#include<iostream>

#include<clocale>

using namespace std;

int\*\* PereChislenie(int key);

void Watch(int \*\*M, int mM);

void KonZap();

void menu();

int\*\* InOne(int \*\*A, int \*\*B, int mA, int mB); // Объединение графиков

int\*\* Peresek(int \*\*A, int \*\*B, int mA, int mB); // Пересечение графиков

int\*\* Pa3HocTb(int \*\*U, int \*\*W, int mU, int mW); // Разность

int\*\* SimRas(int \*\*A, int \*\*B, int mA, int mB); // Симметричная разность A u B

int\*\* Inversion(int \*\*M, int mM); // Инверсия

int\*\* Exposition(int \*\*A, int \*\*B, int mA, int mB); // Композиция графиков

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Введите мощность графика А:\t";

int mA;

cin >> mA; // Пользователь вводит мощность n первого графика A.

int \*\*A = PereChislenie(mA); // Пользователь последовательно вводит n пар первого графика.

cout << "Введите мощность графика B:\t";

int mB;

cin >> mB; //Пользователь вводит мощность m второго графика B.

int \*\*B = PereChislenie(mB);// Пользователь последовательно вводит m пар второго графика.

KonZap();

while (true)

{

system("cls");

cout << "A = "; Watch(A, mA);

cout << "B = "; Watch(B, mB);

menu(); //меню

int Operaziya;

cin >> Operaziya; //Пользователь выбирает исполняемую операцию:

system("cls");

switch (Operaziya)

{

case 1: // Пересечение графиков

{

cout << "Пересечение графиков :" << endl;

int mD = mA + mB + 1;

int \*\*D = new int\*[mD]; // Создаём пустой график D.

for (int i = 0; i <= mD; i++)

D[i] = new int[3];

D = Peresek(A, B, mA, mB);

mD = D[0][0];

Watch(D, mD); //Выводим на экран график D

system("pause");

}

break;

case 2: // Объединение графиков

{

cout << "Объединение графиков :" << endl;

int mC = mA + mB + 1;

int \*\*C = new int\*[mC]; //Создаём пустой график С

for (int i = 0; i <= mC; i++)

C[i] = new int[3];

C = InOne(A, B, mA, mB);

mC = C[0][0];

Watch(C, mC);//Выводим на экран график C

system("pause");

}

break;

case 3: // Разность A и В

{

cout << "Разность A и В :" << endl;

int mR = mA + 1;

int \*\*R = new int\*[mR];// Создаём пустой график R.

for (int i = 0; i <= mR; i++)

R[i] = new int[3];

R = Pa3HocTb(A, B, mA, mB);

mR = R[0][0];

Watch(R, mR);//Выводим на экран график R

system("pause");

}

break;

case 4:// Разность B и A

{

cout << "Разность B и A :" << endl;

int mR = mB + 1;

int \*\*R = new int\*[mR];// Создаём пустой график R.

for (int i = 0; i <= mR; i++)

R[i] = new int[3];

R = Pa3HocTb(B, A, mB, mA);

mR = R[0][0];

Watch(R, mR);//Выводим на экран график R

system("pause");

}

break;

case 5:// Симметричная разность A u B

{

cout << "Симметричная разность A u B :" << endl;

int mS = mB + mA + 1;

int \*\*S = new int\*[mS];// Создаём пустой график S.

for (int i = 0; i <= mS; i++)

S[i] = new int[3];

S = SimRas(B, A, mB, mA);

mS = S[0][0];

Watch(S, mS);//Выводим на экран график S

system("pause");

}

break;

case 6:// Инверсия А

{

cout << "Инверсия А :" << endl;

int mI = mA;

int \*\*I = new int\*[mI];//Создаём пустой график I

for (int i = 0; i <= mI; i++)

I[i] = new int[3];

I = Inversion(A, mA);

mI = I[0][0];

Watch(I, mI);//Выводим на экран график I

system("pause");

}

break;

case 7:// Инверсия В

{

cout << "Инверсия B :" << endl;

int mI = mB;

int \*\*I = new int\*[mI]; //Создаём пустой график I

for (int i = 0; i <= mI; i++)

I[i] = new int[3];

I = Inversion(B, mB);

mI = I[0][0];

