単語（文字列）の出現数を数えよう

Cプログラミング演習

countAnimals.c

何をするものか：ファイル中に現れる動物名(単語)の種類とそれぞれの出現数を答える

入力はなにか：ファイル　animals.txt

出力はなにか：単語と出現数のリスト(の表示)

方法：(1)ファイル中に現れると予想される単語リストを作成

(2)ファイルから読み込んだ単語を配列に記憶

(3) 単語リスト中の単語それぞれに対し、配列で記憶したファイル中の単語それぞれと照合し、出現数をカウント。それを表示する

countAnimals.c の作成

```
#include <stdio.h>
// 文字列操作に必要なヘッダファイルを取り込む
                                    #include  <string.h>

// ファイルの行数の最大値を仮に100とする
#define  LINES 100
// 動物の種類
#define NUM 10
// 動物名の最大長さ
#define  LEN  10
// 読み込み対象のファイル名
#define InFile "animals.txt"
```

問題１: ファイルの行数がこれより多いかも

問題２: 動物（単語）の種類がこれより多いかも

検討事項: 動物（単語）の最大長さは妥当か？

countAnimals.c の作成(2)

```
int main(void) {
  // 配列 Animals と str 他必要な変数の宣言                    LEN
  // 配列 Animals の宣言---ここを直すこと
  // 候補は: いぬ", "ねこ", "うさぎ", "くま", "さる", "たぬき",
            "きつね", "ぱんだ", "ねずみ", "わに
     char Animals[][        ] = { "いぬ", "ねこ", "うさぎ", "くま", "さる",
                        "たぬき", "きつね", "ぱんだ", "ねずみ", "わに"  };
     char str[LINES][LEN];
     int m, i, j , n=0;
```

注釈：
- str にファイル中の単語(長さLEN)を記憶
- m, i, j はあとで使用(i, jはカウンタ変数)
- nは読み込み行数の記憶用

countAnimals.c の作成(3)

```
// ファイルから読み込み準備
FILE *fp = fopen(InFile, "r");
```

注釈： ファイルから読み込むには
FILE変数が必要

```
// 1. ファイルから読み込んだ文字列を配列strに記憶
//    その時、読み込んだ単語の個数を変数 n で記憶すること
// ファイルを閉じる
while ( fscanf(fp, "%s", str[n]) != EOF) {
        n++;
}
fclose(fp);
```

注釈： ファイル読み込みの定番パタン
変数 n で読み込んだ行数を記憶
かつ、読み込んだ文字列をstr[n]に記憶

countAnimals.c の作成(4)

```
// 2.候補の動物名の一個ずつに対し(繰り返し)
for (i=0; i < NUM; i++) {
 //   一致した個数を記憶する変数mに0をセット
        m = 0;
        // str の要素それぞれに対して(繰り返し)
        for (j=0; j < n; j++) {
            //  Animals[i]とstrの内容が一致すれば、m++ とする
            if (strcmp(Animals[i], str[j]) == 0)
                m++;
        }
        //  m>0なら、動物名とmの値を表示
         if (m > 0)
            printf("%10s : %5d ¥n", Animals[i], m);
    }
 // ここまでが2の内容
 return 0;
}
```

注釈　代入のことを「変数に値をセット」ということも多い

注釈: strcmpで文字列を比較、一致したら0

ポイント: LINESではなく実際に読み込んだ行数であるnを書くこと

必ずコンパイルして実行すること

wordファイルとしてプログラムを提出する者がいるが、これはよくない
必ず テクストファイル、拡張子は .c (もしくは .cpp) とすること

またコンパイルを必ずして、実行してみること

今まで見た不思議なケース：
(1) コンパイルするとエラーが出るのに実行結果が添付されている
(2) ソースファイルだけ提出されている。
　　コンパイルしてエラーが出るが、一箇所（それも1字）直せば動いた
(3) (例えば)「m>0なら...せよ」と指定したのに、提出されたプログラムは
　if (m != 0) ....　となっている。しかも２人もいる！

countAnimals.c の実行結果

以下のようなものが表示されるはず

いぬ：　　9
ねこ：　14
うさぎ：　1
くま：　　1
さる：　　4
たぬき：　1
きつね：　1
わに：　　1

countAnimals.cの問題検討

countAnimals.c では次のようなマクロを使っていた:

