2023 年 5 月 24 日　提出

20-413
北野正樹

プログラミング演習 IIIA

Hangman 制作についての報告書

1. プログラムの概要

1.1 Hangman プログラムの説明

今回作成した hangman プログラムはその名の通り規定数で未知の単語を当てるというクイズ形式のゲームとなっている。規定回数で単語を当てるというとてもシンプルな内容であるが、Wordle などと名前を変えて世界で広く遊ばれているミニゲームである。今回の仕様では七回の間に単語を当てる仕様となっている。現在のプログラムでは 7 回で当てるということになっており、これはゲーム内では変えることができない。

英単語は TOEIC の頻出単語 1500 からランダムに出題される仕組みになっており、ファイルは実行プログラム直下においてある。ここのファイルパスを変えることで出題される単語を変えることができる。

ゲームルールとしてはとても単純で一回のターンで一文字英小文字を入力することができ、それを七回以内にすべてのわからない文字を当てるというものである。

1.2 本プログラムの開発環境

本プログラムの開発環境を下記に示す

・マシン：MacBook Pro2020M1

・OS：MacOS Ventura 13.4

・エディタ：Emacs

・言語：Clang

・コンパイラ：Apple clang version 14.0.3 (clang-1403.0.22.14.1)

1.3 本プログラムの動作環境

MacOS Ventura 13.4 以降　検証済

Debian GNU Linux（Mac での仮想環境）　検証済

Windows11（Mac での仮想環境）検証済

OS に依存しないヘッダーを使用して作成。

1.4 本プログラムの使用方法

本ソースコードを上記のコンパイラのバージョンと同じ、またはそれ以上のバージョンでコンパイルし、実行ファイルを生成する。その実行ファイルと同じ階層に単語のファイルを置き実行すると、ゲームがスタートする。設定メニューなどはなく実行した瞬間からゲームが開始する。ゲームに表示されているメニューについて説明する。

・単語：------

ここには一致する文字を入力すると文字が現れていく。文字数と穴食いの場所から単語を予想することができる。

・使われた文字：

ここには今まで使われた文字が表示される。今までというのはそのゲームの文字であり、累計ではない。ゲーム中は同じ文字を入力することはできないようになっている。

・残り回数：

ここにはその名の通りあと何回回答できるかがカウントされる。0 になるとゲームオーバーとなる。

・文字を入力してください：

ここのラインに予想した文字を入れる。予想した文字をタイプして、Enter キーを押すことで次のターンへ移ることができる。

ゲームオーバー、またはゲームクリアした際には正誤と正解単語が表示される。また、引き続きプレイするかを Y または N で選択する。Y を押すとまた新しい単語がロードされ新しくゲームを始めることができる。N を押すとゲームが終了する。

