Глава 1

Использование массивов для решения математических задач

Под треугольником Паскаля подразумевается бесконечный треугольник, состоящий из целых чисел. В каждой строке первый последний элемент равен единице, остальные — сумме элементов, находящихся над ним. На рисунке 1.1 *а* представлены первые 4 строки треугольника Паскаля. Треугольник Паскаля применяется во многих областях алгебры чисел и комбинаторики (например, с его помощью можно определить число сочетаний, числа Фибонначчи, степень числа 2 и многое другое.)

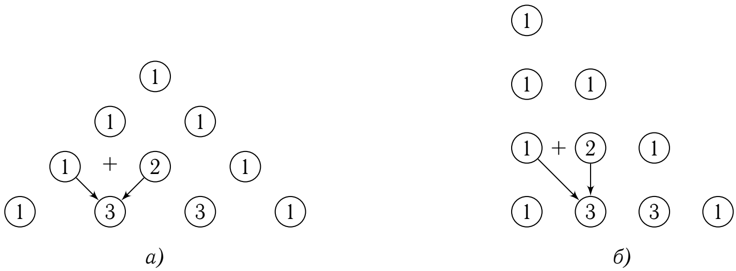

Рис. 1.1. Треугольник Паскаля, представленный в *а)* обычной и *б)* прямоугольной форме

Треугольник Паскаля можно переписать в другом виде (см. рисунок 1.1 *б*). В таком случае треугольник Паскаля можно представить в виде динамического двумерного массива.

Каждая `i`-ая строка содержит `i+1` элементов. Элементы массива с номерами `[i][0]` и `[i][i]` равны единице, элемент с номером `[i][j]` равен сумме элементов с номерами `[i-1][j-1]` и `[i-1][j]`.

В листинге 1.1 приведен код программы, реализующий ввод и вывод треугольника Паскаля длины `n`. Можно запустить эту программу и убедиться, что в результате получается треугольник, изображенный на рисунке 1.1.

Листинг 1.1. Создание треугольника Паскаля

```
#include<iostream>
using namespace std;

int main(){
```

```
int n;
cout << "n ="; cin >> n;          // кол-во строк треугольника
int **a = new int *[n + 1];
for (int i = 0; i <= n; i++)
  a[i] = new int [i + 1];       //выделяем память под i-ую строку
/*********заполнение массива***********************/
a[0][0] = 1;
a[1][0] = a[1][1] = 1;
for (int i = 2; i <= n; i++){
  a[i][0] = 1;
  for (int j = 1; j < i;  j++)
    a[i][j] = a[i-1][j-1] + a[i-1][j];
  a[i][i] = 1;
}
/*********вывод массива***************************/
for (int i = 0; i <= n; i++, cout << endl)
  for (int j = 0; j <=i; j++)
    cout << a[i][j] << "  ";
return 0;
}
```

Другой пример использования массивов — это работа с многочленами.

Пусть многочлен вида $\sum_{i=0}^{n} a_i x^i$ представлен в виде массива своих коэффициентов:

```
int *a = new int [n + 1]; //коэффициенты от a0 до an включительно
```

Тогда задачи нахождения производной или интеграла состоят в нахождении нового массива.

Например, для производной коэффициенты определяются по формуле

$(a_i x^i)' = i a_i x^{i-1}$,

следовательно, элементы нового массива будут определяться как

```
b[i - 1] = i*a[i];
```

Для интегралов аналогично.

Произведение двух многочленов можно представить в следующем виде:

$(a_0 + a_1 x + a_2 x^2) \times (b_0 + b_1 x + b_2 x^2)$

$= a_0 b_0 + a_0 b_1 x + a_0 b_2 x^2 + a_1 b_0 x + a_1 b_1 x^2 + a_1 b_2 x^3 + a_2 b_0 x^2 + a_2 b_1 x^3 + a_2 b_2 x^4$

Распишем отдельно коэффициенты при соответствующих степенях:

$x^0 - a_0 b_0$

$x^1 - a_0 b_1 + a_1 b_0$

$x^2 - a_0 b_2 + a_1 b_1 + a_2 b_0$

$x^3 - a_1 b_2 + a_2 b_1$

$x^4 - a_2 b_2$

Как можно видеть коэффициент при x^k определяется как сумма произведений коэффициентов, сумма индексов которых равна k :

$x^k - \sum\limits_{i+j=k} a_i b_j.$

Следовательно, при произведении двух многочленов степеней n и m, степень нового многочлена будет $m + n$, а коэффициенты вычисляться по приведенной выше формуле:

```
//...
int main(){
 //...
 float *a = new float [n + 1];
 float *b = new float [m + 1];
 float *c = new float [m + n + 1];
  //...
 for (int k = 0; k < m + n + 1; k++){
  float S = 0;
  for (int i = 0; i < n + 1; i++)
   for (int j = 0; j < m + 1; j++)
    if (i + j == k) S += a[i]*b[j];
  c[k] = S;
 }
  //....
  return 0;
}
```