第 12 回 C++ プログラミング実験 II

出題日：2022 年 6 月 27 日（月）

課題 1 （提出期限：6 月 30 日（木）24:00）

ファイルから単語の一覧を読み込み，各単語に “com” という文字列が含まれるかを判定するプログラムを考える．

ソースコード ex12-1.cpp は，コマンド引数で指定したファイルを読み込み，ファイル内の単語を vector 配列の words へ格納した後，各単語に “com” という文字列が含まれるかを判定して，“com” が含まれる単語を表示するプログラムである．

文字列（std::string）から特定の文字列（“com”）が含まれるかを判定する方法として，以下の 2 つが考えられる．

方法 1　std::string のメンバ関数 find を使って文字列 “com” を探し，見つかったか否かで判定する．
方法 2 「“com” が含まれる単語」を正規表現で表現し，そのパターンにマッチするかで判定する．

上記の 2 つの方法で，各単語に “com” という文字列が含まれるかを判定して表示するプログラムとなるよう，ex12-1.cpp を埋めてプログラムを完成させなさい．また，完成したプログラムをコンパイルし，実行例と同じ結果になるかを確認しなさい．

※ CoursePower から ex12-1.cpp と input.txt をダウンロードして使用すること．

実行例

```
./a.out input.txt
Method 1.
company recommend come become common
Method 2.
company recommend come become common
```

ソースコード 1　ex12-1.cpp

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <regex>

int main(int argc, char *argv[]) {
    // コマンド引数の確認
    if (argc < 2) {
        std::cout << "usage: " << argv[0] << " filename\n";
        return 1;
    }
    // ファイル読み込み
    std::string filename{argv[1]};
    std::ifstream fin(filename.c_str());
    if(!fin) {
        std::cout << "file open error\n";
        return 1;
```

```
    }
    // ファイル内の単語をvector配列のwordsへ格納する
    std::string w;
    std::vector<std::string> words;
    while(fin >> w) { words.push_back(w); }

    std::cout << "Method 1.\n";
    // [方法1] findを用いて単語に"com"が含まれるか判定し，含まれている場合は出力する
    std::string s{ "com" };
    for (auto word : words) {
        //
        // ここを埋めてプログラムを完成させよ
        //
    }
    std::cout << "\n";

    std::cout << "Method 2.\n";
    // [方法2] 正規表現を用いて"com"が含まれるか判定し，含まれている場合は出力する
    std::regex r{ /*ここに当てはまる正規表現を書く*/ };
    for (auto word : words) {
        //
        // ここを埋めてプログラムを完成させよ
        //
    }
    std::cout << "\n";

    return 0;
}
```

課題 2　(提出期限 6 月 30 日 (木) 24:00)

第 1 回レポート課題で扱ったような有限オートマトンによって表現できる文字列は正規表現でも表現することができる．例えば，図 1 のような有限オートマトンが受理可能な文字列は正規表現を用いて ab^*c と表せる．そこで，ex12-2.cpp では vec1, vec2 に格納されている文字列がそれぞれ図 2, 3 のような有限オートマトンで受理されるかどうかを正規表現を用いて判定している．空欄を埋めてプログラムを完成させない．

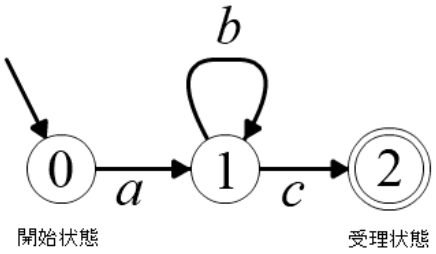

図 1　有限オートマトン 1

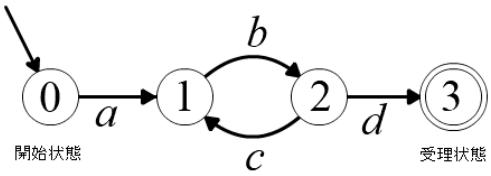

図 2　有限オートマトン 2

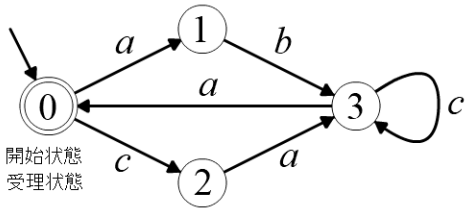

図 3　有限オートマトン 3

ソースコード 1　ex12-2.cpp

```cpp
#include <iostream>
#include <vector>
#include <regex>

        /*
          空欄1 文字列の受理可能性を順番に判定するcheck関数を作成しなさい.