Watch(I, mI);//Выводим на экран график I

system("pause");

}

break;

case 8:// Композиция графиков A и B

{

cout << "Композиция графиков A и B :" << endl;

int mK = mB\*mA + 1;

int \*\*K = new int\*[mK];//Создаём пустой график K

for (int i = 0; i <= mK; i++)

K[i] = new int[3];

K = Exposition(A, B, mA, mB);

mK = K[0][0];

Watch(K, mK);//Выводим на экран график K

system("pause");

}

break;

case 9:// Композиция графиков B и A

{

cout << "Композиция графиков B и A :" << endl;

int mK = mB\*mA + 1;

int \*\*K = new int\*[mK]; //Создаём пустой график K

for (int i = 0; i <= mK; i++)

K[i] = new int[3];

K = Exposition(B, A, mB, mA);

mK = K[0][0];

Watch(K, mK); //Выводим на экран график K

system("pause");

}

break;

case 0:// Выход

return 0; //Завершаем программу

}

}

}

int\*\* PereChislenie(int mM)

{

int \*\*M;

M = new int \*[mM];

for (int i = 1; i <= mM; i++)

M[i] = new int[3];

if (mM < 1)

{

cout << "Пустой график!" << endl;

return M;

}

cout << "Введите пары графика:\n";

for (int i = 1; i <= mM; i++)

{

cout << "Пара №" << i << endl;

for (int j = 1; j < 3; j++)

{

cin >> M[i][j]; //Пользователь последовательно вводит пары графика

}

}

cout << endl;

return M;

}

void Watch(int \*\*M, int mM)

{

cout << "{";

for (int i = 1; i <= mM; i++)

{

cout << "<";

for (int j = 1; j < 3; j++)

{

cout << M[i][j];

if (j == 1)

cout << ", ";

}

cout << ">";

if (i < mM)

cout << ", ";

}

cout << "}" << endl;

}

void KonZap()

{

cout << "Графики заполнены." << endl;

system("pause");

system("cla");

}

void menu()

{

cout << endl << "Выберите операцию:" << endl;

cout << " 1. Пересечение графиков\n 2. Объединение графиков\n 3. Разность A и В\n 4. Разность B и A\n 5. Симметричная разность A u B\n 6. Инверсия А\n 7. Инверсия В\n 8. Композиция графиков A и B\n 9. Композиция графиков B и A\n 0. Выход\n";

cout << "-->";

}

int\*\* InOne(int \*\*A, int \*\*B, int mA, int mB)

{

int mC = mA + mB + 1;

int i, j;

int \*\*C = new int\*[mC]; //Создаём пустой график С

for (int i = 0; i <= mC; i++)

C[i] = new int[3];

if (mB == 0 && mA == 0) // Если числа n и m одновременно равны нулю, то график С - пустой график.

{

cout << "Пустой график! - ";

return C;//Переходим к пункту 17.16.

}

if (mA == 0)//Если число n равно нулю, тогда добавляем кортежи графика В в график С

{

for (i = 1; i <= mB; i++)

{

for (j = 1; j < 3; j++)

{

C[i][j] = B[i][j];

}

}

C[0][0] = mB;

return C;//Переходим к пункту 17.16.

}

if (mB == 0)// Если число m равно нулю, тогда добавляем кортежи графика А в график С.

{

for (i = 1; i <= mA; i++)

{

for (j = 1; j < 3; j++)

{

C[i][j] = A[i][j];

}

}

C[0][0] = mA;

return C;//Переходим к пункту 17.16.

}

for (i = 1; i <= mA; i++)//i = 1 (для графика A).

{

for (j = 1; j < 3; j++)// j = 1 (для графика B).

{

C[i][j] = A[i][j];

}

}

for (int b = 1; b <= mB; b++)

{

for (int a = 1; a <= mA; a++)

{

if (B[b][1] == A[a][1] && B[b][2] == A[a][2])//Если первая компонента i-й пары графика A равна первой компоненте j-й пары графика B

//Вторая компонента i-й пары графика A равна второй компоненте j-й пары графика B, переходим к пункту 17.12.

break;

if (B[b][1] != A[a][1] || B[b][2] != A[a][2]) // Если первая компонента i-й пары равна первой компоненте одной из пар графика D

// Вторая компонента i-й пары равна второй компоненте той же пары графика D, то переходим к пункту 17.12.

