Курс «Основы программирования»

Федорук Елена Владимировна ст. преподаватель каф РК-6 МГТУ им.Н.Э.Баумана

Лекция №11

Матрицы

Матрица — это двумерный массив, то есть массив, имеющий два индекса.

По существу, матрица является массивом одномерных массивов.

Объявление матриц

int a[4][5]; – матрица целого типа из 4 строк и 5 столбцов (индексы меняются первый от 0 до 3, второй от 0 до 4).

float matr[10][20]; – матрица вещественного типа из 10 строк и 20 столбцов

double x[10][10]; — матрица вещественного типа с двойной точностью из 10 строк и 10 столбцов

В памяти матрицы располагаются по строкам. Быстрее изменяется второй индекс.

Инициализация матриц при объявлении

```
short x[3][4] ={{9,6,-56,0},
                {10,3,-4,78},
                {-6,8,45,7}};
int A[4][3]={{12,45,11},
             {67,21,56},
             {90,0,-13},
             {44,-87,-54}};
```

Доступ к элементам матрицы

Пример 1

```
int a[5][4],i,j;
a[0][1]=5.1;          /*прямой доступ*/
i=j=3;
a[i][j]=23;           /*косвенный доступ: значения индексов находятся в переменных*/
```

Ввод матриц

Ввод матриц оуществляется поэлементно, по строкам.

Пример 2

```
for(i=0; i<n; i++)
  for(j=0; j<m; j++)
    scanf("%f",&a[i][j]);
```

Вывод матриц

Вывод матриц осуществляется поэлементно, по строкам или по столбцам, в зависимости от требования программы.

Пример 3

```
for(i=0; i<n; i++)              /* вывод по строкам*/
 {
 for(j=0; j<m; j++)
   printf("%7.2f",a[i][j]);
 printf("\n");
 }
```

Пример 4

```
for(j=0; j<m; j++)
 {
 for(i=0; i<n; i++)             /* вывод по столбцам*/

   printf("%7.2f",a[i][j]);
 printf("\n");
 }
```

Особенности программирования обработки матриц

При обработке матриц используются вложенные циклы. Обработка матриц может производиться как по строкам, так и по столбцам.

Так как матрица расположена в памяти по строкам, второй индекс меняется быстрее. Поэтому при обработке по строкам, внешний цикл индексирует строки, а внутренний столбцы.

Пример 1

```
for(i=0; i<n; i++)
 for(j=0; <m; j++)
  {
  /*обработка элемента a[i][j]*/
  }
```

При необходимости обойти матрицу по столбцам, достаточно изменить последовательность выполнения циклов.

Пример 2

```
for(j=0; j<m; ++)
  for(i=0; i<n; i++)
    {
    /*обработка элемента a[i][j]*/
    }
```

В прим. 1 и прим. 2 **i** – номер элемента в строке, **j** — номер элемента в столбце.

При решении некоторых задач обработки матриц могут быть выделены подзадачи, при программировании которых можно использовать приемы обработки одномерных массивов.

Поэлементная обработка матрицы

Задачи данной группы аналогичны задачам поэлементной обработки одномерных массивов. Реализуются эти задачи использованием вложенных циклов.

Пример 1

Дана матрица вещественного типа `p[3][5]`. Определить максимальный элемент матрицы и его координаты в матрице.

