Оператор дахин тодорхойлох
(Лаборатори №9)

О. Ихбаяр

ХШУИС, Програм хангамж, 3-р түвшин, 17b1num2575@stud.num.edu.mn

1. ОРШИЛ

Оператор дахин тодорхойлох гэж юу болох, түүнийг хэрхэн зөв оновчтой хэрэглэх талаар авч үзэх болно.

2. ЗОРИЛГО

Оператор дахин тодорхойлох аргыг эзэмших ба хэрхэн хэрэглэх талаарх тодорхой хэмжээний ойлголтыг өөрийн болгож авах зорилготой.

3. ОНОЛЫН СУДАЛГАА

3.1 Оператор дахин тодорхойлох

C++ хэлний int, char гэх зэрэг дотоод суурь өгөгдөл дээр +, -, *, +=, … гэх мэтийн операторыг хэрэглэн олон зүйлийн бодолт хийж болно. Харин хэрэглэгчийн тодорхойлсон зохиотол төрлийн хувьд C++ хэлний үндсэн операторуудыг дээрхийн адилаар хэрэглэж болдоггүй. Тиймээс операторыг дахин тодорхойлох C++ хэлний арга технологийг хэрэглэн (+) нэмэх, хасах гэх мэт операторуудын үүргийг дахин тодорхойлж өгснөөр доор үзүүлсэнтэй адил үйлдлийг зохиомол төрлийн өгөгдлийн хувьд хийх боломжтой болно.

3.2 Нэг операндын оператор дахин тодорхойлох

C++ хэлэнд байдаг ганц операндын ++, -- хоёр оператор нь операндынхаа утгыг харгалзан нэгээр нэмэгдүүлж нэгээр хорогдуулах үүрэгтэй. Тэгвэл зохиомол төрлийн объектын хувьд c1++ үйлдэл хйидэг байхын тулд ++ операторыг дахин тодорхойлж өгөх шаардлагатай.

Оператор дахин тодорхойлохдоо:

Return-type()/буцаах утгын төрөл/ + operator/Тусгай түлхүүр үг/ + op/дахин тодорхойлох оператор/+ (argument_list)/авах аргументын жагсаалт/

Нэг операндын оператор дахин тодорхойлох гишүүн функцийг дээрх загварын дагуу void operator ++ (void); гэж зарлана.

3.3 Хос операндын оператор дахин тодорхойлох

A O1, O2, O3; Бүгд нэгэн төрлийн зохиомол O3 = O1 + O2; операндууд байг. Энэ нь O3 = O1.operator(O2) бичиглэлтэй адилхан. Энд зүүн талд байгаа объект O1 нь operator функцийг

дуудаж хэрэглэнэ. Харин баруун талын O2 нь operator Функцийн аргумент болно. Иймд Функцийг тодорхойлохдоо:

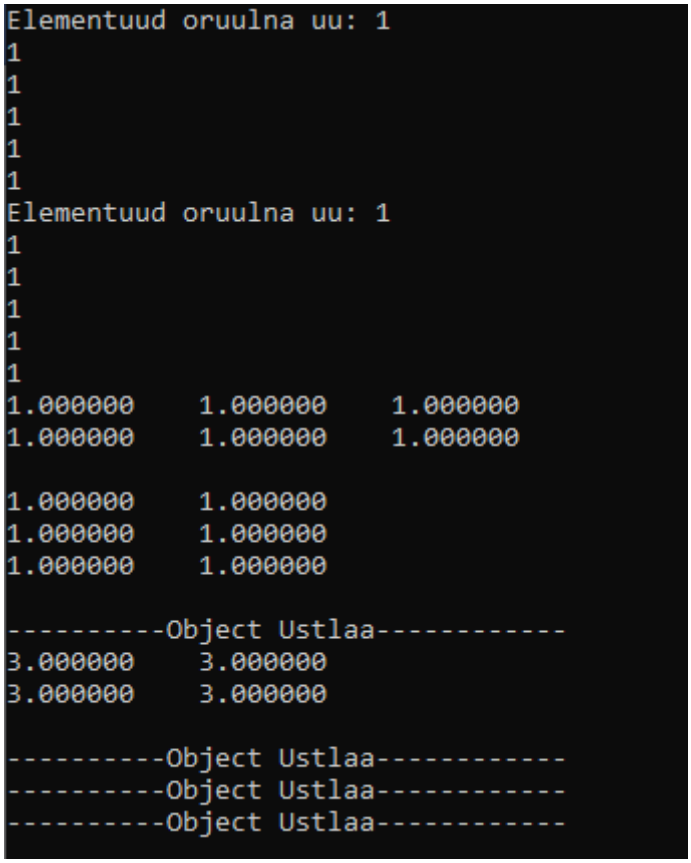
```
        A operator + (A O) бөгөөд өөрөөр
A::operator +(A O){
        Int a = basicpay + O.basicpay;
        Return A(“Total”, a);
} гэж бичиж болно.
```

