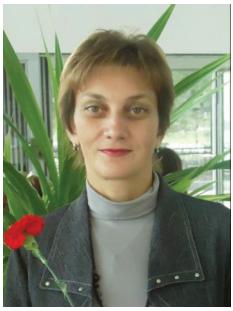

Ларина Элла Семеновна
Учитель информатики
высшей категории
МОУ «Лицей №2» г. Волгограда.

Кратчайшим путём от треугольника Паскаля до бинома Ньютона

В программировании пользоваться типовыми алгоритмами обработки одномерных и двумерных массивов можно как кирпичиками при построении здания – кирпичики стандартные, архитектурный же проект постройки может воплотить любой, даже невероятно изощрённый запрос заказчика. Так же, как и программная реализация решения любой запутанной задачки строится на типовых блоках (кирпичиках-алгоритмах)!

Изучая программирование в школе, каждый старшеклассник сталкивается с типовыми алгоритмами. Какими же скучными кажутся многим эти однообразные алгоритмы поиска суммы элементов массива, инвертирования, вставки, удаления и многие другие! Ну, взгляду зацепиться не за что, читая заунывные формулировки и сухие строки условий задач!

Казалось бы: где жизнь, а где – типовые алгоритмы; где яркость красок, полёт фантазии, а где – скука и однообразие монотонной работы по разбору типовых программных строк. Несовместимые вещи, между ними – пропасть!

А хотите – докажу, что нет? Что и жизнь с её яркими моментами творчества, и сухая монотонность отработки типовых программ может быть совмещена в единой точке под названием «Красивая задача»! И вот, когда дух захватывает от желания разгадать облюбованную ещё древними учёными замысловатую головоломку, когда начинаешь понимать, что она тебе «по зубам», тогда верёвочка решения, сплетённая из типовых алгоритмов, кажется произведением искусства!

Много «Красивых задач» придумали для нас математики прошлого. Программисты настоящего в решении этих задач применяют знания типовых алгоритмов. Какие же типовые алгоритмы вы изучаете в школе при прохождении, скажем, темы «Двумерные массивы»? Вероятно такие:

- **Обработка всего массива** (заполнение, вывод элементов массива, нахождение суммы, произведения, максимального/минимального элемента среди всех элементов и др.).
- **Обработка каждой строки/столбца массива** (нахождение суммы, произведения, максимального/минимального элемента в каждой строке/каждом столбце двумерного массива и др.).
- **Обработка квадратного массива относительно его диагоналей** (нахождение суммы, произведения, максимального/минимального элемента на/выше/ниже главной/побочной диагонали квадратного массива и др.).

Вооружимся же и мы этими типовыми алгоритмами.

А что за задачи мы будем решать, подскажет заголовок этой статьи. Хотя, лучше бы он прозвучал так: «Арифметический квадратный кратчайший путь от треугольника Паскаля до бинома Ньютона». Вот тогда уж точно все разбираемые в статье задачки в нём поместились бы: и «Арифметический квадрат», и «Треугольник Паскаля», и «Бином Ньютона», и «Нахождение кратчайшего пути в двумерном массиве».

Но, вместив в себя названия классических задач, заголовок ничего бы не сказал о методе их решения! В нём был бы скрыт смысл статьи, прозвучи он так: «Кратчайший путь, треугольник Паскаля, бином Ньютона ЧЕРЕЗ арифметический квадрат». Вот и стало всё на свои места! Мы будем решать классические математические задачи, основываясь на методе решения задачи «Арифметический квадрат»! Что ж, приступим.

“Арифметический квадрат“. Дан двумерный массив, размерностью N × M. Необходимо заполнить его таким образом: в каждый элемент массива поместить число, равное количеству путей, ведущих в этот элемент из элемента (1,1). Двигаться по массиву можно только вниз или вправо.

Вопрос: сколько путей ведёт из элемента-клетки (1, 1) в элемент-клетку (N, M)?

Решаем. Понятно, что из клетки (1,1) в любую клетку первой строки и любую клетку первого столбца ведёт только один путь (по первой строке – вправо и по первому столбцу – вниз). Заполняем единицами (обозначающими единственный путь) первую строку и первый столбец (рис. 1).

	1	2	3	4
1	1	1	1	1
2	1			
3	1			
4	1			

Рис. 1

Как видно из рис. 2 в клетку (2, 2) из клетки (1, 1) ведут 2 пути.

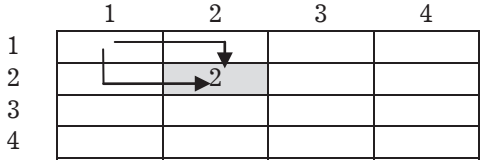

Рис. 2

В клетку (2, 3) ведут 3 пути (рис. 3) и т. д.

