Chapter 5

テキストベースのゲームアプリ

5.1　アプリを作ろう

この演習授業を含め、いままで「C 言語の基礎」を勉強してきたわけだが、勉強した内容が実際に使えないと

意味がない。

それに、やっぱり動くものができると楽しい[1]。是非、楽しみながらプログラミングの腕を上げて欲しい。とはいえ、しょっぱなから真新しいことがたくさん必要になるアプリもたいへんである。いままで学んだものを使ってできるアプリから始めよう。いままで学んだものは基礎であり、実際、今後も何度となく使うであろう。「回数をこなす（＝経験をふやす)」というのもこの演習のねらいである。

5.2　実装についての注意

これから、いくつかの関数を実装していくが、次回以降で作成する関数とともにアプリケーションに組み込む。そのため、ソースファイルをいくつか用意することになる。下の例をみてみよう。まず、ソースファイルa.c、b.cとヘッダファイルb.hだ。

[1]これは、ベテランプログラマーでも同じだったりする。

```
#include "b.h"

int main(int argc, char **argv){
     add(1,2);
}
```

```
#ifndef B_H_
#define B_H_

#include <stdio.h>

void add(int i, int j);

#endif /* B_H_ */
```

```
#include "b.h"
void add(int i, int j){
   printf("%d + %d = %d", i, j , (i+j));
}
```

a.cにmain関数が、b.cにaddという名前の関数が定義されている。ヘッダファイルb.hは、というと、b.cで定義されている関数addのプロトタイプ宣言をしている。ヘッダファイルには、構造体の定義などをすることも多い。これは、ヘッダファイルのそもそもの役割が、他のソースプログラムと関数や構造体の定義を共有することにあるからである[2]。ちなみに、b.hの

```
#ifndef B_H_
#define B_H_
  ...
#endif
```

は、B_H_が定義されていればスルーし[3]、されていなければ定義を加える(#define)という意味である。これは、同一のヘッダファイルがインク

[2]stdio.hのインクルードが、標準的な入出力を行う関数を利用する際に必要であることを考えれば、何気なくだが実はいつも扱っているものではあるわけだ

[3]#ifndef〜#endifは「定義されていなければ」という意味

ルードされても大丈夫なようにする工夫である[4]。このような単純なプログラムでも、関数を定義するファイル、関数を利用するファイルでは、ともにヘッダファイルをインクルードするので、この工夫を活用されたい[5]。

このように作られたプログラムをコンパイルするときには次のようにすればよい。

```
gcc a.c b.c -o test
```

すなわち、拡張子が.cであるファイルを並べ、そのあと-oというオプションをつけ、コンパイル後の実行ファイルに名前をつければよい。これは、内部的にはそれぞれのプログラムのオブジェクトファイルを作って、さらにそれらをリンクするのと同等の処理を行う。いずれにしても結果としてtestというなまえの実行ファイルができあがる[6]。

```
gcc -c a.c
gcc -c b.c
gcc a.o b.o -o test
```

5.3 ゲームの仕様(1)

3回にわたってゲームを作っていくが、今回はクイズゲームを作ってみよう。このクイズは以下の機能をもっているものとする。

- クイズ問題の読み込み
- 回答入力・正誤判定・点数表示

クイズ機能は以下の2つの関数を使って実現する。

- `int  quizReader(QUIZ *quiz, int *n);`

  関数の第一引数はQUIZ構造体の配列、第二引数は読み取った問題数を返すためのポインタである。 QUIZ構造体は以下のように定義されているものとする。

[4]インクルードガードという

[5]これを使わないと、二カ所以上でヘッダファイルをインクルードしなければいけないときの二重宣言の問題を解決するのがかなり大変

[6]これから実行ファイルをたくさんつくるので、さすがにa.outからは卒業しよう。

```
struct _quiz{
  char question[50];
  char ans1[50];
  char ans2[50];
  int correct_ans;
};
typedef struct _quiz QUIZ;
```

まず、テキストファイルquiz.txtをオープンする。オープンできない場合、invalid quiz fileというメッセージを画面に表示し、-1を返す。 quiz.txtはCSVであり、

```
What_is_the_first_month_in_a_year?,January,December,1
Where_is_the_capital_of_Japan?,Kyoto,Tokyo,2
What_is_the_name_of_this_building?,Building_2, Building_4,1
```

のように、一行に問題、解答1、解答2、正解番号が並んでいるとする。これをfscanfを使って読み込むときには

```
fscanf(fp,"%[^,],%[^,],%[^,],%d", question, ans1, ans2, &correct_ans)
```

などとして読むとよい。また、スペースがあると読み込みを終えてしまうので、スペースの代わりにアンダーバー（_）を使っていることに注意すること。 CSVの一行を読み取ったら配列quizの要素である構造体にそれぞれ該当するメンバー変数に格納すること。また、全行読み込みがすんだらポインタnを使って読み込み行数を返す。ただし、読み込み可能行数の最大はN(=100)であるとする。最後にファイルを閉じる。以上が正常に終了したら0を返す。

- `void input();`

  N個のQUIZ型の要素を持つ配列を用意する(ポインタを使うこと。初期化にはmallocを使え。)。 quizReader関数を使って問題を読み込み、もしその戻り値が0であれば以下の例のように問題を提示せよ。

```
What_is_the_first_month_in_a_year?
1:January 2:December
```

一行目が問題で二行目が選択肢であり、QUIZ構造体のans1メンバー、ans2メンバーがそれぞれ1:、2:の後ろに提示される。 scanf関数で解答番号をユーザーに入力させ、correct_ansと等しいときは画面にCORRECT!、等しくないときはWRONG!と画面に表示する。最後の問題が終了したら、Congrat!と表示し、score:として点数を表示する。点数は1問正解ごとに5点追加されるものとする。

以下は出力例である。

```
What_is_the_first_month_in_a_year?
1:January 2:December
1
CORRECT!

Where_is_the_capital_of_Japan?
1:Kyoto 2:Tokyo
1
WRONG!

What_is_the_name_of_this_building?
1:Building_2 2: Building_4
3
WRONG!

Congrat!
score:5
```

これらの関数を、quiz.cというファイルに記述せよ。対応するヘッダファイル(quiz.h)は以下のようになるはず。

```
#ifndef QUIZ_H_
#define QUIZ_H_

#define N 100

struct _quiz{
  char question[50];
  char ans1[50];
  char ans2[50];
```

```
  int correct_ans;
};

typedef struct _quiz QUIZ;

int  quizReader(QUIZ *quiz, int *n);
void input();

#endif /* QUIZ_H_ */
```

なお、以下のプログラム（ファイル名をquiz_driver.c、実行ファイル名をquizとせよ）に上記のプログラムをリンクして実行せよ。

```
#include "quiz.h"

int main(){
  input();
  return 0;
}
```