{

if (a == (mA))

{

C[i][1] = B[b][1];

C[i][2] = B[b][2];

i++;

}

else continue;

}

}

}

C[0][0] = i - 1;

return C;

}

int\*\* Peresek(int \*\*A, int \*\*B, int mA, int mB)

{

int mD = mA + mB + 1;

int \*\*D = new int\*[mD];//Создаём пустой график D

for (int i = 0; i <= mD; i++)

D[i] = new int[3];

int i = 0;

if (mA == 0 || mB == 0)// Если число n = 0 (m = 0) , тогда график D - пустой график.

{

cout << "Пустой график! - ";

return D;

}

for (int b = 1; b <= mB; b++) //i = 1 (для графика A).

{

for (int a = 1; a <= mA; a++) //j = 1 (для графика B).

{

if (B[b][1] == A[a][1] && B[b][2] == A[a][2])// Если первая компонента i-й пары графика A не равна первой компоненте j-й пары графика B, переходим к пункту 16.9.

{

i++;// Увеличиваем на единицу число i.

D[i][1] = B[b][1]; // 1.1 Добавляем i-ю пару графика A в график D.

D[i][2] = B[b][2];

}

}

}

if (i == 0)

cout << "Пустой график! - ";

D[0][0] = i;

return D;//Выводим на экран график D.

}

int\*\* Pa3HocTb(int \*\*U, int \*\*W, int mU, int mW)

{

int mR = mU + 1;

int \*\*R = new int\*[mR]; // Создаём пустой график R.

for (int i = 0; i <= mR; i++)

R[i] = new int[3];

int r = 0;

if (mU == 0) // 1.1 Если число n равно нулю, тогда график R - пустой график.

{

cout << "Пустой график! - ";

return R;

}

if (mW == 0) // 1.1 Если число m равно нулю, тогда добавляем кортежи графика А в график R.

{

for (int i = 1; i <= mU; i++)

{

R[i][1] = U[i][1];

R[i][2] = U[i][2];

}

R[0][0] = mU;

return R;

}

for (int i = 1; i <= mU; i++)// i = 1 (для графика A).

{

for (int j = 1; j <= mW; j++) // j = 1 (для графика B).

{

if (U[i][1] == W[j][1] && U[i][2] == W[j][2]) // Если первая компонента i-й пары графика A равна первой компоненте j-й пары графика B

// 1.1.1 Если вторая компонента i-й пары графика A равна второй компоненте j-й пары графика B, переходим к пункту 18.10.

break;

if (j == (mW)) // 1.1 Если j < m, переходим к пункту 18.6.

{

r++;

R[r][1] = U[i][1];

R[r][2] = U[i][2];

}

}

}

if (r == 0)

cout << "Пустой график! - ";

R[0][0] = r;

return R;

}

int\*\* SimRas(int \*\*A, int \*\*B, int mA, int mB)

{

int mS = mA \* mB + 2;

int \*\*S = new int\*[mS]; // Создаём пустой график S.

for (int i = 0; i <= mS; i++)

S[i] = new int[3];

int s = 0, o = 0;

if (mA == 0 && mB == 0) // Если числа n и m одновременно меньше единицы, тогда график S - пустой график. Переходим к пункту 20.18.

{

cout << "Пустой график!";

return S;

}

if (mA == 0)// Если число n = 0 , добавляем элементы множества А во множество S.

{

for (int i = 1; i <= mB; i++)

{

S[i][1] = B[i][1];

S[i][2] = B[i][2];

}

S[0][0] = mB;

return S;

}

if (mB == 0)// Если число m = 0 , добавляем элементы множества B во множество S.

{

for (int i = 1; i <= mA; i++)

{

S[i][1] = A[i][1];

S[i][2] = A[i][2];

}

S[0][0] = mA;

return S;

}

for (int i = 1; i <= mA; i++)// i = 1 (для графика A).

{

for (int j = 1; j <= mB; j++)// j = 1 (для графика B).

