ch_arrays

```
#include <iostream>

using namespace std;

int main()
{
    int* a = 0;
    int* b = 0;

    a = new int[10];

    for (int i=0; i<10; i++)
        a[i] = i;

    b = &a[6];

    cout << "a = " << a << endl;
    cout << "a[5] = " << a[5] << endl;
    cout << "b = " << b << endl;
    cout << "*b = " << *b << endl;

    delete[] a;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int* a = 0;

    a = new int;

    *a = 42;

    cout << "a = " << a << endl;
    cout << "*a = " << *a << endl;

    delete a;
}
```

```
#include <iostream>

using namespace std;

const int rows = 4;
const int cols = 8;

int main()
{
    int** array = new int*[rows];

    int counter = 0;

    for (int i=0; i<rows; i++)
    {
        array[i] = new int[cols];

        for (int j=0; j<cols; j++)
            array[i][j] = counter++;
    }

    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<cols; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    delete [] array;
}
```

```
#include <iostream>

using namespace std;

const int rows = 4;
const int cols = 8;

int main()
{
    int** array = new int*[rows];
    array[0] = new int[rows*cols];
```

```
    int counter = 0;

    for (int i=0; i<rows; i++)
    {
        array[i] = &array[0][i*cols];

        for (int j=0; j<cols; j++)
            array[i][j] = counter++;
    }

    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<cols; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    delete array[0];
    delete array;
}

#include <iostream>

using namespace std;

void createArray(int** &array, int rows, int cols)
{
    array = new int*[rows];
    array[0] = new int[rows*cols];

    for (int i=0; i<rows; i++)
        array[i] = &array[0][i*cols];
}

void zeroArray(int** &array, int rows, int cols)
{
    for (int i=0; i<rows; i++)
        for (int j=0; j<cols; j++)
            array[i][j] = 0;
}

void deleteArray(int** &array)
{
```

```
    delete array[0];
    delete array;
}

int main()
{
    int** array;

    createArray(array, 4, 8);
    zeroArray(array, 4, 8);

    for (int i=0; i<4; i++)
    {
        for (int j=0; j<8; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    deleteArray(array);
}

#include <iostream>

using namespace std;

struct coord3D {
    double x;
    double y;
};

int main()
{
    coord3D* coords = new coord3D[10];

    double counter = 0.0;

    for (int i=0; i<10; i++)
    {
        coords[i].x = counter++;
        coords[i].y = counter++;
    }

    for (int i=0; i<10; i++)
```

```
        cout << coords[i].x << ", " << coords[i].y << endl;

    delete [] coords;
}
```

```
#include <iostream>

using namespace std;

struct coord3D {
    double x;
    double y;
};

int main()
{
    coord3D** coords = new coord3D*[10];

    double counter = 0.0;

    for (int i=0; i<10; i++)
    {
        coords[i] = new coord3D;
        coords[i]->x = counter++;
        coords[i]->y = counter++;
    }

    for (int i=0; i<10; i++)
        cout << coords[i]->x << ", " << coords[i]->y << endl;

    for (int i=0; i<10; i++)
        delete coords[i];

    delete [] coords;
}
```

ch_input_output

```
#include <iostream>
#include <fstream>
#include <iomanip>
```

```
using namespace std;

int main()
{
    ofstream myfile;
    myfile.open("myfile.txt");
    myfile << "Hello file!" << endl;
    myfile << "Second line" << endl;
    myfile.close();
}

#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

int main()
{
    ofstream outfile;
    outfile.open("myfile.txt", ios::out);
    outfile << "Hello file!" << endl;
    outfile << "Second line" << endl;
    outfile.close();

    outfile.open("myfile.txt", ios::out | ios::app);
    outfile << "Third line" << endl;
    outfile.close();

    ifstream infile;
    infile.open("myfile2.txt");
    if (!infile.is_open())
        cout << "Could not open file!" << endl;
    else
        infile.close();
}

#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int main()
```

```
{
    double pi = 4 * std::atan(1);
    double x = 0.0;
    double y;
    double dx = 0.1;

    ofstream outfile;
    outfile.open("inputfile.dat", ios::out);
    while (x<=pi*0.25)
    {
        outfile << x << " " << sin(x) << endl;
        x += dx;
    }
    outfile.close();

    ifstream infile;
    infile.open("inputfile.dat");
    while (infile.good())
    {
        infile >> x >> y;
        cout << "x = " << x << ", y = " << y << endl;
    }
    infile.close();
}

#include <iostream>
#include <fstream>
#include <sstream>

using namespace std;

int main()
{
    string line;
    ifstream infile;
    infile.open("/etc/hosts");
    while (infile.good())
    {
        getline(infile, line);
        string addr;
        string host;

        istringstream linestream(line);
        linestream >> addr >> host;
```

```
        cout << "address = " << addr << ", host = " << host << endl;
    }
    infile.close();
}

#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <ctime>

using namespace std;

struct particle {
    double x;
    double y;
    double mass;
};

int main()
{
    srand((unsigned)time(0));

    ofstream particlesFile("particles.dat", ios::out | ios::binary);

    particle p;

    for (int i=0; i<10; i++)
    {
        p.x = 100.0*(double)rand()/(double)RAND_MAX;
        p.y = 100.0*(double)rand()/(double)RAND_MAX;
        p.mass = 1.0 + (double)rand()/(double)RAND_MAX;

        particlesFile.write((char*)&p, sizeof(p));
    }

    particlesFile.close();

    ifstream inputParticlesFile("particles.dat", ios::in | ios::binary | i

    if (inputParticlesFile.is_open())
    {
        inputParticlesFile.seekg(0, ios::beg);
```

```
        while (inputParticlesFile.good())
        {
            inputParticlesFile.read((char*)&p, sizeof(p));
            cout << "x = " << p.x;
            cout << ", y = " << p.y;
            cout << ", m = " << p.mass << endl;
        }
    }
    else
        cout << "Could not open file." << endl;

    inputParticlesFile.close();
}

#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <ctime>

using namespace std;

struct particle {
    double x;
    double y;
    double mass;
};

int main()
{
    srand((unsigned)time(0));

    ofstream particlesFile("particles.dat", ios::out | ios::binary);

    particle p;

    for (int i=0; i<10; i++)
    {
        p.x = 100.0*(double)rand()/(double)RAND_MAX;
        p.y = 100.0*(double)rand()/(double)RAND_MAX;
        p.mass = 1.0 + (double)rand()/(double)RAND_MAX;

        particlesFile.write((char*)&p, sizeof(p));
```

```
    }

    particlesFile.close();

    ifstream inputParticlesFile("particles.dat", ios::in | ios::binary );

    int recordSize = sizeof(p);

    if (inputParticlesFile.is_open())
    {
        inputParticlesFile.seekg(0);
        inputParticlesFile.seekg(5 * recordSize);
        inputParticlesFile.read((char*)&p, sizeof(p));

        cout << "x = " << p.x;
        cout << ", y = " << p.y;
        cout << ", m = " << p.mass << endl;
    }
    else
        cout << "Could not open file." << endl;

    inputParticlesFile.close();
}

#include <iostream>

using namespace std;

int main()
{
    int a = 1;
    int b = 2;
    double c = 3.0;

    cout << a << " " << b;
    cout << " " << c << endl;
    cout << &c << endl;
}

#include <iostream>

using namespace std;

int main()
```

```
{
    int a;
    int b;
    double c;

    cin >> a >> b >> c;

    cout << "a = " << a;
    cout << ", b = " << b;
    cout << ", c = " << c << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Standard output" << endl;
    cerr << "Standard error" << endl;
    clog << "Standard logging" << endl;
}
```

