プログラミング演習 IIIA

hangman

21-403 伊藤幸太郎

2024 年 5 月 9 日

第 1 章

プログラムの仕様

1.1 プログラムの機能及び使い方

このプログラムは、未知の単語のスペリングを当てるクイズゲームである。最初に、単語がランダムに選択され、文字数のみプレイヤーに知らされる。プレイヤーはお題の単語に含まれていると思う文字を一回につき一文字入力する。入力された文字が単語に含まれていたら、その文字が使われている位置が表示される。逆に、含まれていなかったら、文字入力の残り回数が減少する。残り回数の制限内にすべて当てられたらゲームクリアである。一ゲームにつき 6 回までの間違いが許されており、7 回間違えるとゲームオーバーになる。

1.2 プログラム内で用いたデータ構造

まず、単語のデータは辞書ファイル `words` から取り出してくるという構成にした。乱数を生成し、その数の分 `fgets()` を実行し、`question` という配列に代入することによってランダムに選択することを実現している。この際、取り出した文字列に含まれる改行記号`\n` を取り除き、終端記号`\0` に置き換える処理を行って表示のバグを防いでいる。また、推測用の配列 `guess` は、`question` の文字数に応じて動的に確保し、ハイフンを代入し、お題の文字を伏せているという状態を表現している。正解した場合は、該当部分の `question` の文字を `guess` に代入して、伏せている部分が明らかになるという表現をしている。

1.3 各関数の仕様

1.3.1 getChar()

■書式 `char getChar(void);`

■戻り値 入力された文字

■処理内容 標準入力から 1 文字入力し、入力があると直ちに戻る。入力文字はエコーしない。
(この関数は資料内に予め用意されたものである。)

1.3.2 tolower_str()

■書式 `char* tolower_str(char* s);`

■引数　小文字に変換したい任意の文字列

■戻り値　小文字に変換された文字列

■処理内容　任意の文字列をすべて英小文字に変換する。

1.3.3　disp_info()

■書式　`void disp_info(char *guess, char* used_char, int wrong_count);`

■引数　推測用の配列、すでに入力されている文字を格納した配列、間違えた回数

■処理内容　推測中のスペル、推測過程ですでに入力された文字、間違えられる残り回数、入力を促すプロンプトを表示

第 2 章

プログラムの正しさの検証

2.1 引数エラー

デフォルトでは引数なしで実行すると、同じディレクトリ内にある words を参照する。第二引数にファイル名を指定すると、そのファイルを参照する。3 つ以上の引数が指定されると以下のようにエラーが表示される。

2.2 入力の制限

ゲーム中は、英小文字の入力のみ受け付けるようにしている。また、続けてプレイするかどうかを選択する場面では、y か n のみ入力可とし、それ以外の入力があると `invaild option.` と表示され、再度入力をさせる。

2.3 検証

資料内に例示されているように `programming` という文字列をお題とし、`e r s p a m b r s m` と入力した場合、どのように変化していくかを机上の検証とプログラムの検証を比較する。

2.3.1 机上での検証

以下のように変化していくと考えられる。ここでの `used_char` は既入力の文字を格納する `char` 型配列、`guess` は推測に用いる最初は伏せられている状態の `char` 型配列、wrong は間違えた回数をカウントする `int` 型配列を示す。

e　合っていないので used_char のみに代入。wrong をインクリメント。
r　合っているので guess の該当の位置に代入。used_char に代入
s　合っていないので used_char のみに代入
p　合っているので guess の該当の位置に代入。used_char に代入
a　同上
m　同上
b　合っていないので used_char のみに代入。wrong をインクリメント。
r　一度入力しているので、既入力と表示される。
s　同上
m　同上