4. ХЭРЭГЖҮҮЛЭЛТ

4.1 Хуулагч функц тодорхойлох

```
Matrix Matrix::operator*(Matrix x)
{
   Matrix tmp(this->m, x.n);
   tmp.m=this->m;
   tmp.n=x.n;
   for(int i=0; i<this->getRow(); i++)
   {
      for(int j=0; j<this->getCol(); j++)
      {
         for(int k=0; k<this->getCol(); k++)
         {
            tmp.values[i][j]+=this->getVal(i,k)*x.getVal(k,j);
         }
      }
   }
   return tmp;
}
```

Тодорхойлолт: Дараах код нь үржих(*) оператор дахин тодорхойлж байгаа бөгөөд энэхүү бичилт хоёр матрицыг хооронд нь үржиж байна. Ингэхдээ аргумэнтээр дамжигдан ирж буй матриц нь үржих матриц болно. Доор ажилласан жишээг харуулав.



5. ДҮГНЭЛТ

Энэхүү лабораторын ажиллаар операторыг хэрхэн дахин тодорхойлох талаар ойлголттой болж авсан.

6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандлагат технологийн С++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.

7. ХАВСРАЛТ

```
#include <iostream>

using namespace std;

/*

   Matrix class

   int n - matrixiin muriin toog hadgalna

   int m - matrixiin baganii toog hadgalna

   values elementiin haygiig hadgalna

*/
```

```
class Matrix
{
   private:
      int m,n;
      float **values;
   public:
      Matrix(int m=1, int n=1)
      {
         this->m=m;
         this->n=n;
         values=new float*[m];
         for(int i=0; i<this->m; i++)
         {
            values[i]=new float[n];
         }
         for(int i=0; i<this->m; i++)
         {
            for(int j=0; j<this->n;j++)
            {
               values[i][j]=0;
            }
         }
      }
      ~Matrix()
      {
         for(int i=0; i<m; i++){
            delete[] values[i];
         }
```

```
            delete[] values;
        cout<<"----------Object Ustlaa----------"<<endl;
    }
    //matrixiin moriin utgiig avch butsaana
    int getRow();
    //matrixiin baganiin utgiig avch butsaana
    int getCol();
    float getVal(int m, int n);
    void print();
    void setVal();
    Matrix operator+(float x);
    Matrix operator+(Matrix x);
    Matrix operator-(Matrix x);
    Matrix operator=(Matrix x);
    Matrix operator++();
    Matrix operator--();
    Matrix operator+=(Matrix x);
    Matrix operator-=(Matrix x);
    Matrix operator*(Matrix x);
    Matrix operator*=(Matrix x);
    Matrix operator&();
//      float* operator[](int x);
};
float Matrix::getVal(int m, int n)
{
   return this->values[m][n];
}
int Matrix::getRow()
```

```
{
   return this->m;
}
int Matrix::getCol()
{
   return this->n;
}
void Matrix:: print()
{
   for(int i=0; i<this->getRow(); i++)
   {
      for(int j=0; j<this->getCol(); j++)
      {
        cout<< fixed << this->getVal(i,j) << "    ";
      }
      cout<<endl;
   }
   cout << endl;
}
void Matrix::setVal()
{
   cout << "Elementuud oruulna uu: ";
   for(int i=0; i<this->m; i++)
   {
      for(int j=0; j<this->n; j++){
         cin>>this->values[i][j];
      }
   }
```

```
}

/*
   + operatoriig dahin todorhoilson
*/
Matrix Matrix::operator+(float x)
{
   Matrix tmp(this->m, this->n);
   for(int i=0; i<tmp.getRow(); i++)
   {
      for(int j=0; j<tmp.getCol(); j++)
      {
        tmp.values[i][j]=this->values[i][j]+x;
      }
   }
   tmp.print();
   return tmp;
}

/*
   + operatoriig dahin todorhoilson
*/
Matrix Matrix::operator+(Matrix x)
{
   Matrix tmp(this->m, this->n);
   tmp.m=this->m;
   tmp.n=this->n;
   for(int i=0; i<tmp.getRow(); i++)
```

```
    {
        for(int j=0; j<tmp.getCol(); j++)
        {
            tmp.values[i][j]=this->getVal(i,j)+x.getVal(i,j);
        }
    }
    return tmp;