// ファイルの行数の最大値を仮に100とする
#define LINES 100
⇒ 問題1.ファイルの行数がこれより多い場合を考慮していない

// 動物の種類
#define NUM 10
⇒ 問題２: 動物（単語）の種類がこれより多い場合を考慮していない
しかも、ファイルに現れる動物(単語)を予め決めていた
… そのため、２種類の単語を見落としている

// 動物名の最大長さ
#define LEN 10
⇒ 検討事項: 事前に扱う単語の文字数を調べる必要がある

countAnimals.c の改良

問題1.ファイルの行数がこれより多い場合を考慮していない

解決策：ファイルに書かれた単語を配列で記憶するのをやめる
⇒ ファイルから一行読み込んだら、その場で単語リストと比較し、
出現数をカウントする。こうすれば、ファイル全部の単語を記憶する
必要がなくなる

問題２: 動物（単語）の種類がこれより多い場合を考慮していない
しかも、ファイルに現れる動物(単語)を予め決めていた

解決策：ファイルに現れた単語のリストを作成する
そのリストにない単語が出てきたら、単語リストに追加する
なお、これには、今までに出現した「単語の種類数」（単語リストの長さ）
を記憶する変数が必要

countAnimals.c の改良

新たな問題: main関数だけで仕事をやりすぎている

方法：(1)ファイル中に現れると予想される単語リストを作成
(2)ファイルから読み込んだ単語を配列に記憶
(3) 単語リスト中の単語それぞれに対し、配列で記憶したファイル中の単語それぞれと照合し、出現数をカウント。それを表示する

仕事を分ける: (1) はmain関数の変数の宣言　－ 大域変数としてもよい
(2)は、ファイルから単語を読み込む関数(getWordsFromFile)を作ってもよい
(3)がこの中で一番複雑—だから仕事を分ける
「単語リスト中の単語それぞれに対し
配列で記憶したファイル中の単語それぞれと照合」
を行う関数findAnimalを作る

getWordsFromFile

入力：ファイル名(fName)と、ファイル中の単語を記憶する配列(str)
出力：ファイルから読み込んだ単語数(ファイルの行数）
処理：

(1) FILE変数fpを宣言。
　また整数型変数n に0を初期値として与える
(2)ファイルを開く(fpにfopen(fName,”r”)の値をセット）
(3) EOFとなるまで以下を繰り返す
　fscanfを用いて str[n] にファイルから読み込んだ文字列を記憶
　nに1を足す
(4) nの値を返す

countAnimals3.c

何をするものか：ファイル中に現れる動物名(単語)の種類とそれぞれの出現数を答える

入力はなにか：ファイル animals.txt

出力はなにか：単語と出現数のリスト(の表示)

方法：(1)ファイル中に現れると予想される単語リストを作成

~~(2)ファイルから読み込んだ単語を配列に記憶~~

(3) ファイルから一行ずつ読み込み、読み込むたびに単語リストと照合。合致した単語に対応する「頻度リスト」を更新していく

countAnimals3.c を作る

```
#include <stdio.h>
// 文字列操作に必要なヘッダファイルを取り込む
#include <string.h>
// 動物の種類
#define NUM 10
// 動物名の最大長さ
#define  LEN  8
// 読み込み対象のファイル名
#define InFile "animals.txt"

// 大域変数として「文字ポインタ(=文字列)の配列」Animalsを用意
char* Animals[] = {"いぬ","ねこ","うさぎ","くま","さる", "たぬき",
                   "きつね","ぱんだ","ねずみ","わに"};
```

注意：ここでは単語リスト Animals を大域変数としている

countAnimals3.c を作る(2)

```
// 文字ポインタ(=文字列)の中身と一致するものが Animalsにあれば
// その添字を返す.　なければ -1 を返す
int findAnimal(char* line) {
   int i;
   for (i=0; i < NUM; i++) {
        if (                                    )     // Animals[i]とlineを比較
              return i;    // 一致した場合は その添字 i を返す
   }
   return -1;  // 見つからなかった場合は -1を返す
}
```

countAnimals3.c を作る(3)

```
int main(void) {
   int i, result, freq[NUM];   // Animals[i]の出現数を頻度リスト freq[i]で記憶
   FILE *fp;
   char str[LEN];              // 一行だけファイルを読み込む
   // freqの要素をすべて 0 にセットする（繰り返しを用いる）

  // InFileから読み込みの準備

  // InFileから一行ずつ読み込み(strに記憶)
  // findAnimalを用いて登録されている動物かどうかを調べ(戻り値をresultにセット)
  // resultの値が 0以上なら(つまり「あった」場合), freq[result]の値を更新する

   for(i=0;i<NUM; i++)     // 答えの表示
         if (freq[i] > 0)    printf("%s の出現数 = %d¥n", Animals[i], freq[i]);
   fclose(fp);  return 0;
}
```