1.3 本プログラムの必要な変数

本プログラムの変数の管理は構造体などは用いずにグローバル変数として作成した。新規インスタンスを生成する必要がなく、変数を包括する必要がないと判断したためである。

char *correctWord: 正解の単語を格納するポインタ配列。ゲーム開始時に毎回ランダムな文字列が代入される。サイズは動的確保。

int wordLength: 正解単語の文字列長を格納しておく変数。

int remainCount: 残りの回答数を格納しておく変数。

char usedChar[LIMIT_TIMES+1]: 使用した単語を格納しておく配列。サイズは回答数を用いて静的確保。

char *correctFlag: ゲーム内で出力するためのポインタ配列。ここの配列をその都度変更してゲームに出力している。サイズは動的確保

Boolean isCorrect: ゲームクリアかゲームオーバーかを判定する変数。

Boolean isRestart: ゲームを再スタートするかどうかを判定する変数。

1.5 各関数についての説明

1.5.1 main 関数

本関数では、ゲーム全体の流れを司っている。入力に対して出力、判定などほとんどの処理をここで処理している。

1.5.2 input 関数

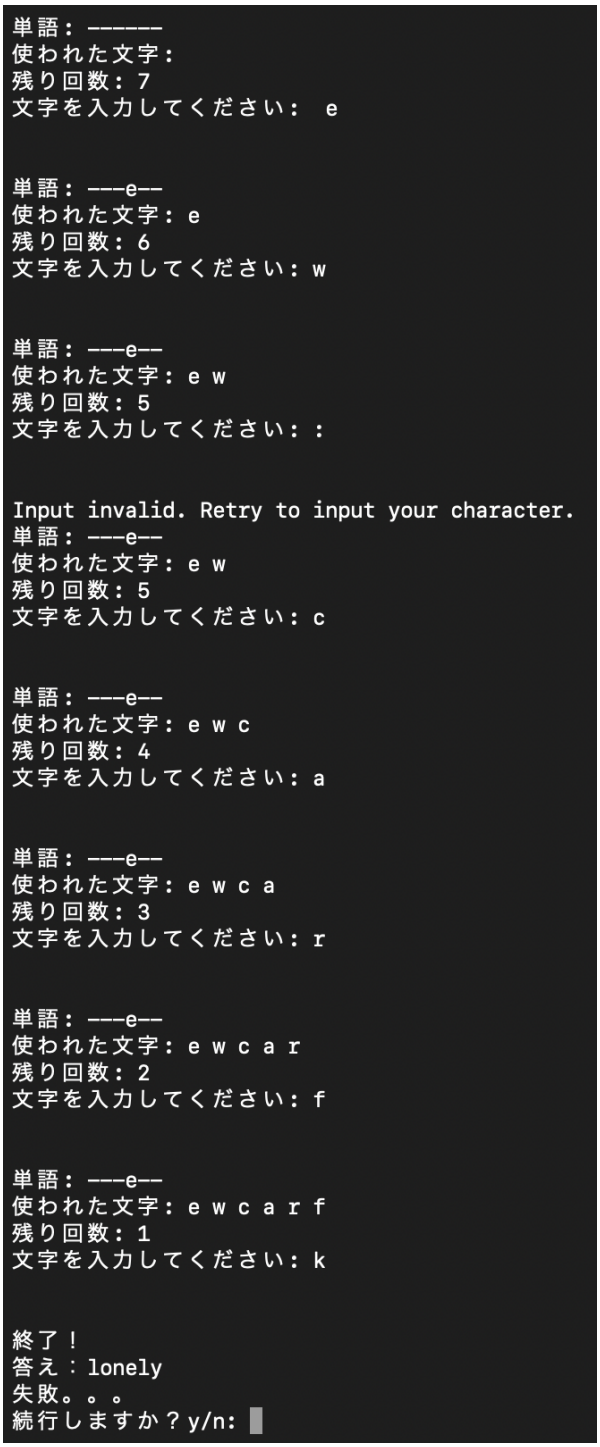
本関数では、コマンドラインから一文字受け取る。エスケープシーケンスを用いて無駄な文字が入らないようになっている。

1.5.3 initialize 関数

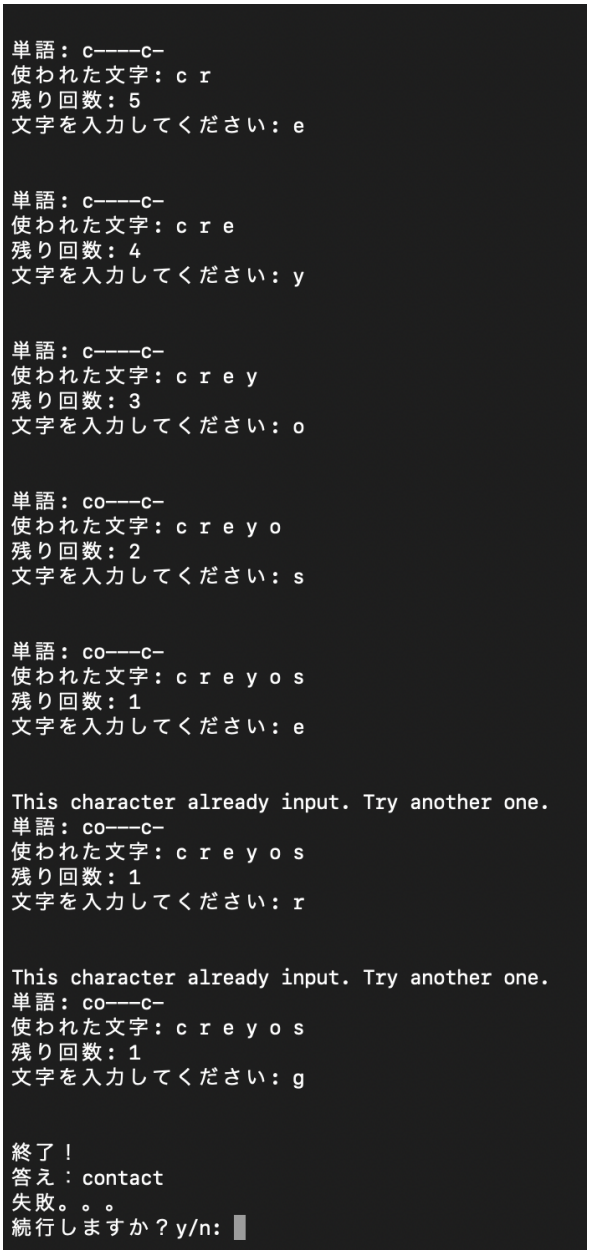
本関数では、ゲーム開始時のランダムシードの初期化や各変数の処理か処理を行っている。

