第 6 回 C++ プログラミング I　演習

2021 年 11 月 5 日（金）

課題 1 （提出期限：11 月 5 日（金）15:00）

2 次元配列や 3 次元配列を実行例のように表示するプログラムを作りたい．

下に示したプログラム（ex6-1.cpp）は，2 次元配列 x を初期化し，2 次元配列 x の中の 0 を 5 に置き換えてから，x の各要素を表示する．次に，3 次元配列 y を初期化し，x と同じように各要素を表示する．

① 2 次元配列 x の中の 0 を 5 に置き換える処理，② 2 次元配列 x の各要素を表示する処理，③ 3 次元配列 y の各要素を表示する処理の 3 つは未完成である．実行例を参考に，下に示したプログラム（ex6-1.cpp）を完成させなさい．

ソースコード（ex6-1.cpp）

```cpp
#include <iostream>
#include <vector>
using std::vector, std::cout;

int main() {
    // 2次元配列 x の初期化
    vector<vector<int>> x { {1,4},{2,0},{3,6} };

    /*
    ① 2次元配列 x 中の 0 を 5 に置き換えるプログラムを作成せよ．
    */
    x[?][?] = 5;     // ? に適切な数値を入れよ．

    cout << "1st output: ";
    /*
    ② 実行例に合うように2次元配列 x の各要素を表示するプログラムを作成せよ．
    */
    cout << "\n";

    // 3次元配列 y の初期化
    vector<vector<vector<int>>> y { {{1,1},{4,4}},{{2,2},{5,5}},{{3,3},{6,6}} };

    cout << "2nd output: ";
    /*
    ③ 実行例に合うように3次元配列 y の各要素を表示するプログラムを作成せよ．
    */
    cout << "\n";

    return 0;
}
```

実行例

```
./ex6-1.exe
1st output: 1 2 3 4 5 6
2nd output: 1 2 3 4 5 6 1 2 3 4 5 6
```

課題 2 (提出期限 11 月 7 日 (日) 23:59)

三次元空間における複数の点の位置を二次元配列で記録し、添え字が連続である二つの点の間の距離を求めたい。以下の実行結果をみて、ex6-2.cpp にコードを追加して，プログラムを完成させなさい。

ソースコード: ex6-2.cpp

```
#include <iostream>
#include <vector>
#include <cmath>

using std::vector, std::cout;
int main() {

        /*
        ２次元配列を宣言しなさい
        */

        vector<double> point1{ 0.0, 0.3, 1.8 }, point2{ 1.0, 1.0 ,1.0 },
                       point3{ 5.5, 6.2, 8.4 }, point4{ 5.9, 2.0, 8.0 },
                       point5{ 3.6, 8.2 ,9.7 }, point6{ 14.3, 12.9, 15.0 };
        /*
        point1～point6を宣言した２次元配列に追加しなさい
        */

        /*
        添え字(1～6)が連続である二つの点の間の距離を求めて、出力しなさい
        */
}
```

実行結果；

```
The distance between point1 and point2 is 1.45945
The distance between point2 and point3 is 10.102
The distance between point3 and point4 is 4.23792
The distance between point4 and point5 is 6.82788
The distance between point5 and point6 is 12.8324
```

課題 3　(提出期限 11 月 7 日 (日) 23:59)

行列の積を計算する関数を作りたい．Course Power にアップロードされている ex6-3.cpp に関数 matrix_product() を追加し，プログラムを完成させなさい．

ソースコード 1: ex6-3.cpp

```cpp
#include <iostream>
#include <vector>
using std::vector, std::cout;

    /*
    2つの正方行列を引数で受け取り，その行列の積を計算し，
    結果の正方行列を返す matrix_product関数を作成せよ
    */

int main()
{
    //2×2の正方行列A, Bに対し, C=A×B
    vector<vector<int>> A{{1,2},{3,4}}, B{{2,2},{3,3}};
    vector<vector<int>> C{matrix_product(A,B)};

    cout << "A*B=\n";  //計算結果を表示
    for (auto& v: C){
        for (auto e: v)
            cout << e << "␣";
        cout << "\n";
    }

    //3×3の正方行列D, Eに対し, F=D×E
    vector<vector<int>> D{{1,2,3},{4,5,6},{7,8,9}},
                        E{{-1,2,1},{-2,3,2},{2,-3,5}};
    vector<vector<int>> F{matrix_product(D,E)};

    cout << "\nD*E=\n";  //計算結果を表示
    for (auto& v: F){
        for (auto e: v)
            cout << e << "␣";
        cout << "\n";
    }
    return 0;
}
```

実行結果

```
$ ./ex6-3
A*B=
8 8
18 18

D*E=
1 -1 20
-2 5 44
-5 11 68
```

注意事項

- main 関数は変更してはいけない．
- 行列は**各成分が整数の正方行列のみ**を想定してよい．ただし，任意の自然数 n に対し，n 次正方行列の積が計算できる関数を作成すること．
- 難しい人は，まずは 2 次正方行列の積を計算する関数 matrix2D_product() と，3 次正方行列の席を計算する matrix3D_product() をそれぞれ作成してみて，どうすれば一般化できるか考えてみること．(この場合，main 関数内の関数名も変えてよい．)