できたらコンパイルして実行

以下の様な表示が出るはず

いぬ の出現数 = 9
ねこ の出現数 = 14
うさぎ の出現数 = 1
くま の出現数 = 1
さる の出現数 = 4
たぬき の出現数 = 1
きつね の出現数 = 1
わに の出現数 = 1

countAnimals3.c の検討

countAnimals.c では、ファイルから単語を全部読み込み、配列にいれた

countAnimals3.c では読み込むたびに単語リストと比較することにより、読み込んだ単語を記憶しなくてすんだ

⇒ メモリの効率的な使用ができている

「問題１」の解決

countAnimals.c でも countAnimals3.c でも単語リストを予め作成

「問題２」は解決されていない

ただし、

countAnimals3.c では、読み込んだ単語が「単語リスト」にない場合がわかるようになっている--- findAnimal が マイナスの値を返す

⇒ これを使って単語リストに新たな単語を加えることが可能になる！

wordCollect2.c

単語リストを予め作らず、ファイルから単語を読み込むたびに
それが初出である（単語リストにない）場合に、単語リストに
追加するようにすれば、「問題２」も解決されるはず。

そのプログラムを wordCount.c とする

それを作る前に、ここでは「ファイルに出現する単語の種類」
（異なり語)を答えるプログラム wordCollect2.c を作って、考えが
正しいことを確かめよう。そして、その後で wordCount.c を作る

wordCollect2.cを作る

```
#include <stdio.h>
// 文字列操作に必要なヘッダファイルを取り込む
#include <string.h>

// 単語の種類(最大数)
#define NUM 100

// 単語の最大長さ (漢字の場合、文字数×2+1)
#define  LEN  21

// 読み込み対象のファイル名
#define InFile "animals.txt"

// 大域変数として 文字列の配列 terms を用意する
char terms[NUM][LEN];     // LENの長さの単語をNUM個、記憶する
```

注釈：
terms にファイル中に現れる単語を記録

wordCollect2.cを作る(2)

```
int findTerm(char str[], int number) {
   // 繰り返しを用いる
   // 文字列 str に一致する terms の要素(要素数がnumber)が
   //あれば 1を返し、そうでなければ -1 を返す




}
```

注釈：
前に作った findAnimals 参照

wordCollect2.cを作る(3)

```
int main(void) {
    int i, number = 0;  // number: 記憶した単語数
    char str[LEN];    // ファイルから読み込む文字列を記憶
    // InFile からデータを読み込む準備をする
    FILE *fp = fopen(InFile, "r");
    // ファイルから読み込み、str で記憶する
     while ( fscanf(fp, "%s", str) != EOF) {
       // str の内容がterms(今までに記憶した単語数は numberで記憶している)
       // の中にあるかどうかを調べる: findTerm(str, number) を用いる
       // もしもなければ terms に str の中身を登録し(strcpyを用いる)て
       //                    numberの数を１増やす
}
```

wordCollect2.cを作る(4)

```
 fclose(fp);
   // 結果を報告: InFileの中にある動物を表示する
    printf("読み込まれた単語の種類は %d 個、その内訳は...¥n", number);
    for(i=0; i<number; i++) {
        printf("%s¥n",terms[i]);
    }
    return 0;
}
```

できたらコンパイルして実行

次のような表示が得られるはず

読み込まれた単語の種類は 10 個、その内訳は...
いぬ
ねこ
さる
わに
とら
たぬき
うさぎ
へび
くま
きつね

さあ、wordCount.c をつくろう

これは countAnimals3.c の発展形

大きな違いは：

countAnimals.c の「問題２」の解決
　　つまり、予め単語リストを作っておかない！

⇒ ファイルを読み込んで、既出の単語であれば、その単語の頻度を更新
　　既出でなければ、単語リストに登録し、その単語の頻度を１とする
これを繰り返す…

つまり、wordCount.c = countAnimals3.c + wordCollect2.c – 余計なもの