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int main()
{
    double pi = 4 * std::atan(1);
    double x = 0.0;
    double dx = 0.1;

    cout << setw(15) << left << "X";
    cout << setw(10) << right << "f(x)" << endl;
    cout << setfill('-');
    cout << setw(25) << "" << endl;
    cout << setfill(' ');
    cout << setprecision(6) << fixed;
    //cout.unsetf(ios_base::fixed);

    while (x<=pi*0.25)
```

```
    {
        cout << setw(15) << left << x;
        cout << setw(10) << right << sin(x);
        cout << endl;
        x += dx;
    }
}
```

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    bool flag = true;

    cout << "flag = " << flag << endl;
    cout << boolalpha;
    cout << "flag = " << flag << endl;

    int n = 42;

    cout << hex << "n = " << n << endl;
    cout << hex << showbase << "n = " << n << endl;
    cout << oct << "n = " << n << endl;
    cout << dec << "n = " << n << endl;
}
```

ch_data_structures

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

int main()
{
    vector<int> vec;
    vector<int>::iterator it;
```

```
    vec.push_back(5);
    vec.push_back(1);
    vec.push_back(3);
    vec.push_back(4);
    vec.push_back(4);
    vec.push_back(8);

    it = find(vec.begin(), vec.end(), 4);

    if (it!=vec.end())
        cout << "found " << *it << endl;
    else
        cout << "Value not found." << endl;

    it = find(++it, vec.end(), 4);

    if (it!=vec.end())
        cout << "found " << *it << endl;
    else
        cout << "Value not found." << endl;

    it = find(++it, vec.end(), 4);

    if (it!=vec.end())
        cout << "found " << *it << endl;
    else
        cout << "Value not found." << endl;

}

#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

void myfunc(int i)
{
    cout << i << endl;
}

bool greaterThan5(int i)
```

```
{
    return i>5;
}

int main()
{
    vector<int> vec;
    vector<int>::iterator it;

    vec.push_back(5);
    vec.push_back(1);
    vec.push_back(3);
    vec.push_back(4);
    vec.push_back(4);
    vec.push_back(8);

    for_each(vec.begin(), vec.end(), myfunc);

    it = find_if(vec.begin(), vec.end(), greaterThan5);
    cout << "found " << *it << endl;

    sort(vec.begin(), vec.end());

    for_each(vec.begin(), vec.end(), myfunc);

    cout << "max value = " << *max_element(vec.begin(), vec.end()) << endl

    cout << "min value = " << *min_element(vec.begin(), vec.end()) << endl

    fill(vec.begin(), vec.end(), 0);

    for_each(vec.begin(), vec.end(), myfunc);
}

#include <iostream>
#include <deque>

using namespace std;

int main()
{
    deque<int> q;

    for (int i=0; i<=5; i++)
```

```
        q.push_back(i);

    for (int i=6; i<=10; i++)
        q.push_front(i);

    deque<int>::iterator it;

    for (it=q.begin(); it!=q.end(); it++)
        cout << *it << ", ";

    cout << endl;

    cout << "q front = " << q.front() << endl;
    cout << "pop front" << endl;
    q.pop_front();
    cout << "q front = " << q.front() << endl;
    cout << "q back = " << q.back() << endl;
    cout << "pop back" << endl;
    q.pop_back();
    cout << "q back = " << q.back() << endl;
    cout << "q[3] = " << q[3] << endl;
}

#include <iostream>
#include <list>

using namespace std;

int main()
{
    list<int> l;

    for (int i=0; i<=5; i++)
        l.push_back(i);

    for (int i=6; i<=10; i++)
        l.push_front(i);

    list<int>::iterator it;

    for (it=l.begin(); it!=l.end(); it++)
        cout << *it << ", ";

    cout << endl;
```

```
    cout << "l front = " << l.front() << endl;
    cout << "pop front" << endl;
    l.pop_front();
    cout << "l front = " << l.front() << endl;
    cout << "l back = " << l.back() << endl;
    cout << "pop back" << endl;
    l.pop_back();
    cout << "l back = " << l.back() << endl;
}

#include <iostream>
#include <map>
#include <string>

using namespace std;

int main()
{
    map<string,int> m;
    map<string,int>::iterator it;

    m["bob"] = 42;
    m["alice"] = 40;
    m["mike"] = 30;
    m["richard"] = 25;

    for (it=m.begin(); it!=m.end(); it++)
        cout << it->first << ", " << it->second << endl;

    it = m.find("bob");
    cout << "found: " << it->first << ", " << it->second << endl;

    it = m.find("carl");

    if (it!=m.end())
        cout << "found: " << it->first << ", " << it->second << endl;
    else
        cout << "Could not find Carl." << endl;

}

#include <iostream>
#include <map>
```

```
#include <string>

using namespace std;

int main()
{
    map<string,int> m;
    map<string,int>::iterator it;

    m["bob"] = 42;
    m["alice"] = 40;
    m["mike"] = 30;
    m["richard"] = 25;

    for (it=m.begin(); it!=m.end(); it++)
        cout << it->first << ", " << it->second << endl;

    m.erase(m.find("mike"));

    cout << "--" << endl;

    for (it=m.begin(); it!=m.end(); it++)
        cout << it->first << ", " << it->second << endl;

    m.insert(pair<string,int>("carl", 43));

    cout << "--" << endl;

    for (it=m.begin(); it!=m.end(); it++)
        cout << it->first << ", " << it->second << endl;

    m.clear();

    cout << "--" << endl;

    for (it=m.begin(); it!=m.end(); it++)
        cout << it->first << ", " << it->second << endl;

    cout << "m.size() = " << m.size() << endl;

}

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

```
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
    srand((unsigned)time(0));

    vector<int> vec;

    for (int i=0; i<10; i++)
        vec.push_back(rand());

    for (size_t i=0; i<vec.size(); i++)
        cout << vec[i] << endl;
}
```

