1 Одномерные массивы

Массив есть средство для объединения некоторого количества однотипных переменных под одним именем, доступных по номеру. Главное достоинство массива по сравнению с отдельными переменными состоит в том, что номер, выбирающий конкретную переменную из всей совокупности, не обязан быть константой, а может быть результатом любого выражения, возвращающего целый результат.

2 Определение и инициализация

Обычный одномерный массив в C++ определяется следующим образом:

```
тип_элемента имя_массива[число_элементов];
```

При этом можно сочетать в одной декларации определение нескольких массивов с одинаковым типом элементов, и даже сочетать в одной декларации определение массивов с определением обычных переменных указанного типа. Например, декларация

```
int A[10], B[20], C;
```

определяет два массива A и B из 10 и 20 целых чисел соответственно, а также обычную целую переменную C. При определении массива число его элементов должно быть константным выражением, т. е. таким, в которое входят только константы и стандартные операции, и его значение должно быть известно на момент компиляции программы (а в C++ бывают и такие константы, значение которых на момент компиляции неизвестно).

Как и обычные переменные, массивы в C++ можно инициализировать в определении, т. е. прямо в определении указывать начальные значения их элементов. Делается это так:

```
тип_элемента имя_массива[число_элементов] = { элементы_через_запятую };
```

При этом число начальных значений элементов должно быть не больше указанного в квадратных скобках числа элементов массива. Меньше оно может быть; в этом случае недостающие начальные значения считаются нулями. Если начальные значения указаны для всех элементов, число элементов массива в квадратных скобках можно не писать — компилятор может посчитать число начальных значений сам, и выделить под них массив соответствующего размера. Например, следующая декларация инициализирует массив целых чисел нулями

```
int A[100] = { 0 };
```

А следующая — числами по порядку, при этом размер массива определяется автоматически:

```
int A[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

В случае, когда размер массива определяется автоматически, он может быть вычислен при помощи выражения

```
sizeof(имя_массива)/sizeof(тип_элемента)
```

Например, для последней декларации массива A значением выражения sizeof(A)/sizeof(int) будет 10.

Для удобства инициализации массивов символов, если тип элемента массива — char, то можно инициализировать такой массив при помощи строковой константы. Составляющие ее символы в этом случае записываются в массив, начиная с первого элемента по порядку, и после последнего символа строковой константы записывается еще один символ с кодом 0. Это нужно для того, чтобы строка, записанная таким образом в массив, правильно обрабатывалась функциями для работы со строками из стандартной библиотеки языка C, доставшейся языку C++ по наследству. К только что описанному способу инициализации массивов символов относятся те же замечания, что и к обычному (насчет количества начальных значений).

Например, можно написать следующее:

```
char s[10] = "Hello!";
```

Такая декларация не только определит массив s из 10 символов, но и запишет в него строку «Hello!» с завершающим символом с кодом 0. А вот следующая декларация

```
char s[6] = "Hello!";
```

ошибочна, поскольку в такой массив завершающий нулевой символ не влезет.

3 Доступ к элементу

Доступ к элементу одномерного массива осуществляется при помощи конструкции

```
имя_массива[индекс]
```

причем эту конструкцию можно использовать в выражениях (тогда берется значение указанного элемента массива) или в левой части операции присваивания (тогда указанному элементу массива присваивается значение выражения, стоящего в правой части этой операции присваивания).

Однако, нужно помнить следующие правила доступа к элементам массивов в C++:

1) в C++ индексы всех массивов всегда начинаются с 0, и должны быть целыми числами;

2) C++ не проверяет правильность индекса, т. е. тот факт, что индекс лежит в допустимых пределах (от 0 до числа элементов массива−1).

Поскольку тип char считается в C++ целочисленным, допустимо использовать переменные и константы такого типа в качестве индексов, однако нужно помнить, что на некоторых компиляторах для символов, не входящих в набор ASCII, эти значения могут быть отрицательными.

4 Присваивание

Увы, даже если у нас есть два массива одинакового типа, т. е. состоящие из одного и того же числа элементов одного типа, язык C++ не позволяет присвоить один массив другому. Т. е., если у нас есть, например, массивы A и B, определенные декларацией

```
int A[10], B[10];
```

мы не можем переписать значения элементов массива A в массив B, написав в программе B=A; Чтобы это сделать, имеется два варианта:

а) явно воспользоваться циклом:

```
for(int i = 0; i<10; i++) B[i] = A[i];
```

б) воспользоваться алгоритмом copy, написав

