Дана матрица B ранга n. В частном случае, если ранг матрицы равен n = 3 -

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{32} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 3 & 1 & 2 \\ 4 & 6 & 5 \\ 7 & 8 & 9 \end{bmatrix}$$

Задача 4.1. Построить программу транспонирования матрицы B^T, введя значения элементов исходной матрицы
Пояснения.
Транспонированная матрица строится из исходной матрицы заменой строк на столбцы
Пример программы (C++):

```
#include <iostream>
using namespace std;
int main()
{
    // Объявляем массив n*m    A - исходная матрица
    //                         B - транспонированная матрица
   const int n = 2;
   const int m = 3;
   int A[n][m], B[m][n];   // Первый индекс - число строк
                           // Второй индекс - число столбцов
     // Создаём исходную матрицу
   for (int i = 1; i <= n; i++)  {
      for (int j = 1; j <= m; j++)  cin >> A[i][j]; }
     // Печатаем исходную матрицу
   cout << "   Матрица A " <<  endl;
   for (int i = 1; i <= n; i++)  {
      for (int j = 1; j <= m; j++)
         {cout.width(2); cout << A[i][j] ;}
         cout << endl;  }
    // Создаём транспонированную матрицу B
   for (int i = 1; i <= n; i++)  {
      for (int j = 1; j <= m; j++)  B[j][i] = A[i][j]; }
     // Печатаем транспонированную матрицу B
   cout  << "   Матрица B "<<  endl ;
   for (int i = 1; i <= m; i++)  {
      for (int j = 1; j <= n; j++)
         {cout.width(2); cout << B[i][j] ;}
         cout << endl;  }
   return 0;
}
```

Результат работы программы –

```
1
2
3
4
5
6
```

Матрица A
1 2 3
4 5 6
Матрица B
1 4
2 5
3 6

Задача 4.2
Построить программу расчета для произвольной матрицы
а) Суммы элементов матрицы B, лежащих на главной диагонали
в) Определителя матрицы B
с) Обратной матрицы B^{-1}

Пояснения.

$$B^{-1} = \frac{1}{|B|} B_+^T$$

|B| - определитель матрицы B. B_+^T – транспонированная матрица алгебраических дополнений.

$$|B| = b_{11} * \begin{vmatrix} b_{22} & b_{23} \\ b_{32} & b_{33} \end{vmatrix} - b_{12} * \begin{vmatrix} b_{21} & b_{23} \\ b_{31} & b_{33} \end{vmatrix} + b_{13} * \begin{vmatrix} b_{21} & b_{22} \\ b_{31} & b_{32} \end{vmatrix}$$

Элементы матрицы миноров $M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{32} & m_{32} & m_{33} \end{bmatrix}$ рассчитываются через определители исходной матрицы после вычеркивания элементов, стоящих на столбцах и строках. Например, $m_{12} = \begin{vmatrix} b_{21} & b_{23} \\ b_{31} & b_{33} \end{vmatrix}$.
Матрица алгебраических дополнений формируется из матрицы M путем изменения знака у каждого второго элемента –

$$B_+ = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{32} & m_{32} & m_{33} \end{bmatrix}$$

Транспонированная матрица B_+^T получается из B_+ путем симметричной перестановки элементов относительно главной диагонали.

Пример программы (Java):

```
package pr1;
public class rr {
      public static void main(String[] args) {

  // Определение элементов массива
            double [ ] [ ] a = { {3, 1, 2 }, {4, 6, 5}, {7, 8, 9} };
            double [ ] [ ] c, p;
          int n = 3;
            c = new double [n][n];
            p = new double [n][n];
```

```
            System.out.println("                    Матрица a ");
          for (int j=0; j<n; j++){
                for (int i=0; i<n; i++)  System.out.print("  "+a[j][i]);
                System.out.println(); }

          System.out.println("                    Определитель  a ");
          double d=a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1])-
                   a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0])+
                   a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]);
          System.out.println(" Определитель "+d);
          System.out.println();
          System.out.println("        Матрица алгебраических дополнений матрицы a
");
          c[0][0] = +(a[1][1]*a[2][2]-a[1][2]*a[2][1]);
          c[0][1] = -(a[1][0]*a[2][2]-a[1][2]*a[2][0]);
          c[0][2] = +(a[1][0]*a[2][1]-a[1][1]*a[2][0]);

          c[1][0] = -(a[0][1]*a[2][2]-a[0][2]*a[2][1]);
          c[1][1] = +(a[0][0]*a[2][2]-a[0][2]*a[2][0]);
          c[1][2] = -(a[0][0]*a[2][1]-a[0][1]*a[2][0]);

          c[2][0] = +(a[0][1]*a[1][2]-a[1][1]*a[0][2]);
          c[2][1] = -(a[0][0]*a[1][2]-a[0][2]*a[1][0]);
          c[2][2] = +(a[0][0]*a[1][1]-a[1][0]*a[0][1]);

          for (int j=0; j<n; j++){
                for (int i=0; i<n; i++)  System.out.print("  "+c[j][i]);
                System.out.println(); }

          System.out.println("       Транспонированная матрица алгебраических
дополнений матрицы a ");
          double dd=0;
          for (int j=0; j<n; j++)
          for (int i=0; i<n; i++)
            if (i>j){ dd = c[j][i]; c[j][i] = c[i][j]; c[i][j] = dd; }

          for (int j=0; j<n; j++){
                for (int i=0; i<n; i++)  System.out.print("  "+c[j][i]);
                System.out.println(); }

          if(Math.abs(d) > 1.0e-10){
            System.out.println("      Обратная матрица к матрице a ");
            for (int j=0; j<n; j++)
            for (int i=0; i<n; i++) c[j][i] = c[j][i]/d;

            for (int j=0; j<n; j++){
                  for (int i=0; i<n; i++)  System.out.print("  "+c[j][i]);
                  System.out.println(); }

            //  Проверка обратной матрицы
            System.out.println();
            System.out.println("      Произведение матрицы a на обратную матрицу к a
");
                for (int j=0; j<n; j++){
                         for (int i=0; i<n; i++) { p[j][i] = 0;
                            for (int k=0; k<n; k++)  p[j][i]+=a[j][k]*c[k][i];
                            System.out.print("  "+p[j][i]); }  System.out.println();
}


          } else  System.out.println("      Обратная матрица не существует ");
```

```
      }
}
```

Результат работы программы -

```
                   Матрица а
  3.0  1.0  2.0
  4.0  6.0  5.0
  7.0  8.0  9.0
                   Определитель  а
 Определитель 21.0

        Матрица алгебраических дополнений матрицы а
  14.0  -1.0  -10.0
  7.0  13.0  -17.0
  -7.0  -7.0  14.0
       Транспонированная матрица алгебраических дополнений матрицы а
  14.0  7.0  -7.0
  -1.0  13.0  -7.0
  -10.0  -17.0  14.0
      Обратная матрица к матрице а
  0.6666666666666666  0.3333333333333333  -0.3333333333333333
  -0.047619047619047616  0.6190476190476191  -0.3333333333333333
  -0.47619047619047616  -0.8095238095238095  0.6666666666666666

      Произведение матрицы а на обратную матрицу к а
  1.0  0.0  0.0
  0.0  1.0  0.0
  -8.881784197001252E-16  0.0  1.0
```