ЛАБОРАТОРНАЯ РАБОТА
УКАЗАТЕЛИ. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКИХ ДВУМЕРНЫХ МАССИВОВ

5.1. Объявление указателя

Для всех переменных выделяются участки памяти размером, соответствующим типу переменной. Программист имеет возможность работать непосредственно с адресами, для чего определен соответствующий тип данных – *указатель*. Указатель имеет следующий формат:

*тип *имя указателя*

Например:

int *a; **double** *b, *d; **char** *c;

Знак «звездочка» относится к имени указателя. Значение указателя соответствует первому байту участка памяти, на который он ссылается. На один и тот же участок памяти может ссылаться любое число указателей.

В языке Си существует три вида указателей:

1. Указатель на объект известного типа. Содержит адрес объекта определенного типа.

2. Указатель типа **void**. Применяется, если тип объекта заранее не определен.

3. Указатель на функцию.

5.2. Операции над указателями

Над указателями можно провести две унитарные операции:

1. & (**взять адрес**). Указатель получает адрес переменной. Данная операция применима к переменным, под которые выделен соответствующий участок памяти.

2. * (**операция разадресации**). Предназначена для доступа к величине, расположенной по данному адресу.

Над указателями можно выполнять арифметические операции сложения, инкремента, вычитания, декремента и операции сравнения. При выполнении арифметических операций с указателями автоматически учитывается размер данных, на которые он указывает.

Указатели, как правило, используются при работе с динамической памятью (*heap*, или «куча»). Для работы с динамической памятью в языке Си определены следующие функции: **malloc**, **calloc**, **realloc** и **free**.

В языке C++ для выделения и освобождения памяти определены операции **new** и **delete** соответственно. Используют две формы операций:

1. *Тип *указатель* = **new** *тип* (*значение*) – выделение участка памяти в соответствии с указанным типом и занесение туда указанного значения.

delete *указатель* – освобождение выделенной памяти.

2. *Тип *указатель* = **new** *тип*[**n**] – выделение участка памяти размером *n* блоков указанного типа.

delete [ ]*указатель* – освобождение выделенной памяти.

5.3. Создание двумерного динамического массива

Имя любого массива рассматривается компилятором как указатель на нулевой элемент массива. Так как имя двумерного динамического массива является указателем на указатель, то сначала выделятся память под указатели, а затем под соответствующие этим указателям строки. Освобождение выделенной памяти происходит в обратном порядке:

```
float **matrix = new float*[n];
for (int i = 0; i < n; ++i)
{
   matrix[i] = new float[m];
}
```

```
for (int i = 0; i < n; ++i)
{
   delete[] matrix[i];
}
delete[] matrix;
matrix = nullptr;
```

5.4. Пример выполнения работы

Условие 1. Найти минимальный и максимальный элементы матрицы и их координаты.

```
float min = FLT_MAX, max = FLT_MIN;
int iMin = 0, jMin = 0, iMax = 0, jMax = 0;
for (int i = 0; i < n; ++i)
{
   for (int j = 0; j < m; ++j)
   {
      if (matrix[i][j] < min)
      {
         min = matrix[i][j];
         iMin = i;
         jMin = j;
      }
      if (matrix[i][j] > max)
      {
         max = matrix[i][j];
         iMax = i;
         jMax = j;
      }
   }
}
```

Условие 2. Упорядочить строки матрицы по неубыванию их максимальных элементов.

```
for (int i = 0; i < n; ++i)
{
   max[i] = matrix[i][0];
   for (int j = 1; j < m; ++j)
   {
      if (matrix[i][j] > max[i])
      {
         max[i] = matrix[i][j];
      }
   }
}
```

```
for (int i = 0; i < n - 1; ++i)
{
   for (int j = 0; j < n - 1 - i; ++j)
   {
      if (max[j] > max[j + 1])
      {
         float temp = max[j];
         max[j] = max[j + 1];
         max[j + 1] = temp;

         for (int k = 0; k < m; ++k)
         {
            float temp = matrix[j][k];
            matrix[j][k] = matrix[j + 1][k];
            matrix[j + 1][k] = temp;
         }
      }
   }
}
```

5.5. Индивидуальные задания

В работе память для массива должна выделяться динамически. На экран выводить исходные данные и результат.

1. Задана матрица размером *N*x*M*. Получить массив *B*, присвоив его *k*-му элементу значение 0, если все элементы *k*-го столбца матрицы нулевые, и значение 1 – в противном случае.

2. Задана матрица размером *N*x*M*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если элементы *k*-й строки матрицы упорядочены по убыванию, и значение 0 – в противном случае.

3. Задана матрица размером *N*x*M*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если *k*-я строка матрицы симметрична, и значение 0 – в противном случае.

4. Задана матрица размером *N*x*M*. Определить количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером *N*x*M*. Определить количество «особых» элементов матрицы, считая элемент «особым», если в строке слева от него находятся элементы, меньшие его, а справа – большие.

6. Задана символьная матрица размером *N*x*M*. Определить количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

7. Дана матрица размером *N*x*M*. Упорядочить ее строки по возрастанию их первых элементов.

8. Дана матрица размером *N*x*M*. Упорядочить ее строки по возрастанию суммы их элементов.

9. Дана матрица размером *N*x*M*. Упорядочить ее столбцы по возрастанию их наименьших элементов.

10. Определить, является ли заданная квадратная матрица *n*-го порядка симметричной относительно побочной диагонали.

11. Для матрицы размером NxM вывести на экран все седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или наоборот.

12. В матрице размером NxM переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.

13. В матрице размером NxM найти максимальный среди элементов, лежащих ниже побочной диагонали, и минимальный среди элементов, лежащих выше главной диагонали.

14. В матрице размером NxM поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением.

15. Из матрицы размером NxM получить матрицу размером (N–1) x (M–1) путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.