```
copy(A, A+10, B);
```

Этот способ требует включить в программу заголовочный файл algorithm, и будет обсуждаться подробно значительно позже.

5 Передача в функции в качестве параметров

Массивы можно передавать в функции в качестве параметров. Синтаксис этого похож на передачу параметра любого другого типа: в заголовке функции в списке параметров указывается определение массива, а при вызове на соответствующем месте ставится имя массива.

Например, следующая функция принимает массив из 10 вещественных чисел:

```
void do_something(double data[10]);
```

Если x — массив из 10 вещественных чисел, определенный при помощи декларации

```
double x[10];
```

то можно вызвать функцию do_something, передав ей x в качестве параметра, написав

```
do_something(x);
```

Однако, при передаче массивов в качестве параметров всегда передается только адрес первого элемента массива в памяти. Это обстоятельство имеет два важных последствия:

1) массивы всегда передаются по ссылке, т. е. все изменения массива внутри вызываемой функции видны в вызывающей функции;

2) размер массива при передаче его в другую функцию не столь важен; это позволяет использовать конструкцию, допускающую передачу массива из любого числа элементов — нужно просто опустить число элементов (оставив пустые квадратные скобки) в определении соответствующего формального параметра. При этом, однако, необходимо позаботиться о том, чтобы вызываемая функция могла определить размер того массива, который ей передан. Самый популярный и самый простой способ — ввести дополнительный целый параметр, через который передать длину массива. Также возможен вариант, называемый сигнальным значением — вводится договоренность о том, что определенное значение означает последний элемент массива. Например, этот способ используется в функциях обработки строк — символ с кодом 0 считается завершающим строку. Например, следующая функция принимает массив вещественных чисел любой длины и число его элементов и возвращает среднее арифметическое его элементов:

```
double average(double numbers[], int n)
{
    double sum = 0.0;
    for(int i = 0; i<n; i++)
        sum += numbers[i];
    return sum/n;
}
```

При этом, разумеется, компилятор никак не может проверить правильность переданной в отдельном параметре длины массива; за это отвечает программист, и если длина неправильная, программа, скорее всего, работать не будет.

Увы, возвратить из функции массив нельзя. Однако для одномерных массивов это ограничение легко обойти, вернув из функции указатель на первый элемент результирующего массива (подробно мы обсудим это, когда речь пойдет об указателях, на следующем занятии). Нужно только помнить, что (как и для обычных переменных), возвращать даже таким способом локальный массив из функции нельзя (а можно только глобальный, статический (с ключевым словом static перед типом элемента, отличается от обычного локального тем же, чем статическая переменная отличается от обычной локальной) или динамический — что это такое в C++, узнаем позже).

Задачи.

1. Написать функцию, принимающую в качестве параметра массив целых чисел типа ar а) из 25 элементов б) из любого числа элементов и его длину и умножающую элементы этого массива с четными индексами на 2, а с нечетными — на 3.

2. Объявить массив целых чисел из 10 элементов и ввести его элементы с клавиатуры. Посчитать, сколько раз сочетание 1 2 3 встречается в данном массиве (при этом, между 1, 2 и 3 не могут стоять другие элементы). Например, в массиве 1 2 3 0 1 2 0 3 4 6 сочетание 1 2 3 встречается всего 1 раз.

3. Написать функцию, которая принимает в качестве параметра одномерный массив целых чисел из 15 элементов и изменяет этот массив следующим образом: все 5 заменить на 1, а все 2 — на 0.

4. Ввести 50 целых чисел с клавиатуры и посчитать, сколько из введенных чисел было отрицательных, равных 0, 1, …, 100, больших 100. Результат вывести в виде таблицы, в которой моды (значения от 0 до 100, встречающиеся чаще всего) помечены символом «*»).

5. Ввести с клавиатуры массив из 50 чисел от 0 до 20 и распечатать его как обычную столбцовую диаграмму (элементам массива соответствуют столбцы из символов «*»).

6. Написать функцию, принимающую в качестве параметров три массива, первые два по 50 элементов каждый, третий из 100 элементов, и записывающую элементы из первых двух массивов в третий в порядке возрастания (предполагается, что в первых двух массивах элементы также записаны в порядке возрастания).

7. Написать процедуру, принимающую по ссылке массив из 100 элементов и переставляющую его элементы так, чтобы последний элемент оказался на таком месте, чтобы все стоящие до него элементы были не больше его, а все стоящие после — не меньше, наиболее быстрым образом.

8. Массив из вещественных чисел называется пирамидой, если элемент с индексом i не больше каждого из элементов с индексами $2i + 1$ и $2i + 2$. Написать функцию, принимающую массив из 100 вещественных чисел, про который известно, что он — пирамида с точностью до одного элемента, стоящего не на своем месте, и индекс этого элемента, и исправляющую указанный массив при помощи перестановок его элементов, за время порядка двоичного логарифма от длины массива.

9*. Написать функцию, принимающую массив целых чисел и его длину, и вычисляющую наибольшую сумму набора подряд (без пропусков) идущих его элементов за один просмотр этого массива.

10*. Написать функцию, принимающую массив целых чисел и его длину, а также число n, и осуществляющую циклический сдвиг его элементов вправо на n позиций, без использования дополнительных массивов.

(Касьянов В. Н., Сабельфельд В. К., Сборник заданий по практикуму на ЭВМ).

11. Диаметром множества точек называется максимальное расстояние между двумя точками из этого множества (если множество конечное, этот максимум всегда существует). Ввести координаты 15 точек на плоскости и вычислить диаметр этого множества.

12. Ввести координаты 10 точек на плоскости и выбрать из них две различных так, чтобы количества точек, лежащих по разные стороны от прямой, проходящей через пару выбранных, отличались наименьшим образом.

13. Ввести последовательность из 12 вершин многоугольника на плоскости (не обязательно выпуклого, вершины вводятся в порядке их обхода вдоль границы, начиная с любой вершины) и вывести площадь многоугольника.

14. Ввести 15 целых чисел и вывести максимальную длину подпоследовательности, являющейся арифметической прогрессией: а) члены подпоследовательности должны идти в исходной последовательности подряд (между ними не допускается других членов исходной последовательности) б) без ограничения, содержащегося в пункте а).

15. Ввести 15 целых чисел и вывести максимальную длину симметричной подпоследовательности: а) члены подпоследовательности должны идти в исходной последовательности подряд (между ними не допускается других членов исходной последовательности) б) без ограничения, содержащегося в пункте а).

6 Многомерные массивы

Наряду с обычными одномерными массивами в математике часто встречаются матрицы, т. е. таблицы с двумя индексами. Язык C++ поддерживает не только одномерные и двумерные массивы, но массивы с любым числом индексов. Правда, число элементов массива быстро растет с ростом размерности массива, поэтому массивы с большим числом индексов практически не встречаются.

7 Определение и инициализация

Двумерный массив в C++ определяется следующим образом:

