Pointgivende aktivitet

Denne pointgivende aktivitet består af to opgaver, hvor der skal programmeres klasserne Matrix og RotMat2D. Efter løsning af opgave 1 og 2 skal nedenstående main metode kunne køres (main.cpp er vedlagt opgaven). Som svar til opgaverne skal I **aflevere én zip-fil** indeholdende følgende filer:

output.pdf - Outputtet af jeres program og eventuelle fejlmeddelelselser i tilfælde af at jeres program ikke compiler
matrix.h
matrix.cpp
rotmat.h
rotmat.cpp

Opgaven afleveres på itsLearning **senest 26. april kl. 23.59.**

```cpp
#include <iostream>
#include "matrix.h"
#include "rotmat.h"
#include <math.h>
#include <sstream>
#include <string>

void setMatrixValues(Matrix * m, std::vector<double> v){
    for(int r=0; r<m->getRows(); r++){
        for(int c=0; c<m->getCols(); c++){
            m->at(r,c) = v[r*m->getCols() + c];
        }
    }
}

std::string mat2str(Matrix m){
     std::stringstream ss;
    for(int r=0; r<m.getRows(); r++){
        for(int c=0; c<m.getCols(); c++){
            ss << m(r,c);
            if(c<m.getCols()-1){
                ss << " ";
            }
        }
        ss << ";";
        if(r<m.getRows()-1){
            ss<<" ";
        }
    }
    return ss.str();
}

void test(Matrix test, std::string target, std::string msg){
    std::string mStr = mat2str(test);
    std::string equal = mStr==target ? "Success: " : "Failed:  ";
    std::cout << equal << msg << "\t " << mStr << " " << target << std::endl;
}

template <class T1>
void test(T1 test, T1 target, std::string msg){
    std::string equal = test==target ? "Success: " : "Failed:  ";
    std::cout << equal << msg << "\t " << test << " " << target << std::endl;
}
```

```
int main(){
    //Construction and index access
    Matrix mat(3,4);
    test(mat,"0 0 0 0; 0 0 0 0; 0 0 0 0;", "simple constructor");
    Matrix mat2(3,4,3);
    test<double>(mat2.at(0,0),3,"at method");
    test<double>(mat2(2,3),3,"() operator");


    //Transpose
    setMatrixValues(&mat,std::vector<double>{0,1,2,3,4,5,6,7,8,9,10,11});
    Matrix mat3 = mat; //Make a copy of mat
    test(mat3,"0 1 2 3; 4 5 6 7; 8 9 10 11;", "creation of mat3");
    mat3.transpose();
    test(mat3,"0 4 8; 1 5 9; 2 6 10; 3 7 11;", "transpose()");


    //Addition
    Matrix mat4(4,3);
    setMatrixValues(&mat4,std::vector<double>{0,0,1,1,1,2,2,2,3,3,3,4});
    test(mat4,"0 0 1; 1 1 2; 2 2 3; 3 3 4;", "Creation of mat4");
    Matrix mat5 = mat3.add(mat4);
    test(mat5,"0 4 9; 2 6 11; 4 8 13; 6 10 15;", "add()");
    Matrix mat6 = mat4 + mat3;
    test(mat6,"0 4 9; 2 6 11; 4 8 13; 6 10 15;", "+ operator");


    //Inner product
    test(Matrix(1,3,2)*Matrix(3,1,2),"12;","inner product");


    //outer product
    test(Matrix(3,1,2)*Matrix(1,3,2),"4 4 4; 4 4 4; 4 4 4;","outer product");


    //Multiplication
    mat4.transpose();
    Matrix mat7 = mat4.multiply(mat6);
    Matrix mat8 = mat6*mat4;
    test(mat7,"28 52 82; 28 52 82; 40 80 130;","multiply() (matrix multiplication");
    test(mat8,"9 22 35 48; 11 30 49 68; 13 38 63 88; 15 46 77 108;","* operator (matrix multiplication)");
    Matrix mat9 = mat8 * 0.5;
    test(mat9,"4.5 11 17.5 24; 5.5 15 24.5 34; 6.5 19 31.5 44; 7.5 23 38.5 54;","* operator (scalar
multiplication)");


    //Rotation matrices
    RotMat2D identity;
    test(identity,"1 0; 0 1;", "Identity");
    RotMat2D ninety(M_PI/2);
    test(ninety,"6.12323e-17 -1; 1 6.12323e-17;", "ninety");
    ninety.transpose();
    test(ninety,"6.12323e-17 1; -1 6.12323e-17;", "ninety trasposed");
    RotMat2D thirty(30*M_PI/180);
    test(thirty,"0.866025 -0.5; 0.5 0.866025;", "thirty");
    RotMat2D sixty(60*M_PI/180);
    test(sixty,"0.5 -0.866025; 0.866025 0.5;", "sixty");
    Matrix allThree = thirty * sixty;
    test(allThree,"2.22045e-16 -1; 1 2.22045e-16;", "thirty*sixty");
    allThree = allThree * ninety;
    test(allThree,"1 1.60812e-16; -1.60812e-16 1;", "thirty*sixty* ninety (ninety is transposed)");


    //Rotate v by thirty degrees
    Matrix v(2,1,0);
    v(0,0) = 2;
    v(1,0) = 3;
    test(thirty*v,"0.232051; 3.59808;", "Rotation of vector using rotation matrix");


}
```

