C++ Програмчлалын Хэлийг Ашиглан Оператор Дахин Тодорхойлох

(Лаборатори №6)

Н. Энхболд

ХШИУС-МКУТ, Програм хангамжийн 2-р түвшний оюутан, 20B1NUM0102

1. ОРШИЛ

C++ хэлний операторыг дахин тодорхойлох, түүнийг хэрэглээ ба хэрэгжүүлэлийг матриц дээр хийж гүйцэтгэнэ..

2. ЗОРИЛГО

C++ хэлний операторуудыг дахин тодорхойлж тэдгээрийг матриц хооронд нэмэх, хасах, үржих, утга оноох, хөрвүүлэх үйлдэлийг хийдэг байхаар тооцоолж хэрэгжүүлнэ.

Үүний тулд дараах зорилтуудыг тавьж ажиллана.

1. C++ хэлэнд операторыг дахин тодорхойлох арга зам, бичиглэлийг судлана.
2. Матриц гэж юу болох, түүн дээр хийгддэг үйлдлүүдийг судлаж матрицыг классаар хэрхэн илэрхийлэх гишүүн өгөгдөл, функцыг зарлана.
3. Матриц дээр хийгдэх үйлдэл болгонд тохирох операторыг дахин тодорхойлж логик үйлдлүүдийг хийнэ.

3. ОНОЛЫН СУДАЛГАА

3.1 Оператор дахин тодорхойлох

C++ хэлний int, float зэрэг дотоод суурь төрлийн өгөгдөл дээр +,-,*,/ оператороор үйлдэл хийхэд компайлер шууд тайлж уншиж холбогдох үйлдэлийг хийж чаддаг. Харин хэрэглэгчийн зохиосон зохиомол өгөгдлийн төрөл дээр үндсэн операторуудыг шууд ашиглаж болдоггүй. Жишээ нь энэ лабораторын ажилд хэрэгжүүлэх гэж байгаа матрицийг үндсэн + операторoор шууд нэмэж болохгүй. Иймд операторын үйлдлийг дахин тодорхойлох хэрэгтэй болдог ба бидний жишээн дээр + операторыг матриц нэмэх үйлдлийг хийдгээр дахин тодорхойлох юм.

Хоёр матрицийг нэмдэг + операторыг дахин тодорхойлох:

```
Matrix Matrix::operator +(float &a){
    int row = this->getRow(),
            col = this->getCol();
    Matrix temp(row,col);
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
           float valuesSum= this->getValues(&i,&j)+a;
           temp.setValues(&i,&j,&valuesSum);
        }
     }
return temp;
}
```

6. АШИГЛАСАН МАТЕРИАЛ

1. C++ хэл дээр оператор дахин тодорхойлох
https://www.tutorialspoint.com/cplusplus/cpp_overloading.html

2. Объект хандлагат програмчлал – Лекц №07 – Operator Overloading

https://drive.google.com/file/d/134Clg0yTImibNqDBARCCCG_T_3KUCrku/view?usp=sharing

