを読み飛ばす     */
36      }
37      printf("total score = %d\n", score);
38  }
```

上記の例で使われている乱数について説明します。まず、乱数とはある確率で、相互にまったく独立になる

ように作られた一群の数です[1]。コンピュータで使われる乱数は、通常はある初期値から始まり、順次ある関数[2]を繰り返して作られていきます。このような方法で得られる数列は、でたらめには見えるだけで真の意味では乱数ではないので**疑似乱数**といいます。
C 言語での乱数の使い方は、まず乱数の初期化を行ないます。これは、実行するたびに同じ乱数列がでないようにするためのものです。

```
srand((unsigned)time(NULL));                /* 乱数生成の初期化  */
```

乱数の発生は以下のようにします。この例では、0 から 49 の中のどれかの数字が乱数として得られます。この 50 を変えることによって、範囲を変化させることができます。

```
ch = rand()%50;                 /*   0 から 49 までの乱数発生     */
```

簡単なゲームの例

以下に自由課題として適当と思われるゲームのテーマを書きます。課題はこの中から選んでもいいですし、もちろんこれ以外でもいいです。

マスター・マインド

乱数で 0 から 9 の数を 4 桁からなる数を作り、その 4 つの数は同じものがないようにする。そして、プレイヤーはその 4 桁の数を当てるのですが、はずれた場合は 4 つの数の内いくつの数があっているかの正解数しかヒントとして出ません。そのヒントを頼りに正解を何回で見つけるかを競うゲームです。

ブラック・ジャック

トランプのブラック・ジャックです。絵札は 10、A は 1 か 11 として数え、より内和が 21 に近くなるようにカードを引いていくゲームです。親がカードを引くかやめるかの判断をどうするかがポイントです。

ハイ・アンド・ロー

トランプでまず 1 枚をめくり、次にめくるカードがその数より大きいか、小さいかを当てるゲームです。当てている間は引き続け、何回連続して正解するかを競います。

[1]乱数については、Knuth, D. E. “The Art of Computer Programming Vol.2: Seminumerical Algorithms” (アスキー出版から翻訳あり) という名著があります。
[2]合同線形法では、法 M、乗数 λ、増分 μ (いずれも整数) を適切に選び、適当な整数 x_0 から出発して漸化式 $x_i = \lambda x_{i-1} + \mu \ (mod M)$ により数列 $\{x_i\}$ を生成する。

次に、トランプのカードを作るプログラム例を示します。ただ、これはカードの数だけに注目し、4 つのスーツは区別していません。

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <time.h>
 4  main()
 5  {
 6      int tmp, i, j, k;
 7      int card[52] =  {
 8          1, 1, 1, 1,
 9          2, 2, 2, 2,
10          3, 3, 3, 3,
11          4, 4, 4, 4,
12          5, 5, 5, 5,
13          6, 6, 6, 6,
14          7, 7, 7, 7,
15          8, 8, 8, 8,
16          9, 9, 9, 9,
17          10,10,10,10,
18          11,11,11,11,
19          12,12,12,12,
20          13,13,13,13
21          };
22      srand((unsigned)time(NULL));            /*   乱数生成の初期化          */
23      for (k=0;k<1000;k++) {                  /*   カードのシャッフル        */
24          i=rand()%52;
25          j=rand()%52;
26          tmp=card[i]; card[i]=card[j]; card[j]=tmp;
27      }
28      for (k=0;k<52;k++) {                    /*   カードのとりだし          */
29          printf("%d \n",card[k]);
30      }
31  }
```