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
    srand((unsigned)time(0));

    vector<int> vec;

    for (int i=0; i<10; i++)
        vec.push_back(rand());

    vector<int>::iterator it;

    for (it=vec.begin(); it!=vec.end(); it++)
        cout << *it << endl;
}
```

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <algorithm>
```

```
#include <ctime>

using namespace std;

int main()
{
    srand((unsigned)time(0));

    vector<int> vec;

    for (int i=0; i<10; i++)
        vec.push_back(rand());

    sort(vec.begin(), vec.end());
    reverse(vec.begin(), vec.end());

    vector<int>::iterator it;

    for (it=vec.begin(); it!=vec.end(); it++)
        cout << *it << endl;
}
```

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <algorithm>
#include <ctime>

using namespace std;

int main()
{
    srand((unsigned)time(0));

    vector<int> vec;

    for (int i=0; i<10; i++)
        vec.push_back(rand());

    sort(vec.begin(), vec.end());
    reverse(vec.begin(), vec.end());

    vector<int>::iterator it;
```

```
    for (it=vec.begin(); it!=vec.end(); it++)
        cout << *it << endl;
}
```

ch_boost_python

```
#include <boost/python.hpp>

#include <iostream>

using namespace std;

class Particle {
private:
    double m_x;
    double m_y;
    double m_mass;
public:
    Particle(double x, double y);

    void show();
    void move(double dx, double dy);

    void setPosition(double x, double y);
    double x();
    double y();

    void setMass(double m);
    double mass();
};

Particle::Particle(double x, double y)
{
    m_x = x;
    m_y = y;
    m_mass = 1.0;
}

void Particle::show()
{
    cout << "x = " << m_x << ", y = " << m_y << endl;
    cout << "mass = " << m_mass << endl;
}
```

```
void Particle::move(double dx, double dy)
{
    m_x += dx;
    m_y += dy;
}

void Particle::setPosition(double x, double y)
{
    m_x = x;
    m_y = y;
}

double Particle::x()
{
    return m_x;
}

double Particle::y()
{
    return m_y;
}

void Particle::setMass(double m)
{
    m_mass = m;
}

double Particle::mass()
{
    return m_mass;
}

BOOST_PYTHON_MODULE(particles)
{
    using namespace boost::python;

    class_<Particle>("Particle", init<double, double>())
        .def("show", &Particle::show)
        .def("move", &Particle::move)
        .def("setPosition", &Particle::setPosition)
        .def("x", &Particle::x)
        .def("y", &Particle::y)
        .def("setMass", &Particle::setMass)
```

```
        .def("mass", &Particle::mass)
    ;
}
```

ch_matrices

```
#include "calfem.h"

#include <cmath>
#include <set>

arma::mat hooke(TAnalysisType ptype, double E, double v)
{
    using namespace arma;

    mat D;
    switch (ptype) {
        case PLANE_STRESS:
            D.resize(3,3);
            D << 1.0 << v   << 0.0 << endr
            << v   << 1.0 << 0.0 << endr
            << 0.0 << 0.0 << (1.0-v)*0.5 << endr;

            D *= E/(1-v*v);

            break;
        case PLANE_STRAIN:
            D.resize(4,4);
            D << 1.0-v << v     << v     << 0.0 << endr
            << v     << 1.0-v << v     << 0.0 << endr
            << v     << v     << 1.0-v << 0.0 << endr
            << 0.0   << 0.0   << 0.0   << 0.5*(1.0-2*v) << endr;

            D *= E/((1+v)*(1-2*v));

            break;
        default:
            break;
    }
    return D;
}

arma::mat bar2e(arma::rowvec ex, arma::rowvec ey, arma::rowvec ep)
```

```
{
    using namespace arma;

    double E = ep(0);
    double A = ep(1);
    double L = sqrt(pow(ex(1)-ex(0),2)+pow(ey(1)-ey(0),2));
    double C = E*A/L;

    mat Ke_loc(2,2);

    Ke_loc <<  C << -C << endr
           << -C <<  C << endr;

    double nxx = (ex(1)-ex(0))/L;
    double nyx = (ey(1)-ey(0))/L;

    mat G(2,4);

    G << nxx << nyx << 0.0 << 0.0 << endr
      << 0.0 << 0.0 << nxx << nyx << endr;

    mat Ke = G.t()*Ke_loc*G;
    return Ke;
}

double bar2s(arma::rowvec ex, arma::rowvec ey, arma::rowvec ep, arma::rowvec
{
    using namespace arma;

    double E = ep(0);
    double A = ep(1);
    double L = sqrt(pow(ex(1)-ex(0),2)+pow(ey(1)-ey(0),2));
    double C = E*A/L;

    double nxx = (ex(1)-ex(0))/L;
    double nyx = (ey(1)-ey(0))/L;

    mat G(2,4);

    G << nxx << nyx << 0.0 << 0.0 << endr
    << 0.0 << 0.0 << nxx << nyx << endr;

    rowvec temp;
    temp << -C << C;
```

```
    return as_scalar(temp * G * ed.t());
}

void assem(arma::imat topo, arma::mat& K, arma::mat& Ke)
{
    for (int row=0; row<Ke.n_rows; row++)
        for (int col=0; col<Ke.n_cols; col++)
            K(topo(row), topo(col)) += Ke(row,col);
}

void solveq(arma::mat& K, arma::mat&f, arma::irowvec& bcDofs, arma::rowvec&
{
    using namespace std;
    using namespace arma;

    set<int> bc;

    for (int i=0; i<bcDofs.size(); i++)
        bc.insert(bcDofs(i));

    uvec allIndices(K.n_rows-bc.size());
    uvec colIndices;

    colIndices << 0;

    int count = 0;

    for (int i=0; i<K.n_rows; i++)
        if (bc.find(i)==bc.end())
            allIndices(count++) = i;

    mat Ksolve = K(allIndices, allIndices);
    mat fsolve = f(allIndices, colIndices);
    mat asolve = solve(Ksolve, fsolve);

    a.zeros();
    a(allIndices, colIndices) = asolve;

    /* Q=K*asmatrix(a)-f */

    r = K*a-f;
}
```

```
void extractEldisp(arma::imat& edof, arma::mat& a, arma::mat& ed)
{
    int nDofs = edof.n_cols;
    int nElements = edof.n_rows;

    ed.resize(nElements, nDofs);

    for (int i=0; i<nElements; i++)
        for (int j=0; j<nDofs; j++)
            ed(i,j) = a(edof(i,j),0);
}
```

