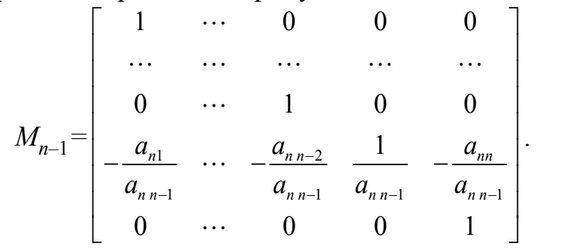
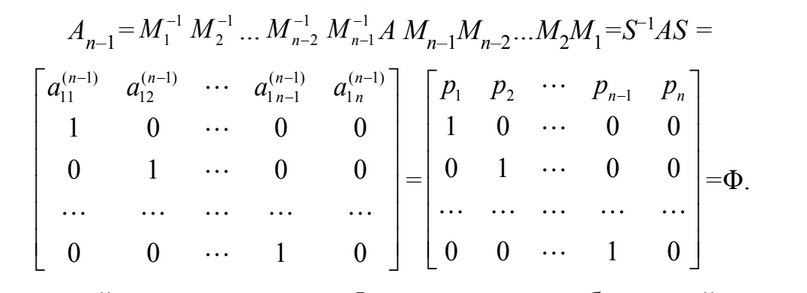
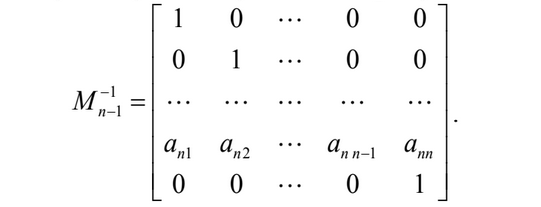
**Лабораторная работа №7**

**Задача.**

Разработать программу приведения матрицы к канонической

форме Фробениуса методом Данилевского (регулярный случай); сохранить матрицы, используемые для получения собственных векторов.





**Входные данные.**

Size = 4, eps = 0.0000001

**Листинг программы.**

#include <iostream>  
#include <valarray>  
#include <chrono>  
  
using namespace std;  
const int n = 4;  
float \*A;  
float \*M;  
float \*min\_M;  
  
float randomFloat(int a, int b)  
{  
 return (float)(a + (rand() % (b - a))) + (float)(rand()) / (float)(RAND\_MAX);  
}  
  
void FillA() {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 A[i \* n + j] =randomFloat(-50,50);  
 }  
 }  
}  
float \* MatrixMultiplication(const float \*A\_, const float \*X) {  
  
 float \* F = new float[n \* n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 float elem = 0;  
 for (int k = 0; k < n; k++) {  
 elem += A\_[i \* n + k] \* X[k \* n + j];  
 }  
 F[i\*n+j] = elem;  
 }  
 }  
 return F;  
}  
  
void MatrixMultiplication(float \*&F, const float \*A\_, const float \*X) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 float elem = 0;  
 for (int k = 0; k < n; k++) {  
 elem += A\_[i \* n + k] \* X[k \* n + j];  
 }  
 F[i\*n+j] = elem;  
 }  
 }  
}  
void printMatrix(int size, float \*Mat){  
 for(int i=0;i<size;i++){  
 for(int j=0;j<size;j++){  
 cout<<Mat[size\*i+j]<<" ";  
 }  
 cout<<endl;  
 }  
}  
void Fill\_M\_k\_part(int k,float \*&M){  
 for(int i=0;i<n;i++){  
 if(i!=k){  
 for(int j=0;j<n;j++){  
 if(i!=j){  
 M[k\*n\*n+i\*n+j]=0;  
 }else{  
 M[k\*n\*n+i\*n+j]=1;  
 }  
 }  
  
 }else{  
 for(int j=0;j<n;j++){  
 if(j!=k){  
 M[k\*n\*n+i\*n+j]=-A[n\*(k+1)+j]/A[n\*(k+1)+k];  
 }else{  
 M[k\*n\*n+i\*n+j]=1/A[n\*(k+1)+k];  
 }  
 }  
 }  
 }  
}  
void reversed\_M\_k(int k, float \*&min\_M){  
 for(int i=0;i<n;i++){  
 if(i!=k){  
 for(int j=0;j<n;j++){  
 if(i!=j){  
 min\_M[k\*n\*n+i\*n+j]=0;  
 }else{  
 min\_M[k\*n\*n+i\*n+j]=1;  
 }  
 }  
 }else{  
 for(int j=0;j<n;j++){  
 min\_M[k\*n\*n+i\*n+j]=A[n\*(k+1)+j];  
 }  
 }  
 }  
}  
int main() {  
*// srand(time(0));* A=new float [n\*n];  
 FillA();  
  
 float trace=0;  
 for(int i=0;i<n;i++){  
 trace+=A[i \* n + i];  
 }  
  
  
 cout<<"Matrix A:"<<endl;  
 printMatrix(n, A);  
 M=new float [n\*n\*(n-1)];  
 min\_M=new float[n\*n\*(n-1)];  
 int k=n-2;  
 float a;  
 float \*min\_M\_k=new float[n\*n];  
 float \*M\_k=new float[n\*n];  
 while(k>=0){  
 a=A[n\*(k+1)+k];  
 if(a!=0 && abs(a)>=pow(10,-8)){  
 Fill\_M\_k\_part(k,M);  
 reversed\_M\_k(k, min\_M);  
 for(int i=0;i<n;i++){  
 for(int j=0;j<n;j++){  
 min\_M\_k[i\*n+j]=min\_M[k\*n\*n+n\*i+j];  
 M\_k[i\*n+j]=M[k\*n\*n+n\*i+j];  
 }  
 }  
 float \* temp = MatrixMultiplication(A,M\_k);  
 MatrixMultiplication(A,min\_M\_k, temp);  
  
  
 }else{  
 cout<<"The leading element is 0, rewriting matrix"<<endl;  
 FillA();  
 cout<<"A:"<<endl;  
 printMatrix(n, A);  
  
 trace=0;  
 for(int i=0;i<n;i++){  
 trace+=A[i \* n + i];  
 }  
 k=n-2;  
 }  
 k--;  
 }  
 cout<<endl<<"Frobenius form\n";  
 printMatrix(n, A);  
 for(int i=n-1;i>0;i--){  
 cout<<"M"<<i<<endl;  
 for(int j=0;j<n;j++){  
 for(int t=0;t<n;t++){  
 cout<<M[(i-1)\*n\*n+j\*n+t] <<" ";  
 }  
 cout<<endl;  
 }  
 cout<<endl;  
 }  
 cout<<"p1 "<<A[0]<<endl;  
 cout << "SpA " << trace;  
  
 return 0;  
}

**Выходные данные.**

Matrix A:

-8.43641 -15.1913 19.4799 28.896

12.7466 -44.1411 31.5135 11.015

45.3645 -22.8341 41.4457 -47.9953

42.3779 -28.4288 -31.3928 -2.33695

Frobenius form

-13.4687 4169.29 78578.4 -2.02755e+06

1 0 0 0

0 1 0 0

0 0 1 0

M3

1 0 0 0

0 1 0 0

1.34992 -0.905583 -0.0318544 -0.0744423

0 0 0 1

M2

1 0 0 0

1.55468 0.000389118 -0.0160902 -0.980659

0 0 1 0

0 0 0 1

M1

-6.36209e-05 0.00125425 0.210859 -2.08998

0 1 0 0

0 0 1 0

0 0 0 1

p1 -13.4687

SpA -13.4687