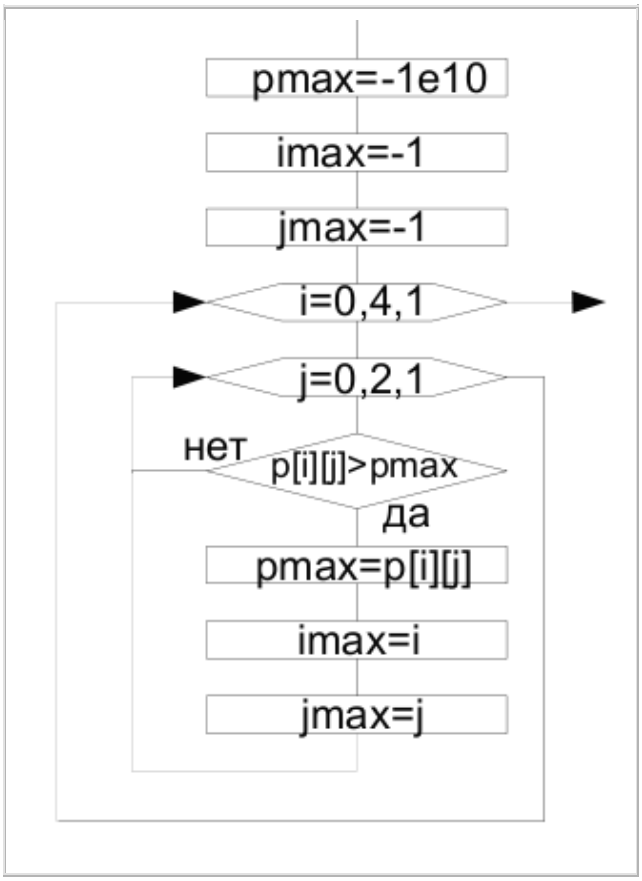

Рис. 1.

```
int max_matr(float p[][5])
{ float pmax;
int i,j,imax,jmax;
pmax=-1e+6;
imax=-1;
jmax=-1;
for(i=0; i<3; i++)
  for(j=0; j<5; j++)
    if (p[i][j]>pmax)
      {pmax=p[i][j];
       imax=i;
       jmax=j;
      }
printf("Max Elem. = %7.2f",pmax);
printf(" imax= %4d",imax+1);
printf(" jmax=%4d\n",jmax+1);
return 0;
}
```

Пример 2

Дана вещественная матрица. Определить номер строки, содержащей самую большую сумму элементов.

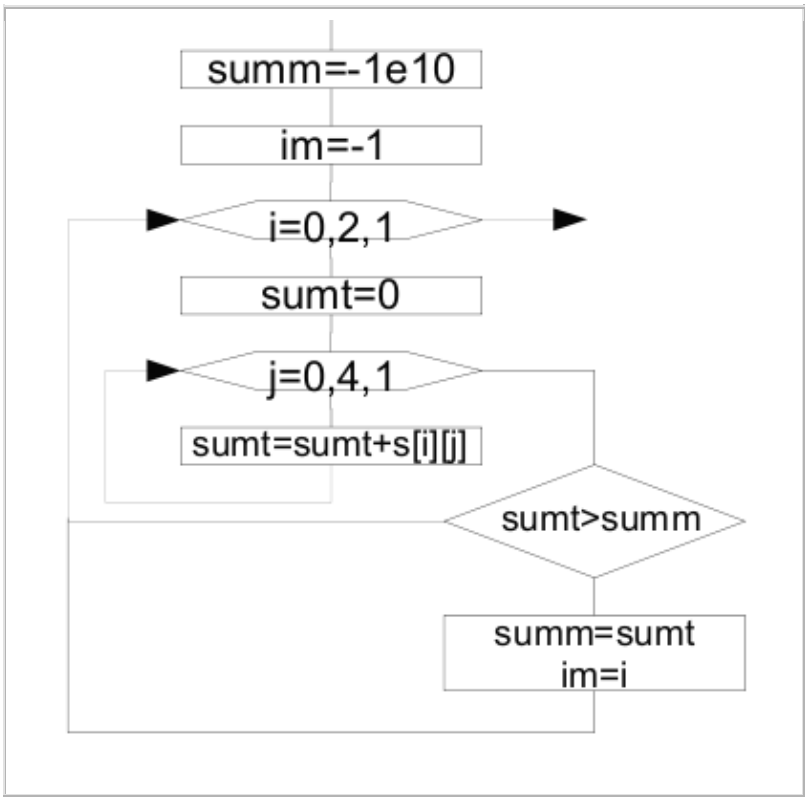

Рис. 2.

```
int maix_sum(float s[][5])
{float summ,sumt;
int i,j,im;
summ=-1e+6;
im=-1;
for(i=0; i<3; i++)
  {sumt=0;
   for(j=0; j<5; j++)
     sumt+=s[i][j];
   if (sumt>summ)
      {summ=sumt;
       im=i;
      }
   }
printf("Max Sum Elem. =");
printf("%7.2f",Summ);
printf(" im= %4d \n",im+1);
return 0;
}
```

Выборочная обработка матрицы

Задачи данной группы аналогичны задачам выборочной обработки одномерных массивов. Реализуются эти задачи с помощью вложенных циклов.

Пример 1

Дана целочисленная матрица. Определить среди четных строк, строку, имеющую наибольшее среднее арифметическое ее элементов.

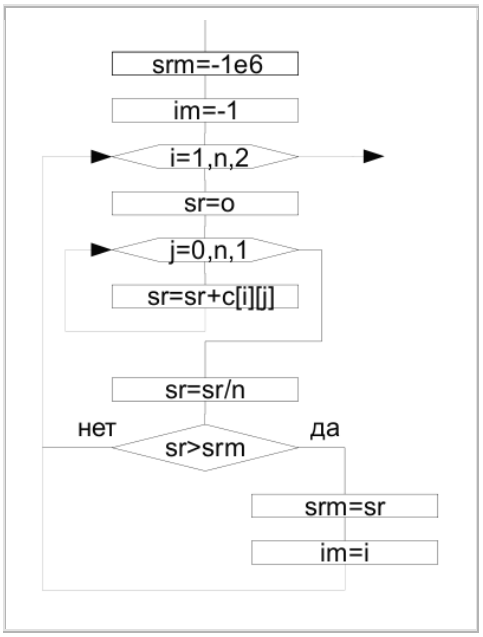

Рис. 1.

```
int matr(int c[][5], int n)
{float srm,sr;
int i,j,im;
srm=-1e+6;
im=-1;
for(i=1; i<n; i=i+2)
  {sr=0;
   for(j=0; j<3; j++)
     sr+=c[i][j];
   sr=sr/n;
   if (sr>srm)
    {srm=sr;
     im=i;
    }
   }
printf("Max SR Sum Elem.= ");
printf(" %7.3f",srm);
printf(" im= %4d \n",im+1);
return 0;
}
```

Переформирование матрицы

Задачи данной группы аналогичны задачам переформирования одномерных массивов. Реализуются эти задачи использованием вложенных циклов.

Пример 1

Дана целочисленная матрица. Отсортировать ее по возрастанию элементов последнего столбца.

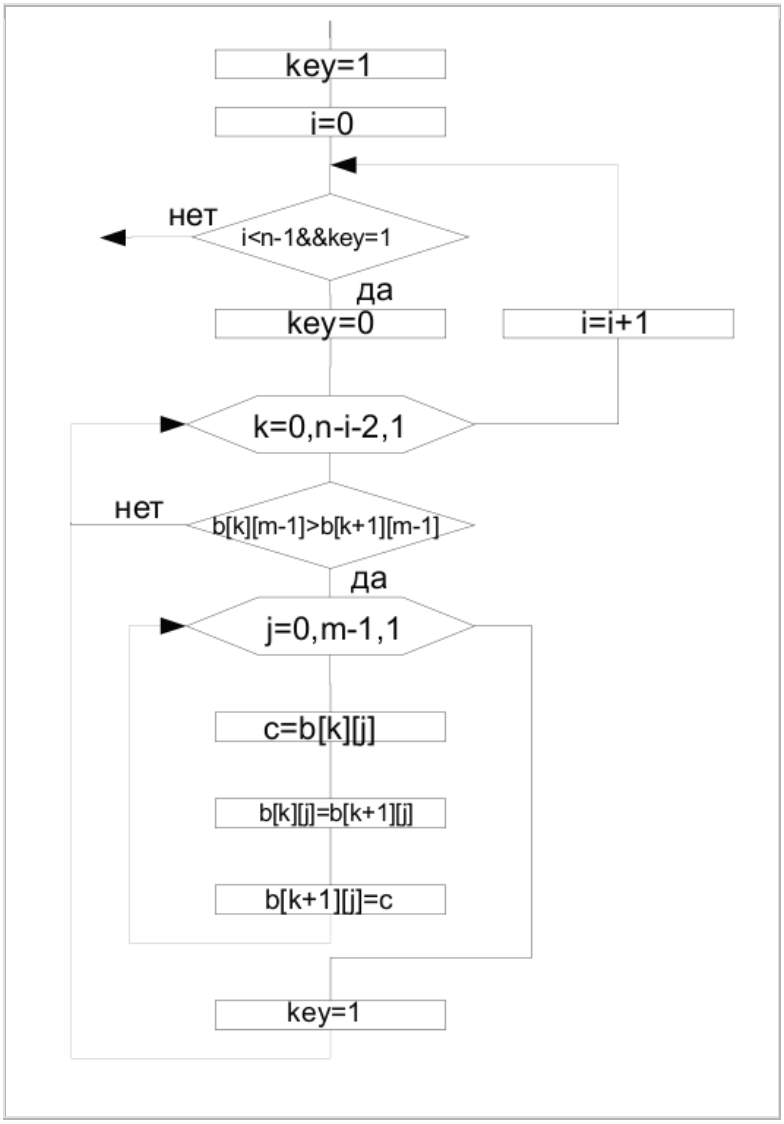

Рис. 1.

```
int matr_sort(int b[][10], int n, int m)    /*функция сортирует матрицу n на m*/
{int i,j,m,k,key,c;
key=1;
i=0;
while((i<n-1) && (key==1))
  {key=0;
   for(k=0; k<n-i-1; k++)
   if(b[k][m-1]>b[k+1][m-1])
     {for(j=0; j<m; j++)
        {c=b[k][j];
         b[k][j]=b[k+1][j];
         b[k+1][j]=c;
```

```
      }
     key=1;
    }
  i=i+1;
 }
}
```

Примечание 1

Переменная **key** используется для того, чтобы отслеживать ситуацию, когда массив уже отсортирован, а полный цикл просмотра матрицы еще не закончился.

Одновременная обработка матриц и подмассивов

Задачи данной группы аналогичны задачам одновременной обработки одномерных массивов и подмассивов. Реализуются эти задачи использованием вложенных циклов.

Пример 1

Дана матрица **a** целого типа. Определить сумму элементов каждой строки и записать ее в новый массив **s**.

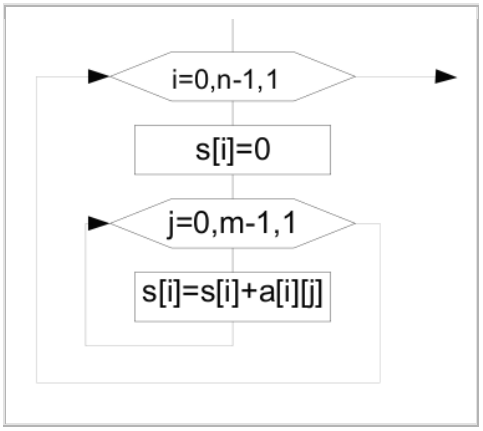

Рис. 1.

```
const int N=5;
int matr_sum(int a[][N], int n, int m, int s[])
{ int i,j;
for(i=0; i<n; i++)
  for (j=0,s[i]=0; j<m; j++)
    s[i]+=a[i][j];
}
```

Обработка матрицы по частям

Кроме задач, по приемам похожих на обработку одномерных массивов, есть большая группа задач, связанных с различными вариантами обхода матриц, и задач обработки разных групп элементов. В таких задачах необходимо исследовать закономерность изменения индексов.

Пример 1

Дана целочисленная матрица **F(10,10)**. Определить сумму отрицательных элементов матрицы и их количество, среди элементов, лежащих выше главной диагонали.

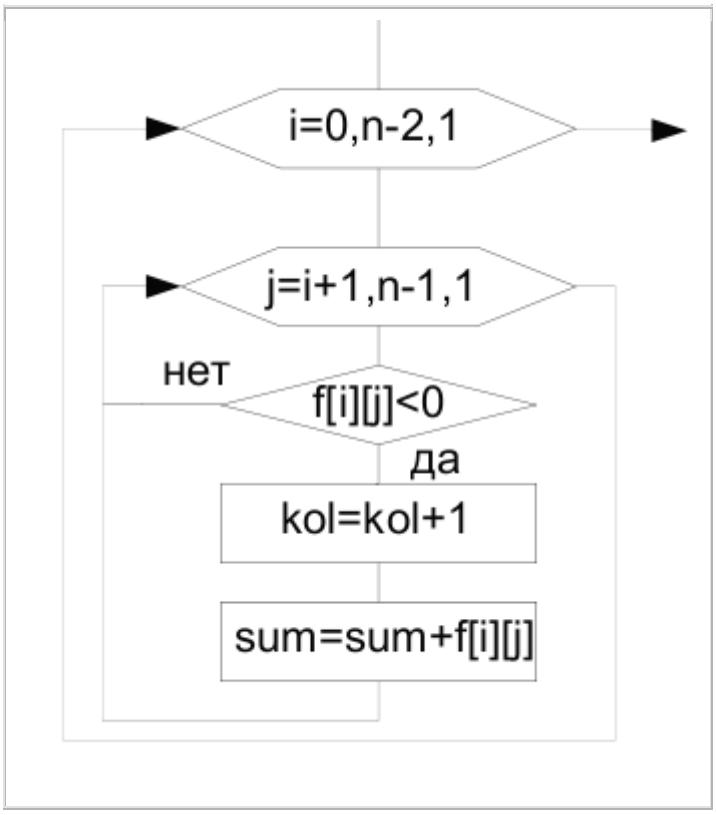

Рис. 1.

```
#define N 10
int diag_matr(int f[][N])
{int i,j,sum,kol;
sum=0;
kol=0;
for(i=0; i<N-1; ++)
   for(j=i+1; j<N;j++)
      if(f[i][j]<0)
        { sum=sum+f[i][j];
          kol=kol+1;
         }
printf("Summa ");
```

```
printf("%7d otricatelnyx",kol);
printf("elementov");
printf(" = %8d\n",sum);
return 0;
}
```