Лабораторийн ажил №9

Н. Мөнхжин

ХШУИС, МКУТ, Компьютерийн ухаан, nyamkamunhjin@gmail.com

1. ОРШИЛ.

С++ хэл дээр класс ашиглан математик матриц хүснэстийг тодорхойлж түүн дээр хийгдэх нэмэх, хасах, үржих үйлдлүүдийг оператор дахин тодорхойлж хийж гүйцэтгэнэ.

2. ЗОРИЛГО

Матриц объектийг тодорхойлох нь:

- Матриц классыг тодорхойлж түүн дээх хийгдэх гишүүн өгөгдөл, функцуудыг тодорхойлох.
- Матриц дээр хийгддэг нэмэх, хасах, үйлдлийг операторыг дахин тодорхойлж гүйцэтгэх. Дахин тодорхойлох операторын гүйцэтгэх үүрэг:

1. (+) : буцаах утга: матриц + тоо, матриц + матриц, матриц .
2. (-) : буцаах утга: матриц - матриц.
3. (*) : буцаах утга: матриц * матриц.
4. (++) : матрицийн утгуудыг нэгээр нэмэгдүүлэх.
5. (--) : матрицийн утгуудыг нэгээр хорогдуулах.
6. (+=) : матриц дээр матрицийн утгуудыг нэмэж хадгалах.
7. (-=) : матрицаас матрицийн утгуудыг хасаж хадгалах.
8. (*=) : матрицийг матрицаар үржүүлж хадгалах.

3. ОНОЛЫН СУДАЛГАА

3.1 Оператор дахин тодорхойлох.

Хэрэглэгчийн тодорхойлсон зохиомол өгөгдлийн төрлүүд дээр энгийн операторуудыг хэрэглэж болдоггүй. Зөвхөн тухайн зохиомол өгөгдлийн төрөл дээр ашиглагдах боломжтой операторуудыг дахин тодорхойлогдсон оператор гэнэ. Дахин тодорхойлогдсон операторуудыг ашиглснаар кодыг энгийн, ойлгомжтой, богино болгох боломжтой.[1]

4. ХЭРЭГЖҮҮЛЭЛТ

Матриц классын гишүүн функцууд. Шинээр үүсгэж буй матриц хүснэгтэд утга оноохдоо давхар хаяг ашиглан шийдсэнээр динамик буюу хүснэгтийн хэмжээг өөрчлөх боломжтой болж байна.

```
class Matrix {

    private:
       int m, n; // 2-D matrix dimensions
       float **values;
```

Байгуулагч-руу дамжуулсан хүснэгтийн хэмжээнээс хамааран Матриц-д тохирох санах ойн хэмжээг нөөцлөж байна.

```
    // Constructor
Matrix(int n=1, int m=1) {
    this->n = n;
    this->m = m;
    // allocate memory for matrix
    this->values = new float*[n];
    for(int i = 0; i < n; i++) {
        this->values[i] = new float[m];
    }
}
```

Устгагч байгуулагч дуудагдах үед нөөцөлсөн санах ойг чөлөөлөн ойн цоорхой үүсхээс сэргийлнэ.

```
    // Destructor
      ~Matrix() {
          // freeing memory
          for(int i = 0; i < this->n; i++)
              delete[] this->values[i];
          delete[] this->values;
      }
```

Матриц хүснэгт дээр хэрэглэгдэх нэмэх, хасах, үржих үйлдлүүд.

```
Matrix add(const Matrix &matrix) {
        Matrix temp = *this;
        for(int i = 0; i < temp.n; i++)
            for(int j = 0; j < temp.m; j++)
                temp.values[i][j] += matrix.values[i][j];
        return temp;
}

    Matrix matmul(const Matrix &matrix) {
        if(this->m != matrix.n) {
            printf("Matrices doesn't match (%d, %d) != (%d, %d).",
             this->n, this->m, matrix.n, matrix.m);
            printf("Returned first matrix.\n");
            return (*this);
        }

        Matrix temp(this->n, matrix.m);

        for(int i = 0; i < temp.n; i++) {
            for(int j = 0; j < temp.m; j++) {
                float sum = 0;
                for(int k = 0; k < this->m; k++) {
                    sum += this->values[i][k]
* matrix.values[k][j];
                }
                temp.values[i][j] = sum;
            }
        }
        return temp;
    }
```