        */

int main(){
    std::vector<std::string> vec1{"a", "abc", "abd", "abd",
                                  "abcbcd", "babd", "abcbd"};
    std::vector<std::string> vec2{"", "abcac", "aba", "caca",
                                  "cacaca", "abcabc", "abcacaca"};

    std::regex r1{ /* 空欄2 */ }; //図2の有限オートマトンを受理する正規表現
    std::regex r2{ /* 空欄3 */ }; //図3の有限オートマトンを受理する正規表現

    check(r1, vec1);
    check(r2, vec2);
}
```

実行結果

```
$ g++ -std=c++17 ex12-2.cpp
$ ./a
----------
a: reject
abc: reject
abd: accept
abd: accept
abcbcd: reject
babd: reject
abcbd: accept
----------
: accept
abcac: reject
aba: accept
caca: accept
cacaca: reject
abcabc: reject
abcacaca: accept
```

課題３（提出期限：6 月 30 日（木）24:00）

prac3.txt ファイルから(1) 先頭が T または B である単語 と (2)末尾が e である単語をそれぞれ抽出して，出現回数を数え，アルファベット順（string の小さい順）に表示するプログラムを作成せよ(ex12-3_skel.cpp を利用)。最後に，ファイルの行数，(1) の条件を満たした単語の総数，(2)の条件を満たした単語の総数，(1)の条件を満たした単語の種類数，(2)の条件を満たした単語の種類数をそれぞれ表示すること。

ヒント：(2)を実現するために単語の直後のスペースや改行を使用する

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <regex>

int main() {
    std::cout << "----  Read file ----" << "¥n";
    std::ifstream fin("prac3.txt");
    if (!fin) { std::cerr << "file open error¥n"; return 1; }
    std::string line;
    int w1count{ 0 }, w2count{ 0 }, linenum{ 0 };   // 単語数，行数

    //空欄１：(1)冒頭はTかBである単語の正規表現

    //空欄２：(2)末尾はeである単語の正規表現

    std::map<std::string, int> wlist1{}; // 空欄１に対応する単語と出現回数をpairとするmap
    std::map<std::string, int> wlist2{}; // 空欄２に対応する単語と出現回数をpairとするmap

    while (getline(fin, line))  // ファイルの終わりまで，ファイルから１行ずつlineへ読む
    {
        ++linenum;  //行数をカウント

        //空欄３：正規表現に一致する単語の格納と出現回数の数え上げ


    }

    // 空欄４：空欄１の正規表現に一致する単語と出現回数を表示（単語のアルファベット順）

    std::cout <<"==========================" << "¥n";
    // 空欄５：空欄２の正規表現に一致する単語と出現回数を表示（単語のアルファベット順）


    /* 空欄６： ファイルの行数，(1)の条件を満たした単語の総数，(2)の条件を満たした単語の総数，
    (1)の条件を満たした単語の種類数，(2)の条件を満たした単語の種類数をそれぞれ表示* /
}
```

実行例

```
----  Read file ----
But : 1
The : 2
==============================
The  : 2
before  : 1
came  : 1
come  : 1
face  : 1
gentle  : 1
have  : 5
hope  : 1
house  : 1
live  : 1
the  : 5
there  : 1
time  : 1
true  : 1
voice  : 1
---------
ファイルの行数 : 19
(1)の条件を満たした単語の総数 : 3
(2)の条件を満たした単語の総数 : 24
(1)の条件を満たした単語の種類数 : 2
(2)の条件を満たした単語の種類数 : 15
```