2.3.2 プログラムでの検証

以下にプログラム上で検証した結果を示す。表示は以下のように対応している。

単語	guess
使われた文字	used_char
残り回数	7 - wrong

両者を比較すると、正しく動作していることが確認できる。

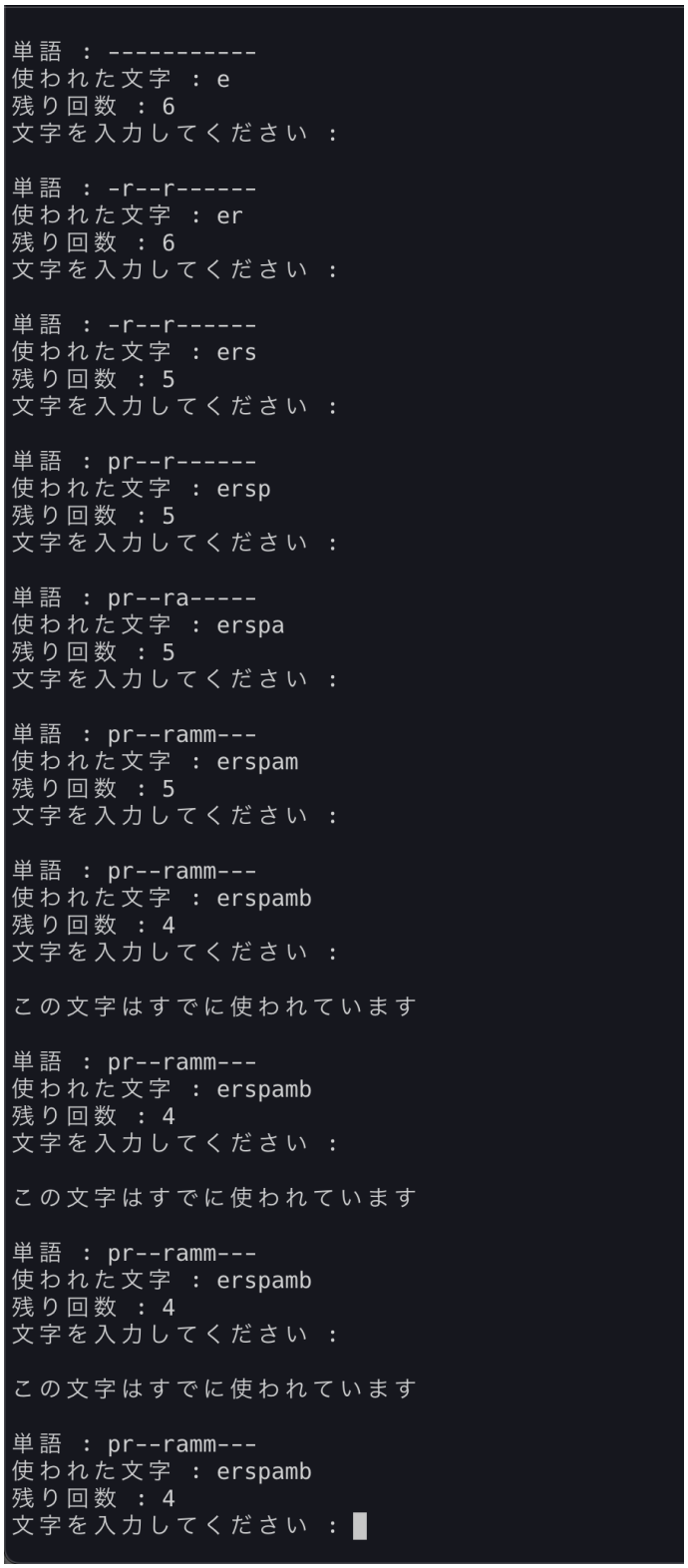

図 2.1　プログラムでの検証結果

第 3 章

問題点等

まず、ほぼすべての機能を main() 文に実装したため、非常にわかりずらいプログラムとなってしまった。機能を拡張したりする際にこの状態では非常に不便である。これは設計の段階で機能を分けて考え、それぞれ関数として実装することによって改善できる。次からはしっかり全体の構成を考えてから書き始めたいと思った。

また、ゲーム機能を実装するとき、変数の数が多くなってしまい、可読性が低いだけでなく、機能拡張等に対応しにくいプログラムとなってしまった。変数を使わない条件式を設定することによってわかりやすく、変化に柔軟なコードに改善可能である。

付録 A

プログラムリスト

Listing A.1　j21403.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <string.h>
5 #include <termio.h>
6 #include <ctype.h>
7
8 /*単語の最大長*/
9 #define MAX 50
10
11 char getChar(void);
12 char* tolower_str(char* s);
13 void disp_info(char *guess, char* used_char, int wrong_count);
14
15 int main(int argc, char *argv[]) {
16   /*単語データ読み込み*/
17   /*引数なしであればwords、引数で指定があればそのファイルを読み込む*/
18   FILE *fp = NULL;
19   if(argc == 1) {
20     fp = fopen("words", "r");
21   }
22   else if (argc == 2) {
23     fp = fopen(argv[1], "r");
24   }
25   else {
26     fprintf(stderr, "Usage : %s <Words file name>\n", argv[0]);
27     return 1;
28   }
29
30   if(fp == NULL) {
31     fprintf(stderr, "%s is not found.\n", argv[1]);
32     return 1;
33   }
34   /*読み込むファイルの単語数（行数）の把握*/
35   int file_length = 0;
36   char question[MAX];
37   while(fgets(question, MAX, fp) != NULL) {
38     file_length++;
39   }
40
```