Операторыг дахин тодорхойлов.

```
Matrix operator + (const Matrix &matrix) {
        return this->add(matrix);
    }

    // * operator
    Matrix operator * (const Matrix &matrix) {
        return this->matmul(matrix);
    }

    // - operator
    Matrix operator - (const Matrix &matrix) {
        Matrix temp = *this;
        for(int i = 0; i < temp.n; i++)
            for(int j = 0; j < temp.m; j++)
                temp.values[i][j] -= matrix.values[i][j];

        return temp;
    }
```

5. ДҮГНЭЛТ

Дээрх даалгаврын хүрээнд обьект хандлагат програмчлалын класс ашиглан матриц хүснэгтийг түүний үйлдлүүдийг тодорхойлов. Операторуудыг матриц хүснэгтэд тохируулж дахин тодорхойлсноор код бичиж байгаа хүний хувьд илүү цэгцтэй, энгийн, ойлгомжтой болгов.

6. АШИГЛАСАН МАТЕРИАЛ

1. https://www.tutorialspoint.com/cplusplus/cpp_overloading.htm

7. ХАВСРАЛТ

```
#include <iostream>
```

```
using namespace std;

class Matrix {
private:
   int m, n; // 2-D matrix dimensions
   float **values;

public:
   // Constructor
   Matrix(int n=1, int m=1) {
       this->n = n;
       this->m = m;
       // allocate memory for matrix
       this->values = new float*[n];
       for(int i = 0; i < n; i++) {
           this->values[i] = new float[m];
       }

       // put zeros in matrix
       for(int i = 0; i < n; i++)
           for(int j = 0; j < m; j++)
               this->values[i][j] = 0;

   }
   // Copy constructor
   Matrix(const Matrix &copy) {
       // setting dimenstions
       this->n = copy.n;
       this->m = copy.m;

       // allocate memory for matrix
       this->values = new float*[n];
       for(int i = 0; i < n; i++) {
```

```
        this->values[i] = new float[m];
    }

    // setting values
    for(int i = 0; i < n; i++)
        for(int j = 0; j < m; j++)
            this->values[i][j] = copy.values[i][j];
}

// Destructor
~Matrix() {
    // freeing memory
    for(int i = 0; i < this->n; i++)
        delete[] this->values[i];
    delete[] this->values;
}

// printing matrix values
void print() {
    printf("[ ");
    for(int i = 0; i < this->n; i++) {
        if(i > 0) printf("  ");
        for(int j = 0; j < this->m; j++) {
            printf("%.3f, ", this->values[i][j]);

            if(i == this->n - 1 && j == this->m - 1) printf("]");
        }
        printf("\n");
    }
    printf("\n");
}

void setValue(int i, int j, float value) {
```

```
    if(i >= n || j >= m) {
        printf("invalid index\n");
        return;
    }
    this->values[i][j] = value;
}
void setValues(float **values) {
    // freeing memory
    for(int i = 0; i < this->n; i++)
        delete this->values[i];
    delete this->values;

    // allocate memory for matrix
    this->values = new float*[n];
    for(int i = 0; i < n; i++) {
        this->values[i] = new float[m];
    }

    // setting values
    for(int i = 0; i < n; i++)
        for(int j = 0; j < m; j++)
            this->values[i][j] = values[i][j];
}

float getValue(int i, int j) {
    if(i >= n || j >= m) {
        printf("invalid index\n");
        return -9999999999;
    }
    return this->values[i][j];
}

float** getValues() {
```

```
    return this->values;
}

void shape() {
    printf("%d %d", this->n, this->m);
}

// ones
void ones() {
    for(int i = 0; i < n; i++)
        for(int j = 0; j < m; j++)
            this->values[i][j] = 1;
}

Matrix add(float num) {
   Matrix temp = *this;
    for(int i = 0; i < temp.n; i++)
        for(int j = 0; j < temp.m; j++)
            temp.values[i][j] += num;
    return temp;

}

Matrix add(const Matrix &matrix) {
    Matrix temp = *this;
    for(int i = 0; i < temp.n; i++)
        for(int j = 0; j < temp.m; j++)
            temp.values[i][j] += matrix.values[i][j];
    return temp;


}
```

```
Matrix matmul(const Matrix &matrix) {
    if(this->m != matrix.n) {