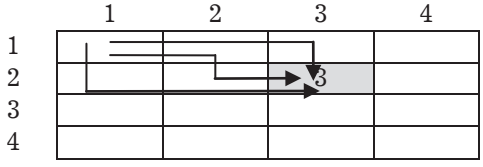

Рис. 3

Таким образом, количество путей, ведущих в клетку из клетки (1, 1), зависит от значений верхнего и левого элементов массива (рис. 4).

	1	2	3	4
1		1	1	1
2	1	1+1=2	2+1=3	3+1=4
3	1	2+1=3	3+3=6	6+4=10
4	1	1+3=4	4+6=10	10+10=20

Рис. 4

Арифметический квадрат заполнен! Итого в клетку (4, 4) ведёт 20 путей (при условии, что двигаться можно только вниз или вправо, начало пути 0 клетка (1, 1)).

Решение задачи на Паскале:

```
program pr;
const n,mm=10;
var    a: array [1..nn,1..mm] of integer;
       i,j: integer;
begin
       writeln ('введите размерность массива');
       readln (n,m);
       for j:=1 to m do
              a[1,j]:=1;
       for i:=1 to n do
              a[i,1]:=1;
       for i:=2 to n do
              for j:=2 to m
                     a[i,j]:=a[i-1,j]+a[i,j-1]
       writeln (a [n,m]);
end.
```

Решение следующих задач будет основываться на разобранном выше алгоритме заполнения арифметического квадрата. Как, например, классическая математическая задача *"Треугольник Паскаля". Нам предстоит вывести его на экран* (рис. 5).

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

Рис. 5

При внимательном рассмотрении рис. 5 становится понятным правило получения чисел, составляющих треугольник. А теперь проведём мысленный эксперимент – наклоните голову к правому плечу и посмотрите на рис. 5. Знакомая картинка? Правильно, треугольник Паскаля совпадает с элементами, расположенными выше и на побочной диагонали арифметического квадрата (рис. 6)!

	1	2	3	4
1	1	1	1	1
2	1	2	3	4
3	1	3	6	10
4	1	4	10	20

Рис. 6

Значит, нам необходимо заполнить арифметический квадрат (используя при этом квадратный массив) и вывести на экран элементы, расположенные на и выше побочной диагонали квадратного массива.

Решение задачи на Паскале:

```
program pr;
const  nn=10;
var    a: array [1..nn,1..nn] of integer;
       n,i,j: integer;
```

```
begin
       writeln ('введите размерность массива');
       readln (n);
       for j:=1 to do
            begin
            a[1,j]:=1;
            a[i,1]:=1;
            end;
       for i:=2 to n do
            for j:=2 to m
                  a[i,j]:=a[i-1,j]+a[i,j-1];
       for i:=1 to n do
            begin
            for j:=1 to (n-i+1) do
                  write (a[i,j]);
            writeln;
            end;
end.
```

Вот и настала очередь *«Бинома Ньютона». Необходимо раскрыть скобки в выражении* $(A+B)^n$.

Для начала вспомним математическую составляющую этой задачи. Из курса математики мы знаем, что:

$(A+B)^2 = A^2B^0+2A^1B^1+A^0B^2$
$(A+B)^3 = A^3B^0+3A^2B^1+3A^1B^2+A^0B^3$
$(A+B)^4 = A^4B^0+4A^3B^1+6A^2B^2+4A^1B^3+A^0B^4$
$(A+B)^5 = A^5B^0+5A^4B^1+10A^3B^2+10A^2B^3+5A^1B^4+A^0B^5$
и т. д.

Обратите внимание, что количество слагаемых при раскрытии скобок равно $(n+1)$. В качестве слагаемых выступают произведения A×B. Степень сомножителя A уменьшается от n до 0, степень B – увеличивается от 0 до n. Так как n нам будет дано (значение вводится с клавиатуры), то запрограммировать все зависимости от n не составит труда.

А вот где взять коэффициенты при слагаемых (так называемые биномиальные коэффициенты): для $n=4$ это 1, 4, 6, 6, 4, 1; для $n=5$ это 1, 5, 10, 10, 5, 1 и т. д.?

Ответ прост: эти коэффициенты находятся *на основании треугольника Паскаля* (на побочной диагонали арифметического квадрата размерностью $(n+1)\times(n+1)$)!