```
41  /*変数宣言*/
42  char* guess;
43  char used_char[26];
44  int used_char_index = 0;
45  char input;
46  int wrong = 0, true = 0, found = 0, flag = 1;
47  int q_no, q_len;
48
49  /*ゲーム部分*/
50  while(1) {
51   /*問題をランダムに選択*/
52    srand((unsigned)time(NULL));
53    int q_no = rand() % file_length;
54    rewind(fp);
55    for(int i = 0; i < q_no; i++) {
56      fgets(question, MAX, fp);
57    }
58    /*改行文字を終端文字に置き換え*/
59    char* temp = strchr(question, '\n');
60    if (temp != NULL) {
61      *temp = '\0';
62    }
63    tolower_str(question);
64    q_len = strlen(question);
65
66    /*推測用の配列を動的に確保*/
67    guess = (char *)malloc(strlen(question));
68    for(int i = 0; i < q_len; i++) {
69      guess[i] = '-';
70    }
71    guess[q_len] = '\0';
72
73    /*推測と判定*/
74    while(flag) {
75      true = 0, found = 0;
76      /*状態を表示*/
77      disp_info(guess, used_char, wrong);
78
79      /*当てる文字を入力させる*/
80      /*英小文字だけ受け付ける*/
81      do {
82        input = getChar();
83      }
84      while(input < 'a' || input > 'z');
85
86      printf("\n\n");
87      /*すでに入力されているか判定*/
88      for(int i = 0; i < 26; i++) {
89        if(input == used_char[i]) {
90          printf("この文字はすでに使われています\n\n");
91          flag = 0;
92        }
93      }
94      if(!flag) {
95        flag = 1;
96        continue;
97      }
```

```
98
99        /*判定*/
100       for(int j = 0; j < q_len; j++) {
101         /*合っていたら*/
102         if(input == question[j]) {
103           /*推測用配列に代入*/
104           guess[j] = question[j];
105
106           /*既入力の配列に代入*/
107           for(int i = 0; i < 26; i++) {
108             if(input == used_char[i]) {
109               found = 1;
110               break;
111             }
112           }
113           if(!found) {
114             used_char[used_char_index++] = input;
115           }
116
117           /*クリア判定*/
118           if(strcmp(question, guess) == 0) {
119             printf("%s\n", guess);
120             printf("ゲームクリア\n");
121             flag = 0;
122             free(guess);
123           }
124           true++;
125         }
126       }
127
128       /*間違っていたら*/
129       if(!true) {
130         used_char[used_char_index++] = input;
131         wrong++;
132         true = 0;
133         /*ゲームオーバー判定*/
134         if(wrong > 6) {
135           flag = 0;
136           wrong = 0;
137           printf("答え : %s\n", question);
138           printf("ゲームオーバー\n");
139           free(guess);
140         }
141       }
142     }
143
144     /*繰り返しプレイの確認*/
145     printf("続けますか？(y/n)：");
146     input = getChar();
147     printf("\n");
148     char option = toupper(input);
149     switch(option)
150     {
151       case 'Y':
152         flag = 1;
153         for(int i = 0; i < 26; i++) {
154           used_char[i] = '\0';
```

```
155         }
156         used_char_index = 0;
157         continue;
158
159       case 'N':
160         printf("\nbye\n");
161         return 0;
162
163       default:
164         printf("\nInvalid option.\n");
165         flag = 0;
166         continue;
167     }
168   }
169
170   fclose(fp);
171   return 0;
172 }
173
174
175 char getChar(void) {
176   struct termio old_term, new_term;
177   char c;
178   /* 現在の設定を得る */
179   ioctl(0, TCGETA, &old_term);
180   /* 設定のコピーをつくる */
181   new_term = old_term;
182   /* 入力文字のエコーを抑止する場合 */
183   new_term.c_lflag &= ~(ICANON | ECHO);
184   /* エコーは止めない場合 */
185   //new_term.c_lflag &= ~(ICANON);
186   /* 新しい設定を反映する */
187   ioctl(0, TCSETAW, &new_term);
188   /* 1 文字入力 */
189   c = getchar();
190   /* 古い設定に戻す */
191   ioctl(0, TCSETAW, &old_term);
192   return (c);
193 }
194
195 char* tolower_str(char* s)
196 {
197   for (char* p = s; *p != '\0'; ++p) {
198     *p = tolower(*p);
199   }
200   return s;
201 }
202
203 void disp_info(char* guess, char* used_char, int wrong_count) {
204   printf("単語 : %s\n", guess);
205   printf("使われた文字 : ");
206   printf("%s\n", used_char);
207   printf("残り回数 : %d\n", (7 - wrong_count));
208   printf("文字を入力してください : ");
209 }
```