```
тип_элемента имя_массива[число_строк][число_столбцов];
```

Увы, эту запись нельзя сократить до

```
тип_элемента имя_массива[число_строк,число_столбцов];
```

как в Pascal.

Можно определить и массив большей размерности, для чего нужно написать соответствующее количество квадратных скобок с указанием размера массива в каждой координате (соответствующий индекс меняется от 0 до указанного числа−1). Например, трехмерный массив можно определить так:

```
int cube[10][10][10];
```

Число элементов многомерного массива растет очень быстро (как показательная функция) в зависимости от его размерности, поэтому массивы большой размерности практически не используются.

Инициализировать двумерный массив можно двумя способами: либо перечислить все его элементы подряд по строкам, как и одномерный массив, либо указав элементы каждой строки отдельно, заключив их в {}. Например, обе следующих декларации вполне законны и присваивают элементам массива M одинаковые начальные значения:

```
int M[3][2] = { 1, 2, 3, 4, 5, 6 };
int M[3][2] = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```

8 Доступ к элементу

Доступ к элементу двумерного массива осуществляется следующим образом:

```
имя_массива[индекс_строки][индекс_столбца];
```

Как и в определении двумерного массива, каждый индекс должен быть заключен в свои квадратные скобки.

9 Передача в функции в качестве параметров

Многомерные массивы также можно передавать в функции в качестве параметров. При этом в заголовке функции можно опустить только самый первый размер, все остальные должны быть указаны явно. Например, следующая функция принимает в качестве параметра квадратную матрицу порядка 10:

```
void f1(double M[10][10]);
```

А следующая — двумерный массив из любого числа строк, каждая из которых содержит 15 элементов:

```
void f2(double M[][15]);
```

А следующий прототип ошибочен (второй размер массива не указан):

```
void f3(double M[][]);
```

Если A — квадратная матрица порядка 10, то можно вызвать функцию f1, передав ей A в качестве параметра так: f1(A);

Также, если функции требуется, например, одномерный массив, а у нас есть двумерный, то можно передать в эту функцию какую-либо строку двумерного массива. Для этого в вызове нужно поставить имя многомерного массва и за ним в квадратных скобках индекс строки (всего одна пара квадратных скобок). Например, если есть функция

```
void f1(double M[10]);
```

и массив

```
double ar[17][10];
```

то можно вызвать f1, передав ей в качестве параметра i-ю строку матрицы ar, написав f1(ar[i]);

Двумерный массив труднее возвратить из функции, поскольку для этого нужно описать функцию как возвращающую указатель на массив, а именно на строку того массива, который мы хотим вернуть, и нужно вернуть из функции указатель на первую строку результата.