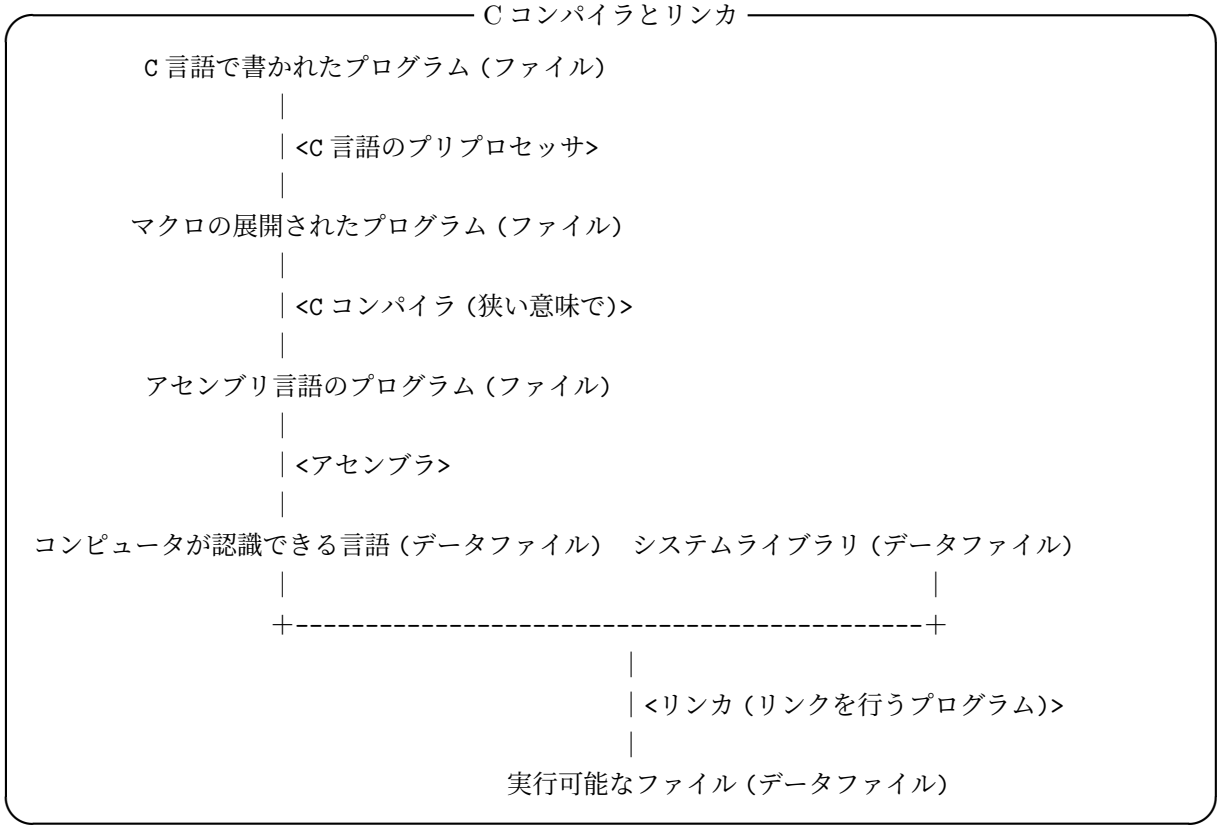
(注)

```
    int card[52] =  {
        1, 1, 1, 1,
        2, 2, 2, 2,
        ....
        ....
        13,13,13,13
        };
```

というところは、配列 card を宣言する時にその初期値を代入しているものです。

17. C 言語プログラムの処理

プログラムを C 言語で書いても、コンピュータはプログラムをそのまま実行するわけではありません。C 言語で書かれたプログラムはコンピュータが理解できる言語 (機械語といいます) に変換する必要があります。この変換プログラムのことをコンパイラ (compiler) といいます。下の図はどのような手順でその変換が行われているかを示しています。



上に出てきた単語を説明します。

- ファイル
  ファイルとはある意味をもって集められた情報の集まりです。例えば、C 言語でプログラムを作りそれを '1116ex1.c' という名前でフロッピディスクに書き込んだとすると、それが 1 つのファイルとなり、'1116ex1.c' がファイル名になります。
- プリプロセッサ
  これは# で始まる今までおまじないと呼んでいた部分を処理するものです。#で始まる行以外の部分は全く処理されません。#で始まる命令をマクロ (正式にはプリ・プロセッサ制御文) と呼びます。詳しくは次の節で説明します。
- アセンブリ言語
  アセンブリ言語は機械語にほぼ 1 対 1 に対応している言語です。機械語は数の並びですが、アセンブリ言語はそれに比べれば人間が理解しやすい表記になっています。
- アセンブラ
  アセンブラはアセンブリ言語を機械語に変換するプログラムです。よって、アセンブリ言語用のコンパイラとみなせるわけですが、これをコンパイラと呼ばずアセンブラと呼んでいます。
- コンピュータが認識できる言語 (機械語)
  コンピュータは電気信号で動いています。電気の信号はオンかオフしかありません。数学では 2 進数

です。この信号線が 16 本あると 2^{16} 種類の信号を表現できます。さらに時間の同期をとることで 2^{16} 種類のデータを複数伝えることが可能です。2 進数の足し算と、引き算はアルゴリズム的に単純であり、1 本の信号線の AND,OR,XOR で表現可能です。これがもっと複数につながってコンピュータの認識できる言語が成り立っています。

- リンカ
  システム中にある関数 (例えば printf,scanf など) のプログラムは既に機械語になっています。これと個人で作ったプログラム (コンパイルして機械語になったもの) を 1 つにまとめて 1 つのファイルにして実行可能なファイルにします。この操作を リンク と呼びます。システムに最初から存在する関数はライブラリといいます。つまり、実習でのリンクは個人の作ったプログラムとシステムのライブラリを 1 つの実行できるファイルにしています。システムにあるライブラリは 200 個程度の関数が入っています。今まで printf,scanf,sqrt など自分自身で宣言、定義していない関数も使えたのはこの理由からです。