1.6 本プログラムの検証

本プログラムの検証を行ったため、下に示す。



このように失敗する旨の表示がある。また、英小文字ではない場合はもう一度入力させるようになっている。



このように同じ文字を入力するともう一度入力するように言われる。

1.7 本プログラムの課題

本プログラムでは変数の動的割り当てなどにより少し処理が複雑化してしまった。また、ほとんどの処理をメイン関数内に収めたことで、少し可読性が悪くなってしまった。もう少し関数にし可読性をあげ、今後も拡張しやすいようなプログラムを書けるようにするべきだと感じた。

1.8 ソースコードの全容

今回作成したソースコードを以下に示す。

```
    /* 入力された文字が正解文字列に含まれていた場合、表示用配列に反映する */
    i = 0;
    while (correctWord[i] != '\0') {
      if (correctWord[i] == buffer) {
        correctFlag[i] = buffer;
      }
      i++;
    }


    /* 入力した文字がすでに入力されているかを判定。もし重複していたら最初に戻る */
    i = 0;
    Boolean isDuplicate = FALSE;
    while (usedChar[i] != '\0') {
      if (usedChar[i] == buffer) {
        isDuplicate = TRUE;
        break;
      }
      i++;
    }
    if (isDuplicate == FALSE) {
      usedChar[7 - remainCount] = buffer;
      remainCount--;
    } else {
      printf("This character already input. Try another one.\n");
      continue;
    }


    /* 表示用文字列と正解文字列が一致していた場合、プログラムを終了させる。そうでなければ続行する */
    i = 0;
    Boolean isContinue = FALSE;
    while (correctFlag[i] != '\0') {
      if (correctFlag[i] == '-') {
        isContinue = TRUE;
      }
      i++;
    }
    if (isContinue == FALSE) {
      isCorrect = TRUE;
    }
  }

  /* 最終結果を出力する。 */
  printf("終了！\n");
  printf("答え：%s\n", correctWord);
  if (isCorrect == 1) {
    printf("成功！\n");
  } else {
    printf("失敗。。。\n");
  }

  /* リトライするかを聞く */
  char buffer = 'A';
  while (buffer != 'y' && buffer != 'Y' && buffer != 'n' && buffer != 'N') {
```

```
      printf("続行しますか？y/n: ");
      buffer = input();
    }
    if (buffer == 'y' || buffer == 'Y') {
      isRestart = TRUE;
      isCorrect = FALSE;
      remainCount = LIMIT_TIMES;
    } else {
      isRestart = FALSE;
    }
  } while (isRestart == TRUE);
  free(correctWord);
  free(correctFlag);
  return 0;
}

/** 文字を一文字入力するための関数 */
char input() {
   char buffer;
   scanf("%c", &buffer);
   scanf("%*[^\n]");
   scanf("%*c");
   printf("\n\n");
   return buffer;
}

/** ゲームに必要な変数や初期化をする関数 */
void initialize() {
  int i;
  char buffer[512+1];
  char word[64+1];

  free(correctWord);
  free(correctFlag);

  /* ファイルオープン */
  FILE *fp = fopen("toeic1500_utf.dat", "r");
  if (fp == NULL) {
    printf("word file cannot open!\n");
    exit(-1);
  }
  /* ここでファイルから読み出した単語をランダムにロードする */
  srand((unsigned int) time(NULL));
  for (int i = rand() % 1500; i >= 0 && fgets(buffer, 512+1, fp) != NULL; i--);
  int spaceCount = 0;
  int wordCount = 0;
  i = 0;
  /* ロードしてきた単語を整形し、正解単語変数に格納する */
  while (buffer[i] != '\0') {