        printf("Matrices doesn't match (%d, %d) != (%d, %d). ",
         this->n, this->m, matrix.n, matrix.m);
        printf("Returned first matrix.\n");
        return (*this);
    }

    Matrix temp(this->n, matrix.m);

    for(int i = 0; i < temp.n; i++) {
        for(int j = 0; j < temp.m; j++) {
            float sum = 0;
            for(int k = 0; k < this->m; k++) {
                sum += this->values[i][k] * matrix.values[k][j];
            }
            temp.values[i][j] = sum;
        }
    }
    return temp;
}

// Operator overloading

// + operator
Matrix operator + (float num) {
    return this->add(num);
}

Matrix operator + (const Matrix &matrix) {
    return this->add(matrix);
}
```

```
// * operator
Matrix operator * (const Matrix &matrix) {
    return this->matmul(matrix);
}

// - operator
Matrix operator - (const Matrix &matrix) {
    Matrix temp = *this;
    for(int i = 0; i < temp.n; i++)
        for(int j = 0; j < temp.m; j++)
            temp.values[i][j] -= matrix.values[i][j];

    return temp;
}

// ++ operator
void operator ++ (int) {
    for(int i = 0; i < this->n; i++)
        for(int j = 0; j < this->m; j++)
            this->values[i][j] += 1;
}

// -- operator
void operator -- (int) {
    for(int i = 0; i < this->n; i++)
        for(int j = 0; j < this->m; j++)
            this->values[i][j] -= 1;
}

// += operator
void operator += (const Matrix &matrix) {
    for(int i = 0; i < this->n; i++)
        for(int j = 0; j < this->m; j++)
```

```
            this->values[i][j] += matrix.values[i][j];
}

// -= operator
void operator -= (const Matrix &matrix) {
    for(int i = 0; i < this->n; i++)
        for(int j = 0; j < this->m; j++)
            this->values[i][j] -= matrix.values[i][j];
}

// *= operator
void operator *= (const Matrix &matrix) {
    if(this->m != matrix.n) {
        printf("Matrices doesn't match (%d, %d) != (%d, %d). ",
         this->n, this->m, matrix.n, matrix.m);
        printf("Returned first matrix.\n");
        return;
    }

    Matrix temp(this->n, matrix.m);

    for(int i = 0; i < temp.n; i++) {
        for(int j = 0; j < temp.m; j++) {
            float sum = 0;
            for(int k = 0; k < this->m; k++) {
                sum += this->values[i][k] * matrix.values[k][j];
            }
            temp.values[i][j] = sum;
        }
    }

    for(int i = 0; i < temp.n; i++) {
        for(int j = 0; j < temp.m; j++) {
```

```
                this->values[i][j] = temp.values[i][j];
            }
        }
    }

    // transposing matrix
    Matrix t() {
        Matrix temp(this->m, this->n);

        for(int i = 0; i < this->n; i++)
            for(int j = 0; j < this->m; j++)
                temp.values[j][i] = this->values[i][j];

        return temp;
    }


};

int main() {
    // testing
    printf("(matrix 1)\n");
    Matrix test(2, 3);
    test.setValue(0, 0, 1);
    test.setValue(0, 1, 2);
    test.setValue(0, 2, 3);
    test.setValue(1, 0, 4);
    test.setValue(1, 1, 5);
    test.setValue(1, 2, 6);

    test.print();
    printf("(matrix 2)\n");
    Matrix test1(3, 2);
```

```
test1.setValue(0, 0, 1);
test1.setValue(0, 1, 2);
test1.setValue(1, 0, 3);
test1.setValue(1, 1, 4);
test1.setValue(2, 0, 5);
test1.setValue(2, 1, 6);
test1.print();

printf("(matrix 1 + (10.0))\n");
(test + 10).print();

printf("(matrix 1 + matrix 1)\n");
(test + test).print();

printf("(matrix 1 - matrix 1)\n");
(test - test).print();

printf("(matrix 1 * matrix 2)\n");
(test * test1).print();

printf("(matrix 1 += matrix 1)\n");
test += test;
test.print();

printf("(matrix 1 -= matrix 1)\n");
test -= test;
test.print();

printf("(matrix 1 *= matrix 2)\n");
test *= test1;
test.print();

printf("(matrix 2 ++ )\n");
```

```
    test1++;
    test1.print();

    printf("(matrix 2 -- )\n");
    test1--;
    test1.print();

    return 0;
}
```