C++プログラミング III 実験／（後期）2022 年

出題日　2022年10月13日（木）

履修者は自分の担当者に CoursePower で提出．提出〆切 10 月 19 日（水）**24:00**

課題１：CoursePower から prac03_skel.cpp をダウンロードし使用すること．

transition_matrix.txt に複数の遷移行列が記載されており，これらの遷移行列の行方向の和が 1 になるかチェックするプログラムを作成したい．prac03_skel.cpp ではで部分的に作成されており，main 関数が動作するようにプログラムの必要な部分を追記しなさい（main 関数部分も追記する）．

ただし，transition_matrix.txt に記載された遷移行列の個数および行列の大きさは事前に分からないとする．このため，遷移行列のクラス Transp は，データメンバとして行列の名前，行列の大きさ，行列の値を格納するための 2 次元配列を動的に構築するためのポインタをとる．

```
// transition_matrix.txt の中身の例
// <TRANSP>  行列のサイズ  行列の名前
<TRANSP> 5 a1
0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 3.771155e-01 6.228845e-01 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 7.209600e-01 2.790401e-01 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00 6.577078e-01 3.422922e-01
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
<TRANSP> 3 b1
0.000000e+00 1.000000e+00 0.000000e+00
0.000000e+00 9.588912e-01 4.110880e-02
0.000000e+00 0.000000e+00 0.000000e+00
<TRANSP> 4 c1
0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 8.989010e-01 1.010990e-01 0.000000e+00
0.000000e+00 0.000000e+00 5.848860e-02 9.415114e-01
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

```
// 実行例
$./a.exe
a1 の 1 行目は正常でした
a1 の 2 行目は正常でした
a1 の 3 行目は正常でした
a1 の 4 行目は正常でした
b1 の 1 行目は正常でした
b1 の 2 行目は正常でした
c1 の 1 行目は正常でした
c1 の 2 行目は正常でした
c1 の 3 行目は正常でした
```

```cpp
//prac03_skel.cpp
#include <iostream>
#include <fstream>
#include <cstdlib>
using std::cout, std::string, std::endl;
class Transp
{
    string name;          //行列の名前
    int size;             //行列の大きさ
    double **probability; //行列の値を格納する 2 次元配列のポインタ
public:
    Transp() { size = 0; } //コンストラクタ
    void checkprob()
    { //行列の行の数値の和が 1 になっているかチェックする関数
        double sum;
        if (size == 0)
        { // size が 0 のとき警告を出して止める
            std::cerr << "size が 0" << endl;
            exit(EXIT_FAILURE);
        }
        else
        {
            for (int i = 0; i < (size - 1); i++)
            { //最後の行は 0 が入っているためチェックしない(遷移行列であるため)
                sum = probability[i][0];
                for (int j = 1; j < size; j++)
                {
                    sum += probability[i][j];
                }
                if ((sum < (1 - 0.00001e-01)) || ((1 + 0.00001e-01) < sum))
                { //誤差を許容する
                    std::cerr << i + 1 << "行目の値が 1 でない" << endl;
                    exit(EXIT_FAILURE);
                }
                else
                {
                    cout << name << "の" << i + 1 << "行目は正常でした¥n";
                }
            }
        }
    }
```

```
    friend void readtransitionmatrix(string, Transp[]);
};

//引数で指定されたファイルにある<TRANSP>の数をリターンする
int gettranskosuu(string filename)
{
    string line;
    std::ifstream fin(filename);
    if (!fin)
    { //エラー処理
        std::cerr << "エラー:ファイルを開けません" << filename << "¥n";
        exit(EXIT_FAILURE);
    }
    int num{0};
    while (fin >> line)
    { //<TRANSP>の数を調べるために空読み
        if (line == "<TRANSP>")
        {
            num++;
        }
    }
    return num;
}
// ファイルの情報をメモリをとりながら transmatrix に読み込む
/* ファイルの中身:<TRANSP> 行列サイズ 名前
① string line は<TRANSP>の文字列のチェックおよび行列の名前用
② int sizematrix は行列のサイズを保存および行列の数値用
*/
void readtransitionmatrix(string filename, Transp transmatrix[])
{
    string line;    //ファイルからの読み込み用
    int sizematrix; //行列の大きさ

    std::ifstream fin(filename);
    if (!fin)
    { //エラー処理
        std::cerr << "エラー:ファイルを開けません" << filename << "¥n";
        exit(EXIT_FAILURE);
    }
    int num{0}; // transmatrix[num]でも可
    while (fin >> line)
```

```
{
    if (line != "<TRANSP>")
    { //エラー処理
        std::cerr << "エラー:<TRANSP>がありません" << filename << "¥n";
        exit(EXIT_FAILURE);
    }
    if (fin >> sizematrix)
    {
        /* 追記する */
        /* transition_matrix.txt 内の num 個目の行列に対して:
           transmatrix のデータメンバに行列サイズを保存する
           行列の値を格納する 2 次元配列の動的確保(講義資料 p.22 以降参照)
           行列の値を格納する 1 次元配列の動的確保
        */
    }
    else
    {
        std::cerr << "エラー:行列の大きさがありません" << filename << "¥n";
        exit(EXIT_FAILURE);
    }
    if (fin >> line)
    {
        // 追記する
        // transition_matrix.txt 内の num 個目の行列に対して, transmatrix のデータメンバに行列
の名前を保存する
    }
    else
    {
        std::cerr << "エラー:名前がありません" << filename << "¥n";
        exit(EXIT_FAILURE);
    }
    for (// 追記する)
    {
        for (// 追記する)
        {
            if (// 追記する)
            {// 処理はなし
            }
            else
            {
                std::cerr << "エラー:行列の値がありません" << filename << "¥n";
```

```
                exit(EXIT_FAILURE);
            }
        }
    }
    // 必要な処理を追記する
  }
}

int main()
{
    string inputfile = "transition_matrix.txt";// 読み込むファイル
    Transp *transmatrix;

    // 必要な transmatrix(遷移行列)の数を求めて、numtrans に代入
    int numtrans = gettranskosuu(inputfile);

    // 上で求めら transmatrix 分の領域を確保する
    // 追記する

    // ファイル内の数値の読み込み
    readtransitionmatrix(inputfile, transmatrix);

    // transmatrix の検証
    for (int i = 0; i < numtrans; i++)
    {
        transmatrix[i].checkprob();
    }
    return 0;
}
```