7. ХАВСРАЛТ

```
#include <iostream>
using namespace std;


class Matrix{
    public:
        int row;
        int col;
        float *values;
    public:
         Matrix(int row,int col);
```

```
        Matrix* operator +(Matrix &a);
        Matrix* operator +(float &a);
        Matrix* operator *(Matrix &a);
        Matrix* operator -(Matrix &a);
        void operator =(Matrix &a);
        ~Matrix();
        void display();
        int getCol();
        int getRow();
        float* getAllValuess();
        float getValues(int *row, int *col);

        void setAll(int *row, int *col, float *values);
        void setRow(int *row);
        void setCol(int *col);
        void setValues(int *row, int *col, float *values);
};


Matrix::Matrix(int row, int col){
    setAll(&row,&col,NULL);
}

Matrix::~Matrix(){
    delete [] values;
    cout<<"matrix ustaw"<<endl;
}
```

```
Matrix Matrix::operator +(Matrix  &a){
    int row = this->getRow(),
        col = this->getCol();
    Matrix temp(row,col);
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
        float valuesSum=
this->getValues(&i,&j)+a.getValues(&i,&j);
        temp.setValues(&i,&j,&valuesSum);
        }
    }
    return temp;
}

Matrix Matrix::operator +(float &a){
    int row = this->getRow(),
        col = this->getCol();
    Matrix temp(row,col);
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
           float valuesSum= this->getValues(&i,&j)+a;
           temp.setValues(&i,&j,&valuesSum);
        }
    }
    return temp;
}

Matrix Matrix::operator *(Matrix &a){
```

```
    int row = this->getRow(),
        col = this->getCol();
    Matrix temp(row,col);
    int i, j, k;
     for(i = 0; i < row; ++i)
     {
          for(j = 0; j < col; ++j)
          {
            float aa=0;
               for(k=0; k<col; ++k)
               {
                aa+=this->getValues(&i,&k) * a.getValues(&k,&j);

               }
            temp.setValues(&i,&j,&aa);
          }
     }

    return temp;
}

Matrix Matrix::operator -(Matrix &a){
    int row = this->getRow(),
        col = this->getCol();
    Matrix temp(row,col);
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
           float valuesSum= this->getValues(&i,&j)-
```

```
a.getValues(&i,&j);
            temp.setValues(&i,&j,&valuesSum);
        }
    }
    return temp;
}

Matrix& Matrix::operator =(Matrix a){
    int row=a.getRow(),
        col=a.getCol();
    this->setRow(&row);
    this->setCol(&col);
    delete [] values;

    this->values = new float(col*row);
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
            float values =a.getValues(&i,&j);
            this->setValues(&i,&j,&values);
        }
    }

    return (*this);
}


void Matrix::operator ++(int){
    for(int i=0;i<this->getRow();i++){
```

```
        for(int j=0;j<this->getCol();j++){
           float valuesSum= this->getValues(&i,&j)+1.0;
           setValues(&i,&j,&valuesSum);
        }
    }

}


void Matrix::operator --(int){
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
           float valuesSab= this->getValues(&i,&j)-1.0;
           setValues(&i,&j,&valuesSab);
        }
    }

}


void Matrix::operator +=(Matrix &a){
    int row = this->getRow(),
    col = this->getCol();
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
        float valuesSum=
this->getValues(&i,&j)+a.getValues(&i,&j);
        this->setValues(&i,&j,&valuesSum);
```

```
        }
    }
}


void Matrix::operator -=(Matrix &a){
    int row = this->getRow(),
    col = this->getCol();
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
        float valuesSum= this->getValues(&i,&j)-
a.getValues(&i,&j);
        this->setValues(&i,&j,&valuesSum);
        }
    }
}


void Matrix::operator *=(Matrix &a){
    int row = this->getRow(),
    col = this->getCol();
    int i, j, k;
    float bb[row][col];
     for(i = 0; i < row; ++i)
     {
          for(j = 0; j < col; ++j)
          {
               for(k=0; k<col; k++)
```

```
                {
                 bb[i][j]+=this->getValues(&i,&k) *
a.getValues(&k,&j);
                }
           }
      }

    for(i = 0; i < row; ++i)
     {
          for(j = 0; j < col; ++j)
          {
            float *b = &bb[i][j];
            this->setValues(&i,&j,b);
        }
    }
}


 bool Matrix::operator !(void){
    int row = getRow();
    int col = getCol();
        for(int i=0; i<row; i++){
            for(int j=0; j<col; j++){
                cout <<this -> getValues(&j,&i)<<"  ";
            }
            cout << endl;
        }
}
```

```
float* Matrix::getValues2(int *row, int *col){
    return values+(*row*this->col+*col);
}


float* Matrix:: operator [](int a){
    float *bb = new float(col);
    for(int i=0;i<col;i++){
        bb[i]=*getValues2(&a,&i);
    }
    return bb;
}


float* Matrix::getAllValuess(){
    return this->values;
}


void Matrix::display(){
    for(int i=0;i<this->getRow();i++){
        for(int j=0;j<this->getCol();j++){
            cout<<this->values[(i*this->col)+j]<<" ";
        }
        cout<<endl;
    }
        cout<<"--------"<<endl;
```

```
}


int Matrix::getRow(){
    return row;
}


int Matrix::getCol(){
    return col;
}


float Matrix::getValues(int *row, int *col){
    return values[*row*this->col+*col];
}


void Matrix::setAll(int *row, int *col, float *values){
    setRow(row);
    setCol(col);
    setValues(row,col,values);
}


void Matrix::setRow(int *row){
    this->row = *row;
}
```

```
void Matrix::setCol(int *col){
    this->col = *col;
}

void Matrix::setValues(int *row, int *col, float *values){
    if(values==NULL){
        this->values = new float[*row*(*col)];
    }else{
        int pos=((*row)*this->col)+*col;
        this->values[pos]=*values;
    }
}
```