Решение задачи на Паскале:

```
program pr;
const m=10;
var    a: array [1..m,1..m] of integer;
      st,n,i,j: integer;
begin
       writeln ('введите степень ');
       readln (st);
       n:=st+1;
```

```
          for i:=1 to n do
               begin
               a[1,i]:=1;
               a[i,1]:=1;
               end;
          for i:=2 to n do
               for j:=2 to (n-i+1) do
                      a[i,j]:=a[i-1,j]+a[i,j-1];
          for i:=1 to n do
                 write (a[i,n-i+1], '*a^', n-i, '*b^', i-1, '+');
end.
```

Ну, и последняя задача: *дан двумерный массив, заполненный числами. Найти кратчайший путь в двумерном массиве из клетки (1,1) в клетку (N, M) (путём называется сумма значений проходимых клеток). Оговоримся, что двигаться можно только вниз либо вправо.*

Начнём её решать. Допустим, массив заполнен так, как на рис. 7.

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

Рис. 7

Предположим, что мы будем двигаться от клетки (1,1) вправо по первой строке. Мы уже выяснили, что в каждую клетку первой строки ведёт только один путь (как в арифметическом квадрате). Тогда пройденный к каждой клетке путь будет вычисляться как сумма значения данного элемента и элемента, стоящего левее (рис. 8).

	1	2	3	4
1	1	5(1+4)	7(5+2)	8(7+1)
2	5	3	1	4
3	2	1	3	5

Рис. 8

Аналогично заполняем первый столбец (рис. 9):

	1	2	3	4
1	1	5	7	8
2	6(1+5)	3	1	4
3	8(6+2)	1	3	5

Рис. 9

Так же, как и в арифметическом квадрате, в клетку (2, 2) можно попасть двумя путями – из клетки, стоящей выше, или клетки, стоящей левее. Анализируем значения этих клеток (клетки, стоящей выше и клетки, стоящей левее). Из той клетки, значение которой меньше, мы и придём в (2, 2). Значит, значение клетки (2, 2) увеличивается на 5 (рис. 10):

	1	2	3	4
1	1	**5**	7	8
2	**6**	8(3+5)		
3	8			

Рис. 10

Аналогично заполняем остальные элементы массива – анализируем содержимое элемента, стоящего сверху и элемента слева. То значение, которое меньше, и прибавляем к содержимому текущего элемента.

Следующий этап: вычисление координат (обратный ход). Движемся в обратном направлении: из клетки (n, m) к клетке (1, 1). Анализируем содержимое элемента, стоящего левее и выше текущего. Координаты того элемента, значение которого меньше, и выводим на экран (рис. 11):

	1	2	3	4
1	**1**	**5**	**7**	8
2	6	8	**8**	12
3	8	9	**11**	**16**

Рис. 11

Начиная анализ клетки (3,4), двигаемся вверх и влево.

Итого: кратчайший путь – (3,4) – (3,3) – (2,3) – (1,3) – (1,2) – (1,1).

Решение задачи на Паскале:

```
program pr;
const nn=10;
mm=10;
var    a: array [1..nn,1..mm] of integer;
       i,j,n,m,x: integer;
begin
       writeln ('введите размерность массива');
       readln (n, m);
       for i:=1 to n do
              for j:=1 to m do
                 readln (a[i,j]);
       for j:=2 to m do
              a [1,j]:= a [1,j]+ a [1,j-1];
       for i:=2 to n do
               a [i,1]:= a [i,1]+ a [i-1,1];
       for i:=2 to n do
            for j:=2 to m do
              if a [i-1,j]<a [i,j-1] then a [i,j]:=a [i,j]+a [i-1,j]
               else a [i,j]:=a [i,j]+a [i,j-1];
       i:=n;
       j:=m;
       {======вывод координат в обратном порядке======}
       writeln (n,m);
       for x:=1 to n+m-3 do
              begin
              if a [i-1,j]<a [i,j-1] then i:=i-1
              else j:=j-1;
              if (i<>0) and (j<>0) then writeln (i,j);
              end;
       writeln (1,1);
end.
```

Мы решили все задачи, опираясь на типовые алгоритмы обработки двумерных массивов. Задачи эти классические, из области математики. Схожесть с орнаментом геометрической модели этих задач, их практическое применение, выходящее за пределы условия, даже их названия, переплетающиеся с великими именами, – всё это делает эти задачи невероятно «красивыми»! А уж насколько различны их условия, вы могли судить сами! А вот решение у них практически одно – основанное на заполнении арифметического квадрата. Внесём же этот метод заполнения массива в копилочку стандартных типовых алгоритмов обработки двумерных массивов и ещё раз убедимся, насколько они занимательны!