Opgave 1 – Matrix

Denne opgaven går ud på at skrive en klasse til at repræsentere en matrix. Klassen har operator overload som gør det let at bruge klassen til at regne med matricer. Det er således muligt at skrive A + B*C i stedet for det noget mere bøvlede A.add(B.multiply(C)). Vær opmærksom på at selvom opgaven tager udgangspunkt i matrix-øvelsen fra lektion 6, så er den modificeret. I skal således læse nedenstående nøje, og sikre jer at I kan køre den tilhørende test kode i main.cpp.

Klassens deklaration ser således ud:

```
class Matrix
{
public:
    Matrix(int rows, int cols, double val = 0.0);
    int getRows();
    int getCols();
    virtual void print();
    double& at(int i, int j);
    virtual void transpose();
    Matrix add(Matrix matrix);
    Matrix multiply(Matrix matrix);
    double& operator()(int i, int j);
    Matrix operator+(Matrix m);
    Matrix operator*(Matrix m);
    Matrix operator*(double v);
private:
    std::vector<double> data;
    int rows;
    int cols;
};
```

Include: vector

Opret matrix.h som skal indeholde klassens deklaration. Husk at inkludere nødvendige libraries (her std::vector) og husk include guard. Opret desuden matrix.cpp som skal indeholde klassens definition. Implementer følgende metoder:

1. Constructor til Matrix som angiver antal rækker og colonner i matricen. Alle værdier i matricen skal initialiseres til at have værdien val.
2. Getter metoder til rows og cols
3. En print metode som printer matricen til terminalen vha. std::cout. Klassen skal oprettes som virtual
4. at metode som returnerer en reference til det element i matricen som er på index i,j. Obs. der benyttes nul indexering, dvs. det øverste venstre element i matricen er (0,0)
5. transpose som transponerer matricen in place. Dvs. denne metode returnere ikke noget, men ænder matricen til dens transponerede.
6. add metode som returnerer en matrix der er summen af matricen selv og input matricen matrix

7. multiply metode som returnerer en matrix som er matrix produktet af matricen selv og input matricen matrix-regning
8. operator overload for parentes, plus og gange operatorerne
   a) ()-operatoren skal returnere en reference til det element i matricen som er på index i,j.
   b) +operatoren skal returnere summen af de to matricer der adderes
   c) *operatoren skal have to funktionaliteter alt efter om det venstre argument er en anden matrix eller en skalar. Hvis det er en anden matrix skal der returneres matrix produktet. Hvis det er en skalar, skal skalaren ganges på hvert element i matricen. Obs. i denne opgave skal I ikke implementere at man kan gange en skalar på venstre-siden af en matrix.

Opgave 2 – RotMat2D

Denne opgave går ud på at skrive en klasse til at repræsentere en rotationsmatrix til rotationer i 2D. En rotationmatrix har følgende form:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Vær opmærksom på at denne opgave udelukkende benytter radianer til at beskrive vinkler. Da en rotations matrix er en normal matrix med yderligere egenskaber er det oplagt at benytte nedarvning til at genbruge funktionaliteten fra Opgave 1. Opgaven går altså ud på at skrive klassen RotMat2D som beskrevet nedenfor:

1. Opret rotmat.h, som skal indeholde klassens deklaration, og rotmat.cpp som skal indeholde klassens definition.
2. RotMat2D skal nedarve fra Matrix og har én privat variabel, angle, som angiver rotationen (med fortegn og angivet i radianer) som rotationsmatricen repræsenterer. Alle nedenstående metoder er public.
3. Skriv en default constructor RotMat2D() som kalder super-klassens konstructor så den bliver initialiseret som en 2x2 matrix. Initialiser matricen til identitetsmatricen (dvs. [1 0; 0 1]). Initialiser angle til at være 0 radianer.
4. Skriv constructoren RotMat2D(double angle) som tager en vinkel i radianer som input og initialiserer en rotations matrix ud fra denne vinkel. Se definitionen af en rotationsmatrix ovenfor.
5. Skriv klassen print() som overskriver print metoden i super-klassen. Husk at angive metoden med override key-wordet. Metoden skal kalde super-klassens print metode som printer matricen til terminalen, men derover skal den også printe den vinkel som metoden representerer.
6. Skriv klassen transpose() som overskriver transpose metoden i super-klassen. Husk at angive metoden med override key-wordet. Metoden skal kalde super-klassens transpose metode, men derudover skal den også ændre angle til at have omvendt fortegn (Den trasponerede af en rotationsmatrix er dens inverse, dvs. det er en rotation om samme akse men med omvendt fortegn.)