Задачи

1. Написать функцию, которая принимает в качестве параметра двумерный массив 20×20 (квадратную матрицу) и возвращает ее след (сумму диагональных элементов).

2. Написать функцию, принимающую в качестве параметров три двумерных массива A, B и C одинакового размера (числа строк и столбцов — именованные константы) и записывающую сумму матриц $A + B$ в C.

3. Дан двумерный массив целых чисел; определить среднее значение номера позиции первой единицы в каждой строчке (в массиве ((0,1,1),(1,2,1),(0,2,1)) в первой строке первая единица встречается в позиции 1, во второй строке - 0, в третьей - 2, так что среднее $(1+0+2)/3 = 1$).

4. Написать функцию, принимающую в качестве параметра двумерный массив и заменяющую в нем каждый элемент на сумму его соседей сверху, снизу, справа и слева (отсутствующие соседи считаются равными нулю; все элементы должны меняться одновременно, т. е. во всех суммах участвуют старые значения элементов).

5. Дан двумерный массив; найти среднее значение его элементов, причем элементы, окруженные с четырех сторон единицами, в этом среднем не участвуют (для массива $\{\{2,1,5\},\{1,6,1\},\{3,1,8\}\}$ ответ будет (2+1+5+1+1+3+1+8)/8 = 22/8 = 2.75).

6. Написать функцию, принимающую в качестве параметров двумерный массив A и два одномерных массива B и C подходящего размера и записывающую в вектор C произведение матрицы A на вектор B.

7. Написать функцию, принимающую две матрицы A и B подходящего размера и записывающую A^TA в B.

8. Написать функцию, принимающую в качестве параметров три двумерных массива подходящего размера A, B и C и записывающую произведение матриц AB в C.

9. Посчитать многочлен от квадратной матрицы.

10. Написать функцию, принимающую в качестве параметров двумерный массив и два целых числа (номера строк), и меняющую местами две строки с указанными номерами в массиве-параметре.

11. Написать функцию, принимающую в качестве параметров двумерный массив, два целых числа (номера строк) и вещественное число (коэффициент), и вычитающую в массиве-параметре строку с первым номером, умноженную на коэффициент, из строки со вторым номером.

12. Написать процедуру, приводящую матрицу из вещественных чисел к ступенчатому виду при помощи элементарных преобразований над строками (см. задачи 10 и 11). Ступенчатый вид матрицы — такой, при котором каждая следующая строка начинается с большего числа нулей, чем предыдущая, кроме, возможно, нескольких последних строк матрицы, полностью состоящих из нулей.

13. Пользуясь решением задачи 12, написать функцию, вычисляющую определитель квадратной матрицы.

14. Вычислить обратную матрицу.

15. Изобретенная Конвеем игра "Жизнь"состоит в следующем. Плоскость делится на единичные квадраты прямыми x=n и y=k (k и n пробегают независимо все целые числа). В каждом квадрате в каждый момент времени (время дискретно) может быть пусто или жить одна клетка. Состояние клетки в следующий момент времени определяется правилами:

а) если у клетки число живых клеток на восьми соседних квадратах меньше двух или больше трех, клетка погибает;

б) если у пустого квадрата число живых клеток на восьми соседних квадратах равно трем, в этом квадрате рождается новая клетка;

в) в остальных случаях состояние клетки не меняется.

Написать функцию, принимающую двумерный массив клеток и вычисляющую его состояние в следующий момент времени.

16. Дан двумерный массив 8x8, представляющий шахматную доску (элемент, равный 0 означает пустую клетку, 1 означает, что на этой клетке стоит ферзь). Определить, найдутся ли два ферзя, бьющие друг друга.

17. Дан двумерный массив, элементы которого равны 0 или 1, причем известно, что единицы образуют набор прямоугольников, не соприкасающихся сторонами. Посчитать количество таких прямоугольников.

(Касьянов В. Н., Сабельфельд В. К., