6. МАССИВЫ

6.1. Массив

Массив представляет собой пронумерованную последовательность однотипных величин. Основное свойство массива: все его элементы одновременно присутствуют в оперативной памяти, одинаково доступны и могут обрабатываться в любом порядке (см. разделы 2.9.2 и 2.9.3).

Массивы удобны для выполнения одинаковых действий над несколькими величинами. Элементы массива обозначаются одинаковым именем и отличаются индексами. Элемент массива используется таким же образом, как и другие константы или переменные.

Пример 1. Перестановка символов текста t в обратном порядке:

```
int i, j;
char s, t[100];
    . . .
for (i=0, j=99;   i < j;   i++, j--)
{  s=t[i];  t[i]=t[j];  t[j]=s;  }            /* Обмен t[i]  <-->  t[j]       */
```

Цикл повторяется 50 раз с изменением i от 0 до 49, а j - от 99 до 50.

Если в условии задачи фигурируют переменные с индексами, для них не всегда требуется использовать массив. Массив нужен для входных или выходных данных, если их элементы обрабатываются неоднократно или не в том порядке, в котором вводятся или выводятся. В этих случаях приходится одновременно хранить в памяти все элементы в виде массива.

Если же элементы обрабатываются по одному разу в порядке их ввода или вывода, возможна их последовательная обработка и массив не требуется: достаточно хранить в оперативной памяти один текущий элемент.

Использование массива в таком случае не только усложняет программу и замедляет ее работу, но и без необходимости ограничивает ее возможности, т. к. объем данных в этом случае не сможет превосходить размер массива.

В последующих разделах приведены примеры решения задач с одномерными и двумерными массивами.

6.2. Одномерные массивы

Пример 2. Сортировка чисел. Дано целое n и вещественные x1, x2, ..., xn . Составить программу печати заданных вещественных чисел в порядке возрастания.

Тест:

Введите количество чисел
5
Введите числа
3 1.5 9 4 7
Упорядоченные числа:
1.5 3.0 4.0 7.0 9.0

Разработаем алгоритм нисходящим методом, постепенно детализируя данные и алгоритм с использованием псевдокода - неформальной версии языка С.

Порядок обработки (вывода) данных не совпадает с порядком их ввода. Поэтому для хранения обрабатываемых данных нужен массив. Исходные данные вводятся в массив, там сортируются - переставляются по возрастанию, а затем выводятся. Алгоритм на псевдокоде из 3 шагов имеет вид:

```
int n;          /* Количество входных чисел */
float x[n];
/* Печать входных чисел по возрастанию */
1. Ввод массива x;
2. Сортировка массива x по возрастанию;
3. Вывод массива x;
```

Детализируем шаги алгоритма. Это можно делать в любом порядке. Начнем, например, с ввода и вывода**.**

```
/* 1. Ввод массива x    */
int j;
  . . .
Ввод n;
for (j = 0; j < n; ++j)
   Ввод x[j];
```

```
/* 3. Вывод массива x  */
Вывод заголовка «Упорядоченные числа:»;
for (j = 0; j < n; ++j)
   Вывод x[j];
```

Теперь детализируем шаг сортировки. Существуют десятки методов *сортировки* (упорядочивания). Одним из простейших методов сортировки является *метод обмена* (*пузырька*). При просмотре массива, например, от конца к началу, упорядочиваются пары соседних элементов: элементы, расположенные не по возрастанию, меняются местами.

После просмотра минимальный элемент окажется в начале массива, и второй раз просматриваются уже n – 1 элементов, затем n – 2 и т. д.,

последний раз просматриваются два элемента. Время сортировки пропорционально количеству сравнений

$(n-1) + (n-2) + (n-3) + \ldots + 1 = n*(n-1)/2 \approx n^2/2$.

Описанный метод представлен в алгоритме шага 2. Более быстрые (и более сложные!) методы выполняют сортировку за время, пропорциональное n * log n .

```
/* 2. Сортировка массива x по возрастанию методом обмена */
int k;                  /* Минимальный индекс просмотра        */
  . . .
for (k = 0;   k < n-1;  k++)
   /* Просмотр элементов x[n-1], ... , x[k]                     */
   for (j=n-1;  j > k;  j--)
      /* Сортировка пары элементов x[j-1] и x[j]                */
      if (x[j-1] > x[j])                  /* Порядок нарушен  */
         Обмен:  x[j-1] <--> x[j];
```

Разработанный алгоритм реализован в программе 5.1.

Программа 1

```
/*              Печать входных чисел по возрастанию            */
#include <stdio.h>
#define  NMAX  100      /* Макс-е количество входных чисел     */
int main (void)
{  float x[NMAX];       /* Обрабатываемые числа                */
   int   n;             /* Количество чисел                    */
   int   j;             /* Индекс текущего числа               */
   int   k;             /* Максимальный индекс просмотра       */
   float r;             /* Для обмена                          */

   /* 1. Ввод массива x                                        */
   printf ("\nВведите количество чисел\n");
   scanf ("%d", &n);
   printf ("Введите числа\n");
   for (j = 0; j < n; ++j)
      scanf("%f", &x[j]);

   /* 2. Сортировка массива x по возрастанию методом обмена    */
   for (k = 0;   k < n-1;  k++)
      /* Просмотр элементов x[n-1], ... , x[k]                 */
      for (j=n-1;  j > k;  j--)
         /* Сортировка пары элементов x[j-1] и x[j]            */
         if (x[j-1] > x[j])                  /* Порядок нарушен */
         {  /* Обмен:  x[j-1] <--> x[j];                      */
            r=x[j];  x[j]=x[j-1];  x[j-1]=r;
         }
   /*        3. Вывод массива x        */
```

```
    printf("Упорядоченные числа:\n");
    for (j = 0; j < n; ++j)
        printf ("  %4.1f", x[j]);
    return 0;
}
```

В некоторых языках (например, PL/1) разрешается использование *динамических массивов*, размер которых определяется и может изменяться во время работы программы.

Определенный в алгоритме массив **float** x[n]; является динамическим, поскольку количество его элементов n - величина переменная, и его значение определяется (вводится) во время работы программы.

В языках C и Pascal динамические массивы запрещены, в явном виде можно использовать только *статические массивы*, количество элементов в которых задается до начала работы программы, т. е. является константой. Если количество элементов заранее неизвестно, приходится определять массив с запасом в расчете на максимальный размер.

Можно также реализовать динамический массив неявно, с помощью динамического распределения памяти. В определении массива не указывается количество элементов - квадратные скобки остаются пустыми (в многомерном массиве это можно делать только для первого индекса). Для такого массива выделение места в памяти транслятор не производит, его обеспечивает сам программист с помощью функций управления памятью.

Поэтому в программе 2 количество входных чисел ограничено константой NMAX, которая определяет размер массива:

```
#define  NMAX  100   /* Максимальное количество входных чисел */
    . . .
    float x[NMAX];      /* Обрабатываемые числа                    */
```

Массив x содержит NMAX элементов, из которых в программе используются первые n элементов: от x[0] до x[n-1], (n ≤ NMAX). Реальный вариант программы должен был бы иметь защиту от ошибок: содержать проверку, что введенное n не превышает NMAX.

Применение символической константы NMAX облегчает при необходимости изменение размера массива.

Пример 3. Самое длинное слово текста. Входной текст состоит из слов, разделенных пробелами и/или символами "новая строка". Составить программу определения самого длинного слова.

Тест:

```
Я снова  тут,
Я собран весь <Ctrl-z> <Enter>              <Ctrl-z> - конец файла
Самое длинное слово: собран
```

Алгоритм символьной обработки разрабатывается, исходя из структуры читаемого им текста, которую удобно описать в виде синтаксических правил. Разным способам такого описания соответствуют разные варианты алгоритма. Рассмотрим два крайних подхода для данной задачи (возможны и другие решения).

Решение 1. Входной текст рассматривается как последовательность слов. Слово - последовательность символов (отличающихся от пробела, "новой строки" и признака конца файла EOF), которой может предшествовать последовательность разделителей (пробелов или "новых строк").

текст ::= [слово]...EOF
слово ::= [разделитель]... [символ-слова]...
разделитель ::= пробел | новая-строка

Каждому знаку повторения "..." синтаксического правила, описывающего структуру текста, в алгоритме чтения и анализа текста соответствует цикл, знаку "|" (или) – ветвление. Таким образом структура алгоритма повторяет структуру читаемого текста.

Соответствующий алгоритм содержит цикл чтения слов, который включает цикл пропуска пробелов и "новых строк" и цикл чтения символов слова.

Обозначим:
sl - текущее слово,
dsl - длина текущего слова,
maxsl - максимальное слово,
dmaxsl - длина максимального слова.

Алгоритм на псевдокоде:

```
dmaxsl = 0;
while (не конец файла)
{   Пропуск пробелов и "новых строк";
    Чтение текущего слова sl;
    if (dsl > dmaxsl)   Копирование sl в maxsl;
        dmaxsl = dsl;
}
if (dmaxsl > 0)
    Вывод maxsl;
else
    Вывод "В тексте нет слов";
```

В программе 2 текущий символ sim описывается как **int**, чтобы его можно было сравнивать с признаком (символом) конца файла EOF.

Программа 2

```
/*   Определение самого длинного слова входного текста     */
/*      Решение 1:                                          */
/*  текст       ::= [слово]...EOF                           */
/*  слово       ::= [разделитель]... [символ-слова]...      */
/*  разделитель ::= пробел | новая-строка                   */
/*  символ-слова - любой символ, кроме разделителей         */
#include <stdio.h>
#define  DSLMAX  20          /* Максимальная длина слова    */
void main (void)
{  char sl [DSLMAX];          /* Текущее слово               */
   int     dsl;               /* Длина текущего слова        */
   char  maxsl [DSLMAX];      /* Максимальное слово          */
   int     dmaxsl;            /* Длина максимального слова   */
   int     sim;               /*  Текущий символ             */
   int     i;                 /*Текущий индекс копирования   */
   dmaxsl = 0;
   while ((sim = getchar()) != EOF) {
      /* Пропуск пробелов и "новых строк" до слова          */
      while  (sim==' ' || sim=='\n')   sim=getchar();
      /* Чтение слова                                       */
      dsl = 0;
      while (sim!=' ' && sim!='\n' && sim!=EOF)
      {  sl [dsl++] = sim;
         sim = getchar();
      }
      sl [dsl] = '\0';            /* Признак конца слова */
      if (dsl > dmaxsl)
      {  /* Копирование слова */
         for (i = 0; maxsl [i] = sl [i]; i++);
         dmaxsl = dsl;
      }
   }
   if (dmaxsl > 0)
      printf ("\nСамое длинное слово: %s\n", maxsl);
   else
      printf ("\nВ тексте нет слов\n");
}
```

Текущее слово и максимальное слово хранятся в памяти в виде символьного массива с признаком конца. Признаком конца слова служит нулевой байт '\0', т. к. на этот признак рассчитан формат %s для вывода строки функции printf. При другом способе хранения слова (с другим признаком конца или без признака конца с использованием длины слова) его вывод пришлось бы производить в цикле по одному символу.

Решение 2. Здесь входной текст рассматривается как последовательность символов. Символы делятся на два вида: разделители слов (пробел, "новая строка" или признак конца файла EOF) и остальные символы, составляющие слова.

текст ::= символ...EOF
символ ::= разделитель | символ-слова
разделитель ::= пробел | новая-строка

символ-слова - любой символ, кроме разделителей

В соответствии со структурой читаемого текста программа 4 содержит один цикл чтения символов. Так как текст содержит хотя бы один символ (признак конца файла EOF), используется цикл с постусловием.

Внутри цикла - две ветви, соответствующие двум видам символов. Используются такие же данные, как в решении 1, изменяется только структура алгоритма чтения текста. Отсюда получаются алгоритм и программа 3.

Алгоритм на псевдокоде:

```
dmaxsl = 0; dsl = 0;
do
 { if (текущий символ - не разделитель)
        Записать текущий символ в слово sl;
   else /* разделитель - конец слова */
   {   if (dsl > dmaxsl)
       {   Копирование sl в maxsl;
           dmaxsl = dsl;
       }
       dsl = 0;
   }
} while (не конец файла);
if (dmaxsl > 0)  Вывод maxsl;
else   Вывод "В тексте нет слов";
```

Программа 3

```
/*    Определение самого длинного слова входного текста     */
/*       Решение 2:                                         */
/*  текст       ::= символ...EOF                            */
/*  символ      ::= разделитель | символ-слова              */
/*  разделитель ::= пробел | новая-строка                   */
/*  символ-слова - любой символ, кроме разделителей         */
```

```
#include <stdio.h>
#define  DSLMAX  20     /* Максимальная длина слова       */
void main (void)
{  char sl [DSLMAX];    /* Текущее слово                  */
   int     dsl;         /*  Длина текущего слова          */
   char  maxsl [DSLMAX]; /* Максимальное слово            */
   int     dmaxsl;      /*  Длина максимального слова     */
   int     sim;         /*  Текущий символ                */
   int     i;           /*  Текущий индекс копирования    */

   dmaxsl = 0; dsl = 0;
   /* Чтение входного текста */
   do /* Чтение и обработка текущего символа             */
  {   if ((sim=getchar()) != ' ' && sim!='\n' && sim!=EOF)
         sl [dsl++] = sim;
      else                          /* Конец слова       */
      {  sl [dsl] = '\0';           /* Признак конца слова */
         if (dsl > dmaxsl)
         { /* Копирование текущего слова в максимальное */
            for (i = 0; maxsl [i] = sl [i]; i++);
            dmaxsl = dsl;
         }
         dsl = 0;           /* Для чтения следующего слова */
   } while (sim != EOF);
   if (dmaxsl > 0)
      printf ("\nСамое длинное слово: %s\n", maxsl);
   else
      printf ("\nВ тексте нет слов\n");
}
```

6.3. Двумерные массивы

Пример 4. Целочисленная квадратная матрица размера n*n (n≤50) вводится по строкам. Составить программу определения номера столбца, имеющего максимальную сумму элементов, расположенных выше главной диагонали.

Решение.

Входная матрица:
(точками отмечены элементы, не участвующие в задаче)
n=5

	0	1	2	3	4
0	.	6	1	3	-2
1	.	.	4	-1	3

Тест

Результат:
Номер столбца с максимальной суммой равен 3

2	.	.	.	5	1
3	.	.	.	.	2
4	.	.	.	.	.

Программа 4. Обработка матрицы

```
/*Поиск столбца с наибольшей суммой выше главной диагонали */
#include <stdio.h>
#define  NMAX  50          /* Максимальный размер матрицы      */
void main (void)
{  int m[NMAX][NMAX];  /* Матрица                              */
   int  n;                  /* Количество строк и столбцов      */
   int  i;                  /* Индекс текущей строки            */
   int  j;                  /* Индекс текущего столбца          */
   int  jmax;               /* Индекс столбца с максим. суммой  */
   int s, smax;             /* Сумма текущего столбца и максим. */

   /* 1. Ввод матрицы m по строкам                                */
   scanf ("%d", &n);
   for (i=0; i<n; ++i)
       /* Ввод i-й строки                                       */
       for (j=0; j<n; ++j)   scanf("%d", &m[i][j]);
   /* 2.Поиск столбца с максим.суммой выше главной диагонали */
   jmax = 1;   smax = m[0][1];
   for (j=2; j<n; j++)            /* Перебор столбцов          */
   {  /* Вычисление суммы j-го столбца                         */
      for (s=0, i=0; i<j; i++)
        s = s + m[i][j];
      if (s > smax)   { smax=s;   jmax=j; }
   }
   printf ("\nНомер столбца с максим-й суммой = %d", jmax);
}
```

Пример 5. Составить программу для решения следующей задачи. Заполнить квадратную матрицу A = $\{a_{i,j}\}$ (i, j = 0, ..., n-1) значениями, вычисленными по формуле $a_{i,,j}$ = i * j % 5 + i - j. Из матрицы получить вектор X= $\{x_i\}$ (i = 1, ..., n). Элемент x_i вычислять как скалярное произведение i-ой строки матрицы на столбец, содержащий первый по порядку максимальный элемент i-ой строки. Полученный вектор упорядочить по возрастанию методом последовательного нахождения минимума.

Решение приведено в программе 5.

Программа 5. Обработка матрицы и вектора

```
#include <stdio.h>
#define N 10        /*максимальный размер матрицы              */
```

```
main()
{  int n;              /* размер  матрицы                          */
   float a[N][N];      /* матрица                                  */
   float x[N];         /* вектор                                   */
   int i,j,k;          /* текущие индексы строк и столбцов         */
   float m;
   /*          Ввод матрицы                                       */
   printf ("\nВведите размер массива\n");
   scanf("%d",&n);
   for (i=0; i< n; i++)
     for (j=0; j< n; j++)
       a[i][j] = i * j % 5 + i - j;
   /*          Печать  матрицы                                    */
   printf ("\nПолученная матрица\n");
   for (i=0; i< n; i++)
   {  putchar('\n');
      for (j=0; j< n; j++)    printf("%3.0f ",a[i][j]);
   }
   putchar('\n');
   /*          Получение вектора из матрицы                       */
   /* Определение номера столбца, содержащего                     */
   /* первый по порядку максимальный элемент                          */
   for (i=0; i< n; i++)
   {  k=0;
      for (j=1; j< n; j++)
         if (a[i][j] > a[i][k])  k = j;
      /* вычисление скалярного  произведения                      */
      /*   i-й строки на k-й столбец                              */
      x[i] = 0;
      for (j=0; j<n; j++)
         x[i] = x[i] + a[i][j] * a[j][k];
   }
   /*   Печать  вектора                                           */
   printf ("\nПолученный из матрицы вектор\n");
   for (i=0; i< n; i++)
      printf ("%3.0f ", x[i]);
   putchar('\n');
   /*   Упорядочивание вектора                                    */
  for (i=0; i<n; i++)
  {  j = i;   m = x[i];
     for (k=i+1; k<n; k++)
        if (x[k] < m) { j = k;   m = x[k]; }
     x[j]=x[i]; x[i]=m;
  }
  /* Печать  упорядоченного вектора                               */
```

```
  printf ("\nУпорядоченный вектор\n");
  for (i=0; i< n; i++)
    printf("%3.0f ",x[i]);
  putchar('\n');
}
```

Результаты работы программы

```
Введите размер массива
  6
Полученная матрица

0    -1    -2    -3    -4    -5
1     1     1     1     1    -4
2     3     4     0     1    -3
3     5     2     4     1    -2
4     7     5     3     1    -1
5     4     3     2     1     0

Полученный из матрицы вектор

-55    -10     11    27    36    20

Упорядоченный вектор

-55    -10     11    20    27    36
```

Упражнения и задачи

6.1. Составить определение следующих данных.

а) fam - фамилия (не более 15 символов);

б) dni - количество дней в каждом месяце 2000 г.;

в) pl - площади каждой из 144 квартир дома (с точностью до 0.1 $м^2$);

г) rez - оценки 25 студентов по каждому из 4 экзаменов;

д) фамилия (fam), пол (pol) и год рождения (god) каждого из 100 человек

6.2. Составить фрагмент программы заполнения массива A из N целых чисел, где элемент с номером K вычисляется по формуле:

а) 2*к

б) $2^k / k!$

в) $2^k / k$

г) 2^k

д) (к+1) / к

е) к / (к+1)

ж) A[k] - k-е в порядке возрастания простое число (простым называют целое число больше 1, которое без остатка делится только на себя и на 1, например: 2, 53, 19, 97).

6.3. Дано целое N>0 и массив A из N целых чисел. Составить фрагменты программы для решения следующих задач.

а) найти сумму положительных элементов массива;

б) увеличить каждый элемент массива в два раза;

в) каждый второй элемент возвести в квадрат;

г) найти сумму элементов с нечетными индексами;

д) найти сумму элементов с нечетными значениями;

е) найти количество элементов массива, значения которых равны индексу элемента.

ж) найти максимальный элемент и его порядковый номер.

з) переставить местами наибольший и наименьший элементы массива.

и) заменить каждый элемент массива на сумму предыдущего и последующего элементов, если такие существуют.

6.4. Дано целое $N \geq 0$, массив из N целых чисел и целое z. Составить фрагмент программы нахождения минимального индекса такого элемента массива, который равен z. Если искомый элемент отсутствует в массиве, результатом считать -1.

6.5. Дано натуральное число N>0 и последовательность из N целых чисел. Составить программы для решения следующих задач.

а) Определить, сколько чисел отличаются от последнего числа.

в) Распечатать числа в обратном порядке по 6 чисел в строке.

6.6. Дана последовательность из N различных действительных чисел (N>0). Найти сумму чисел, расположенных между максимальным и минимальным числом.

6.7. Дан массив A из M действительных чисел. Известно, что A[0]>0 и что среди остальных элементов есть хотя бы одно отрицательное число. Составить фрагменты программ решения следующих задач.

а) Найти номер элемента, предшествующего первому отрицательному элементу.

б) Найти количество элементов предшествующих первому отрицательному элементу.

6.8. Заданы два упорядоченных по возрастанию массива действительных чисел X[n] и Y[m]. Составить фрагмент программы, объединяющий их в один упорядоченный по возрастанию массив Z[n+m].

6.9. Заданы две упорядоченные по возрастанию последовательности: из n чисел и из m чисел. Составить программу, объединяющую их в одну упорядоченную по возрастанию последовательность из n+m чисел.

а) В двух массивах хранить входные последовательности; выходную последовательность выводить, не записывая в массив.

б) В одном массиве хранить первую из входных последовательностей, в другом - выходную последовательность.

6.10. Дано натуральное N>0 и массив вещественных чисел A[N]. Составить фрагменты программ решения следующих задач.

а) Найти длину самой длинной последовательности подряд идущих элементов массива, равных нулю.

б) Определить, выполняется ли условие, что в массиве нет нулевых элементов, и при этом положительные элементы чередуются с отрицательными элементами.

6.11. Дан текст, длиной не более 80 символов. Составить фрагмент программы определения, симметричен ли он.

6.12. Дан текст произвольной длины, продолжающийся до конца файла. Составить программы для решения следующих задач:

а) Определить, сколько различных литер (символов) входит в текст.

б) Определить, сколько раз каждая литера входит в текст.

в) Определить символ, встречающийся в тексте с максимальной частотой.

г) Напечатать в алфавитном порядке все строчные латинские буквы, которые входят в текст по одному разу.

д) Текст содержит выражение со скобками. Напечатать выражение, заключенное в самые внутренние скобки. Его длина не более 100 символов.

е) Текст состоит из слов, разделенных пробелами и/или переходами на следующую строку. Длина слова не более 20 символов. Найти самое короткое слово.

6.13. Дано целое $N \geq 0$, массив из N вещественных чисел и вещественное z. Составить фрагмент программы перестановки элементов массива так, чтобы

а) от начала массива до некоторого элемента размещались числа, меньшие или равные z, а затем числа, большие или равные z;

б) сначала размещались числа, меньшие чем z, затем равные z, а за ними числа, превышающие z.

6.14. *Скользящее среднее.* Дано целое число K ($0 < K \leq 100$) и последовательность вещественных чисел $X_1, X_2, \ldots, X_N$, продолжающаяся до конца файла (N заранее не известно). Составить программу для получения последовательности чисел $S_1, S_2, \ldots, S_N$, где S_j равно среднему арифметическому значению чисел $X_1, X_2, \ldots, X_j$ при $j \leq K$ и равно среднему арифметическому значению K последних введенных чисел $X_{j-K+1}, X_{j-K+2}, \ldots, X_{j-1}, X_j$ при $j > K$. *Указание*. Для получения очередного среднего значения достаточно помнить текущее входное число, не более K предыдущих введенных чисел (усредняемые числа), их количество и сумму.

6.15. Дано целое $N \geq 0$ и массив из N вещественных (или целых) чисел. Составить фрагмент программы для сортировки, т. е. упорядочивания

массива по неубыванию (чтобы каждый элемент был не меньше предыдущего элемента) следующими способами.

а) *Сортировка выбором*. Выбрать в массиве минимальный элемент и обменять его местами с первым элементом, минимальный из оставшихся элементов обменять со вторым и т. д.

б) *Сортировка обменом.* Просмотреть массив от начала к концу и упорядочить каждую пару соседних элементов, обменивая их местами, если они расположены по убыванию. Подобные просмотры повторять, пока все такие пары не будут отсортированы.

в) *Сортировка вставкой.* Каждый элемент, начиная со второго, вставлять в нужное место среди предыдущих элементов, увеличивая таким образом на единицу длину уже упорядоченной последовательности, пока она не охватит весь массив.

г) *Сортировка подсчетом индекса.* Для каждого элемента подсчитать количество меньших или равных ему элементов, которое и определит его индекс в упорядоченном массиве.

д) *Сортировка подсчетом количества значений.* Подсчитать в массиве количество экземпляров каждого возможного значения (все значения могут быть целыми числами в диапазоне от A до B). Заполнить массив заново в требуемом порядке нужными значениями в подсчитанных количествах.

6.16. Даны целые M и N (не более 250) и матрица x[M][N] из нулей и единиц. Составить фрагменты программы для решения следующих задач.

а) В матрице имеется несколько прямоугольников (подматриц), заполненных единицами. Каждая единица принадлежит одному из прямоугольников. Прямоугольники не пересекаются. Единицы соседних прямоугольников могут касаться друг друга только по диагонали. Подсчитать количество прямоугольников.

б) Подсчитать в матрице количество «пятен». Пятно составлено из единиц, соприкасающихся по горизонтали, вертикали или диагонали.

в) Найти индексы строки и столбца «левого верхнего угла» наибольшего квадрата, заполненного нулями.

г) Найти индексы строки и столбца «левого верхнего угла» наибольшего по площади прямоугольника, заполненного единицами.

6.17. Даны целые M и N (не более 250) и целочисленная матрица x[M][N], вводимая по строкам. Составить программы решения следующих задач.

а) Получить матрицу y[M][N], каждый элемент y[K][L] которой равен сумме всех элементов x[i][j] матрицы x, для которых $i \le K$ и $j \le L$.

б) Найти индексы строки и столбца «левого верхнего угла» квадрата размером K×K с наибольшей суммой элементов ($0 < K \le M$, $K \le N$).

Указание. Использовать матрицу, получаемую как в задаче 5.16 а.

в) Разделить матрицу на четыре непустых сектора двумя ступенчатыми линиями со ступеньками шириной в одно число, направленными на "юго-

восток" и "юго-запад", чтобы минимизировать разность максимальной и минимальной суммы чисел этих секторов. Получить номер строки и номер столбца "южного" углового элемента (он должен быть единственным) "северного" сектора и минимальную разность в сумме чисел секторов. Если результат не однозначен, то выдать ответ с минимальным номером строки, а в этой строке - с минимальным номером столбца.

Пример

Входной файл

```
6 4
1 0 0 1
1 2 2 3
4 3 3 1
7 2 0 8
1 3 1 11
1 2 15 1
```

Выходной файл

```
4 3 2
```