```
#include <iostream>
#include <armadillo>

using namespace std;
using namespace arma;

int main()
{
    mat A = randu<mat>(4,5);
    mat B = randu<mat>(4,5);

    cout << A*B.t() << endl;
}
```

```
#include <iostream>
#include <armadillo>

using namespace std;
using namespace arma;

enum TAnalysisType {PLANE_STRESS, PLANE_STRAIN};

mat hooke(TAnalysisType ptype, double E, double v)
{
    mat D;
    switch (ptype) {
        case PLANE_STRESS:
            D.resize(3,3);
            D << 1.0 << v   << 0.0 << endr
              << v   << 1.0 << 0.0 << endr
```

```
              << 0.0 << 0.0 << (1.0-v)*0.5 << endr;
            break;
        case PLANE_STRAIN:
            D.resize(4,4);
            D << 1.0-v << v     << v     << 0.0 << endr
              << v     << 1.0-v << v     << 0.0 << endr
              << v     << v     << 1.0-v << 0.0 << endr
              << 0.0   << 0.0   << 0.0   << 0.5*(1.0-2*v) << endr;
            break;
        default:
            break;
    }
    return D;
}

int main()
{
    mat Dpstress = hooke(PLANE_STRESS, 2.1e9, 0.35);
    mat Dpstrain = hooke(PLANE_STRAIN, 2.1e9, 0.35);

    cout << "D,pstress = " << endl;
    Dpstress.print();
    cout << "D,pstrain = " << endl;
    Dpstrain.print();
}

#include <iostream>
#include <armadillo>

#include "calfem.h"

using namespace std;
using namespace arma;

int main()
{
    rowvec ex(2);
    rowvec ey(2);
    rowvec ep(2);

    ex << 0.0 << 1.0;
    ey << 0.0 << 1.0;
    ep << 1.0 << 1.0;
```

```
    mat Ke = bar2e(ex, ey, ep);

    cout << Ke << endl;
}

#include <iostream>
#include <armadillo>

#include "calfem.h"

using namespace std;
using namespace arma;

int main()
{
    rowvec ex(2);
    rowvec ey(2);
    rowvec ep(2);
    rowvec ed(4);

    ex << 0.0 << 1.0;
    ey << 0.0 << 1.0;
    ep << 1.0 << 1.0;
    ed << 0.0 << 0.0 << 0.01 << 0.01;

    mat Ke = bar2e(ex, ey, ep);
    cout << "Ke=" << endl;
    cout << Ke << endl;

    double N = bar2s(ex, ey, ep, ed);

    cout << "N = " << N << endl;
}

#include <iostream>
#include <cmath>
#include <armadillo>
#include <set>

#include "calfem.h"

using namespace std;
using namespace arma;
```

```
int main()
{
    // Element topology

    imat edof(10,4);

    edof << 1 << 2 << 5 << 6 << endr
         << 3 << 4 << 7 << 8 << endr
         << 5 << 6 << 9 << 10 << endr
         << 7 << 8 << 11 << 12<< endr
         << 7 << 8 << 5 << 6 << endr
         << 11 << 12 << 9 << 10 << endr
         << 3 << 4 << 5 << 6 << endr
         << 7 << 8 << 9 << 10 << endr
         << 1 << 2 << 7 << 8 << endr
         << 5 << 6 << 11 << 12 << endr;

    edof -= 1;

    // Stiffness matrix

    mat K(12,12);
    K.zeros();

    // Force vector

    mat f(12,1);
    f.zeros();
    f(10,0) = 0.5e6*sin(M_PI/6);
    f(11,0) = -0.5e6*cos(M_PI/6);

    // Material properties

    double A = 25.0e-4;
    double E = 2.1e11;

    rowvec ep(2);

    ep << E << A;

    // Element coordinates

    mat ex(10,2);
    mat ey(10,2);
```

```
ex << 0 << 2 << endr
   << 0 << 2 << endr
   << 2 << 4 << endr
   << 2 << 4 << endr
   << 2 << 2 << endr
   << 4 << 4 << endr
   << 0 << 2 << endr
   << 2 << 4 << endr
   << 0 << 2 << endr
   << 2 << 4 << endr;

ey << 2 << 2 << endr
   << 0 << 0 << endr
   << 2 << 2 << endr
   << 0 << 0 << endr
   << 0 << 2 << endr
   << 0 << 2 << endr
   << 0 << 2 << endr
   << 0 << 2 << endr
   << 2 << 0 << endr
   << 2 << 0 << endr;

// Assemble system

for (int i=0; i<ex.n_rows; i++)
{
    mat Ke = bar2e(ex.row(i), ey.row(i), ep);
    assem(edof.row(i), K, Ke);
}

// Boundary conditions

irowvec bcDofs(4);
bcDofs << 0 << 1 << 2 << 3;

rowvec bcValues(4);
bcValues << 0.0 << 0.0 << 0.0 << 0.0;

// Solution displacment and reaction vector

mat a(K.n_rows, 1);
mat r(K.n_rows, 1);
```

```
    // Solve equation system

    solveq(K, f, bcDofs, bcValues, a, r);

    // Displa

    cout.precision(11);
    cout.setf(ios::fixed);

    cout << "a =" << endl;
    a.raw_print();

    cout << "r =" << endl;
    r.raw_print();

    // Extract element displacements

    mat ed;

    extractEldisp(edof, a, ed);

    cout << "ed = " << endl;
    ed.raw_print();

    // Calculate element forces

    rowvec N(edof.n_rows);

    for (int i=0; i<edof.n_rows; i++)
        N(i) = bar2s(ex.row(i), ey.row(i), ep, ed.row(i));

    cout << "N = " << endl;
    N.raw_print();

}
```

ch_functions

```
#include <iostream>

using namespace std;

void createArray(int**& array, int rows, int cols)
```

```
{
    array = new int*[rows];
    array[0] = new int[rows*cols];

    for (int i=0; i<rows; i++)
        array[i] = &array[0][i*cols];
}

void zeroArray(int**& array, int rows, int cols)
{
    for (int i=0; i<rows; i++)
        for (int j=0; j<cols; j++)
            array[i][j] = 0;
}

void deleteArray(int**& array)
{
    delete array[0];
    delete array;
}

int main()
{
    int** array;

    createArray(array, 4, 8);
    zeroArray(array, 4, 8);

    for (int i=0; i<4; i++)
    {
        for (int j=0; j<8; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    deleteArray(array);
}

#include <iostream>

using namespace std;

int** createArray(int rows, int cols)
```

```
{
    int** array = new int*[rows];
    array[0] = new int[rows*cols];

    for (int i=0; i<rows; i++)
        array[i] = &array[0][i*cols];

    return array;
}

void zeroArray(int**& array, int rows, int cols)
{
    for (int i=0; i<rows; i++)
        for (int j=0; j<cols; j++)
            array[i][j] = 0;
}

void deleteArray(int**& array)
{
    delete array[0];
    delete array;
}

int main()
{
    int** array;

    array = createArray(4, 8);
    zeroArray(array, 4, 8);

    for (int i=0; i<4; i++)
    {
        for (int j=0; j<8; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    deleteArray(array);
}