個人で作成した関数、変数、またはシステムが備えている関数をリンクしたあと実行ファイルができるといいました。個人で作成したたくさんの関数とシステムの関数の中から最初に実行される関数はなんでしょうか? これが main という関数です。これはライブラリにはなく、**個人で定義 (作る) しなければならない**ものです。これがないとどの関数から実行して良いのかわからなくなります。よって C 言語のプログラムには **1 つ main という関数が必ず必要です** 。2 つ以上あるとまた複数始めるところがあると迷ってしまい、エラーとなります。この main という関数がないと試すことも、実行することもできません。C 言語の本などでは main 関数が書いていないことがありますが、main という関数は省略されているだけで、実行するには必ず必要です。

歴史をたどれば、コンピュータが登場した当初は人間が機械語そのものをコンピュータに入力していました。しかし 0 と 1 の数字の列を人間が扱うのはいかにも能率が悪く、もっと人間が扱い易い言語としてアセンブリ言語が開発されました。ただ、前にも述べたようにアセンブリ言語は機械語とほぼ 1 対 1 に対応していますから、アセンブリ言語は使おうとしているコンピュータに依存します。新しいコンピュータが開発されるたびにプログラムを作り直すのは大変です。そこでマシンに依らない言語として、FORTRAN や COBOL が登場しました。C 言語もそのような言語の 1 つで、このようなマシンに依らない言語のことを高水準言語と呼んでいます。これに対してアセンブリ言語のことを低水準言語と呼ぶことがあります。

18. C 言語のマクロ

マクロは#で始まる命令です。この部分だけ、プリプロセッサで処理されます。ここで処理された後はこの命令の跡は残りません。このマクロを処理したことを「マクロを展開した」といいます。たくさんのマクロがあるのですが、今回は以下のマクロについて説明します。

- #include
- #define

`#include`

今までおまじないで済まされてきた `#include` 命令の種明かしをします。これまで `#include <stdio.h>` や `#include <math.h>`を使ってきましたが、これは文字通り '`stdio.h`' や '`math.h`' というファイルをそこに挿入することを意味します。'`stdio.h`' には次のような箇所があり、`putchar()` のマクロ定義 (putchar() は関数ではない) が記してあります。

ファイル stdio.h (部分)

```
struct __FILE {
    /* NOTE: Must match (or be a prefix of) __streambuf! */
    int _flags; /* High-order word is _IO_MAGIC; rest is flags. */
    char* _gptr; /* Current get pointer */
    char* _egptr; /* End of get area. */
    char* _eback; /* Start of putback+get area. */
    char* _pbase; /* Start of put area. */
    char* _pptr; /* Current put pointer. */
    char* _epptr; /* End of put area. */
    char* _base; /* Start of reserve area. */
    char* _ebuf; /* End of reserve area. */
    struct streambuf *_chain;
};
typedef struct __FILE FILE;
(途中省略)
extern struct _fake_filebuf __std_filebuf_0, __std_filebuf_1,
__std_filebuf_2;
#define stdin ((FILE*)&__std_filebuf_0)
#define stdout ((FILE*)&__std_filebuf_1)
#define stderr ((FILE*)&__std_filebuf_2)

#define getchar() getc(stdin)
#define putchar(x) putc((x),stdout)
```

また、ライブラリ関数 sqrt を使う時には '#include <math.h>' として sqrt のプロトタイプ宣言を挿入します。

ファイル名を指定している部分が <> で囲んであると C 言語のシステムが与えているファイルを意味し、" " で囲んであると個人のユーザが作成したファイルを指定することになります。

#define

このマクロには

- #define　識別子
- #define　識別子　展開された形
- #define　識別子 (引数列)　展開された形

といった表記方法があり、

#define　識別子　展開された形

の場合、その #define が現れた以降のプログラムで 「識別子」が現れた部分を「展開された形」に置き換えます。よって次の例では COUNT は展開されません (COUNT をマクロで定義するまえに COUNT を使っているので)。

悪い例

```
 1  #include <stdio.h>
 2  main()
 3  {
 4          int i,s;
 5          s = 0;
 6          for (i = 1 ; i <= COUNT ; i++) {
 7                  s = s + i;
 8          }
 9          printf("1+2+....+%d = %d\",COUNT,s);
10  }
11  #define COUNT 10
```

これまでの授業では

```
#define FACT 40
```

などとして「展開された形」の部分は数値でしたが、この部分は何でも構いません。変数でも 関数名でも C コンパイラ予約語でもなんでも可能です。特殊な例で以下のようなものも可能です。

#define を使った例

```
 1  #include <stdio.h>
 2  #define FOREVER while(1)  /* 無限ループを FOREVER とする。 */
 3  main()
 4  {
 5      int s,i;
 6      s = 0;
 7      FOREVER {
 8          printf("Input Data=");
 9          scanf("%d",&i);
10          if (i == 0) break;
11          s = i + s;
12      }
13      printf("sum = %d\n",s);
14  }
```