}

/*
    - operatoriig dahin todorhoilson
*/
Matrix Matrix::operator-(Matrix x)
{
    Matrix tmp(this->m, this->n);
    tmp.m=this->m;
    tmp.n=this->n;
    for(int i=0; i<tmp.getRow(); i++)
    {
        for(int j=0; j<tmp.getCol(); j++)
        {
            tmp.values[i][j]=this->getVal(i,j)-x.getVal(i,j);
        }
    }
    return tmp;
}

/*
```

```
   = operatoriig dahin todorhoilson
*/
Matrix Matrix::operator=(Matrix x)
{
        //sanah oi chuluulnu
        for(int i=0; i<this->m; i++)
   {
      delete[] values[i];
   }
   delete[] values;

   this->m=x.getRow();
        this->n=x.getCol();
        //shineer sanah oi nuutsulnu
        this->values=new float*[this->m];
   for(int i=0; i<this->m; i++)
   {
      values[i]=new float[this->n];
   }

        for(int i=0; i<x.getRow(); i++)
   {
                for(int j=0; j<x.getCol(); j++)
                {
                        this->values[i][j]=x.getVal(i,j);
                }
        }
        return *this;
```

```
}

/*
   ++ operatoriig dahin todorhoilson
*/
Matrix Matrix::operator++()
{
        for(int i=0; i<this->getRow(); i++)
   {
        for(int j=0; j<this->getCol(); j++)
        {
        this->values[i][j]=this->getVal(i,j)+1;;
        }
        }
        return *this;
}

/*
   -- operatoriig dahin todorhoilson
*/
Matrix Matrix::operator--()
{
        for(int i=0; i<this->getRow(); i++)
   {
        for(int j=0; j<this->getCol(); j++)
        {
        this->values[i][j]=this->getVal(i,j)-1;;
        }
```

```
        }
        return *this;
}

/*
   += operatoriig dahin todorhoilson
*/
Matrix Matrix::operator+=(Matrix x)
{
        for(int i=0; i<this->getRow(); i++)
   {
        for(int j=0; j<this->getCol(); j++)
        {
        this->values[i][j]+=x.getVal(i,j);
        }
        }
        return *this;
}

/*
   -= operatoriig dahin todorhoilson
*/
Matrix Matrix::operator-=(Matrix x)
{
        for(int i=0; i<this->getRow(); i++)
   {
        for(int j=0; j<this->getCol(); j++)
        {
```

```
            this->values[i][j]-=x.getVal(i,j);
            }
            }
            return *this;
}

/*
   * operatoriig dahin todorhoilson
*/
Matrix Matrix::operator*(Matrix x)
{
   Matrix tmp(this->m, x.n);
   tmp.m=this->m;
   tmp.n=x.n;
   for(int i=0; i<this->getRow(); i++)
   {
      for(int j=0; j<this->getCol(); j++)
      {
         for(int k=0; k<this->getCol(); k++)
         {
            tmp.values[i][j]+=this->getVal(i,k)*x.getVal(k,j);
         }
      }
   }
   return tmp;
}

/*
```

```
   *= operatoriig dahin todorhoilson
*/
Matrix Matrix::operator*=(Matrix x)
{
   Matrix tmp(this->m, x.n);
   tmp.m=this->m;
   tmp.n=x.n;
   for(int i=0; i<tmp.getRow(); i++)
   {
      for(int j=0; j<tmp.getCol(); j++)
      {
         for(int k=0; k<x.getCol(); k++)
         {
            tmp.values[i][j]+=this->getVal(i,k)*x.getVal(k,j);
         }
      }
   }
   *this=tmp;
   return *this;
}
//Matrix hurvuuleh function.
Matrix Matrix::operator&()
{
   Matrix tmp(this->n, this->m);
   for(int i=0; i<tmp.getRow(); i++) //matrixiin muru baganiin indexiig solij utga
onoono
   {
      for(int j=0; j<tmp.getCol(); j++)
      {
```

```
            tmp.values[i][j]=this->getVal(j,i);
        }
    }
    this->m=tmp.m;
    this->n=tmp.n;
    //tmp.print();
    return tmp;
}

main(){
    int r,c;
    cout<<"Matrixiiin hemjeeg oruulna uu: ";
    cin>>r>>c;
    Matrix a1(r,c), a2(3, 2);
    a1.setVal();
    a2.setVal();
    a1.print();
    a2.print();
    Matrix a3 = a1 * a2;
    a3.print();
}
```