    if (buffer[i] == ' ') {
      spaceCount++;
      i++;
      continue;
    }
    if (spaceCount == 1) {
      word[wordCount] = buffer[i];
```

```
  for (int i = rand() % 1500; i >= 0 && fgets(buffer, 512+1, fp) != NULL; i--);
  int spaceCount = 0;
  int wordCount = 0;
  i = 0;
  /* ロードしてきた単語を整形し、正解単語変数に格納する */
  while (buffer[i] != '\0') {
    if (buffer[i] == ' ') {
      spaceCount++;
      i++;
      continue;
    }
    if (spaceCount == 1) {
      word[wordCount] = buffer[i];
      wordCount++;
    }
    i++;
  }

  i = 0;
  while (word[i] != '\0') i++;
  wordLength = i;
  correctWord = (char*) malloc(sizeof(char) * (wordLength+1));
  i = 0;
  while (word[i] != '\0') {
    correctWord[i] = word[i];
    i++;
  }

  /* 以下は各変数の初期化 */
  correctFlag = (char*) malloc(sizeof(char) * (wordLength+1));
  remainCount = LIMIT_TIMES;
  isCorrect = FALSE;
  isRestart = FALSE;
  for (i = 0; i < LIMIT_TIMES; i++) {
    usedChar[i] = '\0';
  }
  for (i = 0; i < wordLength; i++) {
    correctFlag[i] = '-';
  }
}
```

```c
/** 各ヘッダーのインクルード */
#include <stdio.h>  // 標準入出力
#include <stdlib.h> // system()を呼び出すのに必要
#include <time.h>

#define LIMIT_TIMES 7  // 回数制限の定義

/** データ型宣言 */
typedef enum Boolean { // フラグを構成する列挙
  FALSE,
  TRUE
} Boolean;

/** hangman根幹変数の宣言 */
char *correctWord;                          // 正解単語を格納する配列
int wordLength;                             // 正解単語の文字列長を格納する変数
int remainCount;                            // 残りの回答回数を格納する変数
char usedChar[LIMIT_TIMES+1];                       // 使われた単語を格納するための配列
char *correctFlag;                          // 画面に状況を表示するための文字列を格納するための配列
Boolean isCorrect;                              // 正誤判定をするためのフラグ
Boolean isRestart;


/** 関数プロトタイプ宣言 */
char input();
void initialize();

/** メイン関数 */
int main(void) {
  /** メインループ。残り回数が0になるまでまたは正解するまで繰り返す */
  do {
    system("clear");
    initialize();
    int i;
    while (remainCount > 0 && isCorrect == FALSE) {
      /* 表示の処理 */
      printf("単語： ");
      for (i = 0; i < wordLength; i++) {
        printf("%c", correctFlag[i]);
      }
      printf("\n");
      printf("使われた文字： ");
      i = 0;
      for (i = 0; i < (7 - remainCount); i++) {
        printf("%c ", usedChar[i]);
      }
      printf("\n");
      printf("残り回数: %d\n", remainCount);
      printf("文字を入力してください： ");
      char buffer = input();

      /* 英小文字以外が入力されたときは警告文を出して最初に戻る */
      if (buffer < 'a' || buffer > 'z') {
        printf("Input invalid. Retry to input your character.\n");
        continue;
      }
```