#include "array_utils.h"

int** createArray(int rows, int cols)
```

```
{
    int** array = new int*[rows];
    array[0] = new int[rows*cols];

    for (int i=0; i<rows; i++)
        array[i] = &array[0][i*cols];

    return array;
}

void zeroArray(int**& array, int rows, int cols)
{
    for (int i=0; i<rows; i++)
        for (int j=0; j<cols; j++)
            array[i][j] = 0;
}

void deleteArray(int**& array)
{
    delete array[0];
    delete array;
}
```

```
#include "array_utils2.h"

int** createArray(int rows, int cols)
{
    int** array = new int*[rows];
    array[0] = new int[rows*cols];

    for (int i=0; i<rows; i++)
        array[i] = &array[0][i*cols];

    return array;
}

void zeroArray(int**& array, int rows, int cols)
{
    for (int i=0; i<rows; i++)
        for (int j=0; j<cols; j++)
            array[i][j] = 0;
}

void deleteArray(int**& array)
```

```
{
    delete array[0];
    delete array;
}
```

```
#include <iostream>

using namespace std;

void simple()
{
    cout << "Hello, from function!" << endl;
}

int main()
{
    simple();
}
```

```
#include <iostream>

using namespace std;

void simple(int a)
{
    cout << "The value of a = " << a << endl;
}

int main()
{
    simple(42);
}
```

```
#include <iostream>

using namespace std;

void simple(int a)
{
    cout << "The value of a = " << a << endl;
    cout << "&a = " << &a << endl;
}
```

```
int main()
{
    int a = 42;
    simple(a);

    cout << "&a = " << &a << endl;
}

#include <iostream>

using namespace std;

void simple(int* a)
{
    cout << "The value of a = " << a << endl;
    cout << "*a = " << *a << endl;
}

int main()
{
    int a = 42;
    simple(&a);
}

#include <iostream>

using namespace std;

void simple(int* a)
{
    *a = 43;
}

int main()
{
    int a = 42;

    cout << "Before function call: a = " << a << endl;
    simple(&a);
    cout << "After function call : a = " << a << endl;
}

#include <iostream>
```

```
using namespace std;

void simple(int& a)
{
    a = 43;
}

int main()
{
    int a = 42;

    cout << "Before function call: a = " << a << endl;
    simple(a);
    cout << "After function call : a = " << a << endl;
}
```

```
#include <iostream>

using namespace std;

void simple(int* a)
{
    for (int i=0; i<4; i++)
        cout << a[i] << ", ";
    cout << endl;
}

int main()
{
    int a[] = { 1, 2, 3, 4 };
    simple(a);
}
```

```
#include <iostream>

using namespace std;

void simple(int* a)
{
    a[3] = 42;
}

int main()
{
```

```
    int a[] = { 1, 2, 3, 4 };
    simple(a);
    cout << "a[3] = " << a[3] << endl;
}

#include <iostream>

#include "array_utils.h"

using namespace std;

int main()
{
    int** array;

    array = createArray(4, 8);
    zeroArray(array, 4, 8);

    for (int i=0; i<4; i++)
    {
        for (int j=0; j<8; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    deleteArray(array);
}

#include <iostream>

#include "array_utils2.h"

using namespace std;

int main()
{
    int** array;

    array = createArray(4, 8);
    zeroArray(array, 4, 8);

    for (int i=0; i<4; i++)
    {
```

```
        for (int j=0; j<8; j++)
            cout << array[i][j] << ", ";

        cout << endl;
    }

    deleteArray(array);
}
```

ch_control_structures

```
#include <iostream>

using namespace std;

int main()
{
    for (int i=1; i<=10; i++)
    {
        cout << "i = " << i << endl;

        if (i==5)
            cout << "i == 5" << endl;
    }
}
```

```
#include <iostream>

using namespace std;

int main()
{
    for (int i=1; i<=10; i++)
    {
        if (i==5)
            cout << "i == 5" << endl;
        else
            cout << "i != 5" << endl;
    }
}
```

```
#include <iostream>
```

```
using namespace std;

int main()
{
    for (int i=1; i<=10; i++)
    {
        cout << "i = " << i << ": ";

        if (i==5)
            cout << "i == 5" << endl;
        else if ( (i>=2)&&(i<=3) )
            cout << "2 <= i <= 3" << endl;
        else
            cout << "-" << endl;
    }
}
```

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
    enum colorType { RED, GREEN, BLUE, YELLOW, ORANGE };
    srand((unsigned)time(0));


    for (int i=0; i<4; i++)
    {
        colorType color = colorType(rand()%5);
        switch (color)
        {
            case RED:
                cout << "Color is red." << endl;
                break;
            case GREEN:
                cout << "Color is green." << endl;
                break;
            default:
                cout << "Color is either BLUE, YELLOW or ORANGE." << endl;
                break;
        }
```

```
    }
}
```

ch_iterating

```
#include <iostream>

using namespace std;

int main()
{
    int counter = 1;

    do
    {
        cout << "counter = " << counter << endl;
        counter = counter + 1;
    }
    while (counter<=10);
}
```

```
#include <iostream>

using namespace std;

int main()
{
    for (int counter = 1; counter<=10; counter++)
        cout << "counter = " << counter << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int sum = 0;

    for (int i=0; i<=10000; sum += ++i);

    cout << "sum = " << sum << endl;
}
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int counter = 1;

    while (counter <= 10)
    {
        cout << "counter = " << counter << endl;

        if (counter == 5)
        {
            counter = counter + 2;
            continue;
        }

        if (counter == 9)
            break;

        counter = counter + 1;
    }
}
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int counter = 1;

    while (counter <= 10)
    {
        cout << "counter = " << counter << endl;
        counter = counter + 1;
    }
}
```

ch_variables

```
#include <iostream>

using namespace std;

int main()
{
    int a[2];

    a[0] = 1;
    a[1] = 2;

    cout << a[0] << ", " << a[1] << endl;

    double b[] = { 1.0, 2.0, 3.0, 4.0 };
    b[3] = 42.0;

    cout << b[0] << ", " << b[1] << ", " << b[2] << ", " << b[3] << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a[] = {1,2,3,4};
    int* b;

    a[3] = 42;

    cout << a[0] << ", " << a[1] << ", " << a[2] << ", " << a[3] << endl;

    b = a;

    cout << b[0] << ", " << b[1] << ", " << b[2] << ", " << b[3] << endl;
}
```

```
#include <iostream>

using namespace std;
```

```
int main()
{
    char c;
    unsigned char uc;
    signed char sc;

    c = 'a';
    uc = 129;
    sc = 130;

    cout << "c = " << c << " int(c) = " << int(c);
    cout << " sizeof(c) = " << sizeof(c) << endl;
    cout << "uc = " << uc << " int(uc) = " << int(uc);
    cout << " sizeof(c) = " << sizeof(c) << endl;
    cout << "sc = " << sc << " int(sc) = " << int(sc);
    cout << " sizeof(c) = " << sizeof(c) << endl;
}
```

```
#include <iostream>

using namespace std;

const int max_size = 4;

int main()
{
    int a[max_size];

    for (int i=0; i<max_size; i++)
        a[i] = 0;
}
```

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int main()
{
    double pi = 4 * std::atan(1);

    float f;
    double d;
```

```
    long double ld;

    f = pi;
    d = pi;
    ld = pi;

    cout << setprecision(15) << "f = " << f << endl;
    cout << setprecision(15) << "d = " << d << endl;
    cout << setprecision(15) << "ld = " << ld << endl;
}

#include <iostream>

using namespace std;

int main()
{
    enum { RED, GREEN, BLUE, YELLOW };

    cout << "RED = " << RED << endl;
    cout << "GREEN = " << GREEN << endl;
    cout << "BLUE = " << BLUE << endl;
    cout << "YELLOW = " << YELLOW << endl;

    enum flavor { VANILLA, CHOCOLATE, ORANGE, STRAWBERRY };

    flavor selectedFlavor = ORANGE;

    cout << "selectedFlavor = " << selectedFlavor << endl;

    enum status { FINISHED = 0, ERROR = 15 };

    status currentStatus = ERROR;

    cout << "currentStatus = " << currentStatus << endl;
}

#include <iostream>

using namespace std;

int main()
{
    int a;
```

```
    unsigned int b;
    long int c;
    unsigned long int d;

    a = -1; b = -1;
    c = -1; d = -1;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
    cout << "d = " << d << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a;
    unsigned int b;
    long int c;
    unsigned long int d;

    a = -1; b = -1;
    c = -1; d = -1;

    cout << "a = " << a << ", sizeof(a) = " << sizeof(a) << endl;
    cout << "b = " << b << ", sizeof(b) = " << sizeof(b) << endl;
    cout << "c = " << c << ", sizeof(c) = " << sizeof(c) << endl;
    cout << "d = " << d << ", sizeof(d) = " << sizeof(d) << endl;
}
```

```
#include <iostream>
#include <limits>

using namespace std;

int main()
{
    cout << "Max, char " << int(numeric_limits<char>::max()) << endl;
    cout << "Min, char " << int(numeric_limits<char>::min()) << endl;
    cout << "Is char signed " << numeric_limits<char>::is_signed << endl;
    cout << "Max, unsigned char " << int(numeric_limits<unsigned char>::max()
```

```
    cout << "Min, unsigned char " << int(numeric_limits<unsigned char>::mi
    cout << "Max, short " << numeric_limits<short>::max() << endl;
    cout << "Min, short " << numeric_limits<short>::min() << endl;
    cout << "Max, int " << numeric_limits<int>::max() << endl;
    cout << "Min, int " << numeric_limits<int>::min() << endl;
    cout << "Max, long " << numeric_limits<long>::max() << endl;
    cout << "Min, long " << numeric_limits<long>::min() << endl;
    cout << "Max, float " << numeric_limits<float>::max() << endl;
    cout << "Min, float " << numeric_limits<float>::min() << endl;
    cout << "Max, double " << numeric_limits<double>::max() << endl;
    cout << "Min, double " << numeric_limits<double>::min() << endl;
    cout << "Max, long double " << numeric_limits<long double>::max() << en
    cout << "Min, long double " << numeric_limits<long double>::min() << en
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a;
    int* b;

    a = 42;
    b = &a;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "&a = " << &a << endl;
    cout << "*b = " << *b << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a[] = {0, 1, 2, 3};
    int* b;
    int* c;

    b = a;
```

```
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a[0] = " << a[0] << endl;
    cout << "b[0] = " << b[0] << endl;
    cout << "*b = " << *b << endl;
    cout << "*a = " << *a << endl;

    c = &a[2];

    cout << "c = " << c << endl;
    cout << "*c = " << *c << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a = 42;
    int& b = a;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "&a = " << &a << endl;
    cout << "&b = " << &b << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    struct coord3D {
        double x;
        double y;
    };

    coord3D c1;
    coord3D c2;

    c1.x = 0.0;
```

```
    c1.y = 0.0;
    c2.x = 1.0;
    c2.y = 1.0;

    cout << "c1.x = " << c1.x << ", c1.y = " << c1.y << endl;
    cout << "c2.x = " << c2.x << ", c2.y = " << c2.y << endl;
    cout << "sizeof(c1) = " << sizeof(c1) << endl;
}

#include <iostream>

using namespace std;

int main()
{
    int a, b, c;

    a = 40;
    b = 2;

    c = a + b;

    cout << "c = a + b = " << c << endl;
}

#include <iostream>

using namespace std;

int main()
{
    int a, b, c;
    double d, e;

    a = 40;
    b = 2;
    d = 52.5654;

    // d will be truncated
    c = a + b + d;

    // a and be will be upcasted to double
    e = a + b + d;
```

```
    cout << "c = a + b + d = " << c << endl;
    cout << "e = a + b + d = " << e << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    char a;
    a = 'a';

    cout << a << endl;

    char line[] = "This is a character string";

    cout << "'" << line << "'" << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a[] = {1,2,3,4};

    a[3] = 42;

    cout << a[0] << ", " << a[1] << ", " << a[2] << ", " << a[3] << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a[] = {1,2,3,4};
    int* b;

    a[3] = 42;
```

```
    cout << a[0] << ", " << a[1] << ", " << a[2] << ", " << a[3] << endl;

    b = a;

    cout << b[0] << ", " << b[1] << ", " << b[2] << ", " << b[3] << endl;
}
```

ch_cpp_fortran

```
#include <iostream>
#include <armadillo>

using namespace std;
using namespace arma;

extern "C" void multiply(double a[], double b[], double c[], int a_rows, in

int main()
{
    mat a = randu<mat>(5,5);
    mat b = randu<mat>(5,5);
    mat c(5,5);

    cout << "a = " << endl;
    a.print();
    cout << "b = " << endl;
    b.print();

    multiply(a.memptr(), b.memptr(), c.memptr(), a.n_rows, a.n_cols, b.n_r

    cout << "c = " << endl;
    c.print();
}
```

ch_oop

```
#include "circle.h"

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

```
using namespace std;

Circle::Circle(double x, double y, double radius)
:Shape(x, y)
{
    this->setName("Circle");
    m_radius = radius;
}

Circle::~Circle()
{
    cout << "Circle destructor called." << endl;
}

void Circle::print()
{
    Shape::print();
    cout << "radius = " << m_radius << endl;
}

double Circle::area()
{
    double pi = 4 * std::atan(1);
    return pow(m_radius,2)*pi;
}

double Circle::radius()
{
    return m_radius;
}

void Circle::setRadius(double radius)
{
    m_radius = radius;
}


#include <iostream>

using namespace std;

class Point {
private:
```

```
    double m_x;
    double m_y;
public:
    Point(double x, double y);

    void print();

    void setPosition(double x, double y);
    double x();
    double y();
};

Point::Point(double x, double y)
{
    m_x = x;
    m_y = y;
}

void Point::print()
{
    cout << "x = " << m_x << ", y = " << m_y << endl;
}

int main()
{
    Point p0 = Point(0.0, 0.0);
    Point p1 = Point(1.0, 1.0);

    p0.print();
    p1.print();
}

#include <iostream>

using namespace std;

class Point {
private:
    double m_x;
    double m_y;
public:
    Point(double x, double y);

    void print();
```

```
    void setPosition(double x, double y);
    double x();
    double y();
};

Point::Point(double x, double y)
{
    m_x = x;
    m_y = y;
}

void Point::print()
{
    cout << "x = " << m_x << ", y = " << m_y << endl;
}

void Point::setPosition(double x, double y)
{
    m_x = x;
    m_y = y;
}

double Point::x()
{
    return m_x;
}

double Point::y()
{
    return m_y;
}

int main()
{
    Point p0 = Point(0.0, 0.0);
    Point p1 = Point(1.0, 1.0);

    cout << "p0.x() = " << p0.x() << endl;
    cout << "p0.y() = " << p0.y() << endl;

    p1.setPosition(0.5, 0.5);

    cout << "p1.x() = " << p1.x() << endl;
```

```
    cout << "p1.y() = " << p1.y() << endl;

    Point* p3 = new Point(2.0, 2.0);
    p3->setPosition(1.5, 1.5);
    cout << "p3->x() = " << p3->x() << endl;
}

#include "composite.h"

#include <iostream>
#include <cmath>
#include <algorithm>

using namespace std;

Composite::Composite(double x, double y)
:Shape(x, y)
{
    this->setName("Composite");
}

Composite::~Composite()
{
    cout << "Composite destructor called." << endl;
    this->clear();
}

void Composite::print()
{
    Shape::print();
    cout << "Number of shapes = " << m_shapes.size() << endl;
}

double Composite::area()
{
    double totalArea = 0.0;

    vector<Shape*>::iterator it;

    for (it=m_shapes.begin(); it!=m_shapes.end(); it++)
        totalArea += (*it)->area();

    return totalArea;
}
```

```
void Composite::add(Shape* shape)
{
    m_shapes.push_back(shape);
}

void Composite::remove(Shape* shape)
{
    vector<Shape*>::iterator it;

    it = find(m_shapes.begin(), m_shapes.end(), shape);

    if (it!=m_shapes.end())
    {
        Shape* shape = *it;
        m_shapes.erase(it);
        delete shape;
    }
}

void Composite::clear()
{
    vector<Shape*>::iterator it;

    for(it=m_shapes.begin(); it!=m_shapes.end(); it++)
        delete *it;

    m_shapes.clear();
}

int Composite::count()
{
    return m_shapes.size();
}

Shape* Composite::at(int idx)
{
    if ((idx>=0)&&(idx<m_shapes.size()))
        return m_shapes[idx];
    else
        return 0;
}

#include "point.h"
```

```
#include <iostream>

using namespace std;

Point::Point(double x, double y)
:Shape(x, y)
{
    this->setName("Point");
}

Point::~Point()
{
    cout << "Point destructor called." << endl;
}

#include "rectangle.h"

#include <iostream>
#include <cmath>

using namespace std;

Rectangle::Rectangle(double x, double y, double width, double height)
:Shape(x, y)
{
    this->setName("Rectangle");
    m_width = width;
    m_height = height;
}

Rectangle::~Rectangle()
{
    cout << "Rectangle destructor called." << endl;
}


void Rectangle::print()
{
    Shape::print();
    cout << "width = " << m_width << endl;
    cout << "height = " << m_height << endl;
}
```

```
double Rectangle::area()
{
    return m_width*m_height;
}

double Rectangle::width()
{
    return m_width;
}

void Rectangle::setWidth(double width)
{
    m_width = width;
}

void Rectangle::setHeight(double height)
{
    m_height = height;
}
```

```
#include "shape.h"

#include <iostream>

using namespace std;

Shape::Shape(double x, double y)
{
    cout << "Shape created." << endl;
    m_x = x;
    m_y = y;
    m_name = "Shape";
}

Shape::~Shape()
{
    cout << "Shape destructor called." << endl;
}

void Shape::print()
{
    cout << "-------------------------" << endl;
    cout << "Shape type: " << m_name << endl;
```

```
    cout << "x = " << m_x << ", y = " << m_y << endl;
    cout << "area = " << this->area() << endl;
}

void Shape::setPosition(double x, double y)
{
    m_x = x;
    m_y = y;
}

double Shape::x()
{
    return m_x;
}

double Shape::y()
{
    return m_y;
}

double Shape::area()
{
    return 0.0;
}

void Shape::setName(const std::string& name)
{
    m_name = name;
}

std::string Shape::name()
{
    return m_name;
}

#include <iostream>

using namespace std;

#include "point.h"
#include "circle.h"
#include "rectangle.h"

int main()
```

```
{
    Point p0(1.0, 1.0);
    Point p1(1.0, 1.0);

    p0.print();
    p1.print();

    Circle c0(0.5, 1.0, 2.0);
    c0.print();

    Rectangle r0(0.0, 0.0, 2.0, 1.0);
    r0.print();

    Rectangle* rect = new Rectangle(0.0, 0.0, 1.0, 2.0);
    rect->print();
    delete rect;

}

#include <iostream>
#include <vector>
#include <memory>

using namespace std;

#include "point.h"
#include "circle.h"
#include "rectangle.h"

int main()
{
    vector<Shape*> shapes;
    vector<Shape*>::iterator si;

    cout << "adding objects ---" << endl;

    shapes.push_back(new Point(0.0, 0.0));
    shapes.push_back(new Circle(1.0, 0.0, 2.0));
    shapes.push_back(new Rectangle(0.0, 1.0, 2.0, 1.0));

    for (si=shapes.begin(); si!=shapes.end(); si++)
        (*si)->print();

    for (si=shapes.begin(); si!=shapes.end(); si++)
```

```
        delete *si;

    shapes.clear();
}

#include <iostream>
#include <vector>
#include <memory>

using namespace std;

#include "point.h"
#include "circle.h"
#include "rectangle.h"
#include "composite.h"

int main()
{
    Composite* composite = new Composite(0.0, 0.0);

    cout << "adding objects ---" << endl;

    composite->add(new Point(0.0, 0.0));
    composite->add(new Circle(1.0, 0.0, 2.0));
    composite->add(new Rectangle(0.0, 1.0, 2.0, 1.0));

    composite->print();

    for (int i=0; i<composite->count(); i++)
        composite->at(i)->print();

    delete composite;
}

#include <iostream>
#include <vector>
#include <memory>

using namespace std;

#include "point.h"
#include "circle.h"
#include "rectangle.h"
```

```
int main()
{
    vector<Shape> shapes;
    vector<Shape>::iterator si;

    cout << "adding objects ---" << endl;

    shapes.push_back(Point(0.0, 0.0));
    shapes.push_back(Circle(1.0, 0.0, 2.0));
    shapes.push_back(Rectangle(0.0, 1.0, 2.0, 1.0));

    for (si=shapes.begin(); si!=shapes.end(); si++)
        (*si).print();

    shapes.clear();
}
```

```
#include <iostream>
#include <memory>

using namespace std;

#include "point.h"
#include "circle.h"
#include "rectangle.h"

int main()
{
    Point p0(1.0, 1.0);
    Point p1(1.0, 1.0);

    p0.print();
    p1.print();

    Circle c0(0.5, 1.0, 2.0);
    c0.print();

    Rectangle r0(0.0, 0.0, 2.0, 1.0);
    r0.print();

    Rectangle* rect = new Rectangle(0.0, 0.0, 1.0, 2.0);
    rect->print();
    delete rect;
```

```
    std::unique_ptr<Rectangle> rect2(new Rectangle(1.0, 1.0, 3.0, 3.0));
    rect2->print();

    std::unique_ptr<Rectangle> rect3(std::move(rect2));
    rect3->print();

    //  rect2->print();  // Not allowed

    std::shared_ptr<Rectangle> rect4(new Rectangle(2.0, 2.0, 4.0, 4.0));
    std::shared_ptr<Rectangle> rect5(nullptr);

    rect5 = rect4;

    std::shared_ptr<Rectangle> rect6(rect5);

    rect5->print();
    rect4->print();

    cout << rect4.use_count() << endl;

}

#include <iostream>
#include <memory>

using namespace std;

#include "point.h"
#include "circle.h"
#include "rectangle.h"

int main()
{
    Point p0(1.0, 1.0);
    Point p1(1.0, 1.0);

    p0.print();
    p1.print();

    Circle c0(0.5, 1.0, 2.0);
    c0.print();

    Rectangle r0(0.0, 0.0, 2.0, 1.0);
    r0.print();
```

```
    Rectangle* rect = new Rectangle(0.0, 0.0, 1.0, 2.0);
    rect->print();
    delete rect;

    std::unique_ptr<Rectangle> rect2(new Rectangle(1.0, 1.0, 3.0, 3.0));
    //std::unique_ptr<Rectangle> rect2 = std::make_unique<Rectangle>(1.0, 1.0
    rect2->print();

    std::unique_ptr<Rectangle> rect3(std::move(rect2));
    rect3->print();

    //  rect2->print();  // Not allowed

    std::shared_ptr<Rectangle> rect4(new Rectangle(2.0, 2.0, 4.0, 4.0));
    std::shared_ptr<Rectangle> rect5(nullptr);

    rect5 = rect4;

    std::shared_ptr<Rectangle> rect6(rect5);

    rect5->print();
    rect4->print();

    cout << rect4.use_count() << endl;

}
```

ch_operators

```
#include <iostream>

using namespace std;

int main()
{
    int a, b;

    a = 42;
    b = 26;

    a += b; // a = a + b
```

```
    cout << "a = " << a << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int a, b, c;

    a = 42;

    b = ++a;
    c = a++;

    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int number;

    cout << "Enter a number : ";

    cin >> number;

    int outValue = (number>50) ? 42 : 21;

    cout << "outValue = " << outValue;
}
```

ch_strings

```
#include <iostream>
#include <string>
```

```
using namespace std;

int main()
{
    string s = "hello";
    cout << "a = '" << s << "'" << endl;
}
```

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string s1 = "hello";
    string s2 = ", world";
    string s3 = s1 + s2;
    cout << "s3 = '" << s3 << "'" << endl;
    cout << "s3.length() = " << s3.length() << endl;
    cout << "s3[7] = " << s3[7] << endl;
    cout << "s3.at(7) = " << s3.at(7) << endl;
}
```

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string s1 = "hello";
    string s2 = ", world";
    string s3 = s1 + s2;

    s3.append(". Strings in C++ are great!");
    cout << "s3 = " << s3 << endl;

    s3.replace(33, 6, "nice! ");
    cout << "s3 = " << s3 << endl;

    s3.insert(33, "great and ");
    cout << "s3 = " << s3 << endl;
```

```
    string s4 = s3.substr(33, 6);
    cout << "s4 = " << s4 << endl;

}
```

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string s = "The quick brown fox jumps over the lazy dog.";

    int p0 = s.find("o");
    int p1 = s.find("o", p0+1);

    cout << "The first 'o' is at position " << p0 << endl;
    cout << "The next 'o' is at position " << p1 << endl;
}
```

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    const char* cString;
    char cString2[128];

    string cppString = "Hello, world!";

    cString = cppString.c_str();

    cout << "cString = " << cString << endl;

    strncpy(cString2, cppString.c_str(), cppString.length()+1);

    cout << "cString2 = " << cString2 << endl;
}
```

ch_vector_and_lists

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

using namespace std;

int main()
{
    vector<double> vec;

    for (int i=0; i<10; i++)
        vec.push_back(i);

    for (int i=0; i<10; i++)
        cout << vec[i] << endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <ctime>

using namespace std;

int main()
{
    srand((unsigned)time(0));

    vector<int> vec;

    for (int i=0; i<10; i++)
        vec.push_back(rand());

    sort(vec.begin(), vec.end());
    reverse(vec.begin(), vec.end());

    vector<int>::iterator it;

    for (it=vec.begin(); it!=vec.end(); it++)
        cout << *it << endl;
}
```

```
#include <iostream>
```

```
#include <vector>
#include <algorithm>
#include <ctime>

using namespace std;

int main()
{
    srand((unsigned)time(0));

    vector<int> vec;

    for (int i=0; i<10; i++)
        vec.push_back(rand());

    sort(vec.begin(), vec.end());
    reverse(vec.begin(), vec.end());

    for (auto v : vec)
        cout << v << endl;

}
```