{

o = 0;

if (A[i][1] == B[j][1] && A[i][2] == B[j][2])// 1.1 Если первая компонента i-й пары графика A равна первой компоненте j-й пары графика B

// 1.1.1 Если вторая компонента i-й пары графика A равна второй компоненте j-й пары графика B, переходим к пункту 20.14.

continue;

for (int l = 1; l <= s; l++)

{

if (S[l][1] == A[i][1] && S[l][2] == A[i][2])// Если первая компонента i-й пары равна первой компоненте одной из пар графика S

// Если вторая компонента i-й пары равна второй компоненте той же пары графика S, то переходим к пункту 20.14.

{

o = 1;

break;

}

}

for (int v = 1; v <= mB; v++)

{

if (A[i][1] == B[v][1] && A[i][2] == B[v][2])// Если первая компонента i-й пары равна первой компоненте одной из пар графика В

// Если вторая компонента i-й пары равна второй компоненте той же пары графика В, то переходим к пункту 20.14.

{

o = 1;

break;

}

}

if (o == 1)

continue;

s++;

S[s][1] = A[i][1];// Добавляем i-ю пару в график S.

S[s][2] = A[i][2];

}

}

for (int i = 1; i <= mB; i++)

{

for (int j = 1; j <= mA; j++)

{

o = 0;

if (B[i][1] == A[j][1] && B[i][2] == A[j][2])

continue;

for (int l = 1; l <= s; l++)

{

if (S[l][1] == B[i][1] && S[l][2] == B[i][2])//Если первая компонента j-й пары равна первой компоненте одной из пар графика S

// Если вторая компонента j-й пары равна второй компоненте той же пары графика S, то переходим к пункту 20.14.

{

o = 1;

break;

}

}

for (int v = 1; v <= mA; v++)

{

if (B[i][1] == A[v][1] && B[i][2] == A[v][2])// Если первая компонента j-й пары равна первой компоненте одной из пар графика A

// Если вторая компонента j-й пары равна второй компоненте той же пары графика A, то переходим к пункту 20.14.

{

o = 1;

break;

}

}

if (o == 1)

continue;

s++;

S[s][1] = B[i][1];// Добавляем j-ю пару в график S.

S[s][2] = B[i][2];

}

}

if (s == 0)

cout << "Пустой график! - ";

S[0][0] = s;

return S;// График S – симметрическая разность графиков A и B.

}

int\*\* Inversion(int \*\*M, int mM)

{

int \*\*I = new int\*[mM];//Создаём пустой график I.

int i = 0;

for (int l = 0; l <= mM; l++)

I[l] = new int[3];

if (mM == 0)// Если число n равно нулю, тогда график I – пустой график.

{

cout << "Пустой график! - ";

return I;//Переходим к пункту 21.10

}

for (i = 1; i <= mM; i++)

{

I[i][1] = M[i][2];// Добавляем вторую компоненту i-й пары графика А на место первой компоненты пары а.

I[i][2] = M[i][1];//Добавляем первую компоненту i-й пары графика A на место второй компоненты пары а.

}

I[0][0] = mM;

return I;//График I — инверсия графика A.

}

int\*\* Exposition(int \*\*A, int \*\*B, int mA, int mB)

{

int mO = mA \* mB + 1;

int \*\*K = new int\*[mO]; // Создаём пустой график К.

int k = 0;

for (int p = 0; p <= mO; p++)

K[p] = new int[3];

if (mA == 0 || mB == 0)// Если число n или m равно 0, тогда график К - пустой график.

{

cout << "Пустой график! - ";

return K;

}

for (int i = 1; i <= mA; i++)// i = 1 (для графика A).

{

for (int j = 1; j <= mB; j++)// j = 1 (для графика B).

{

int o = 0;

if (A[i][2] != B[j][1])// Если вторая компонента i-й пары графика A не равна первой компоненте j-й пары графика B, переходим к пункту 23.11.

continue;

for (int l = 1; l <= k; l++)

if (A[i][1] == K[l][1] && B[j][2] == K[l][2])

{

o = 1;

break;

}

if (o == 1)

continue;

k++;

K[k][1] = A[i][1];// Добавляем первую компоненту i-й пары графика A на место первой компоненты пары a.

K[k][2] = B[j][2];// Добавляем вторую компоненту j-й пары графика B на место второй компоненты пары a.

o = 0;

}

}

if (k == 0)

cout << "Пустой график! - ";

K[0][0] = k;//Добавляем пару a в график K.

return K;

}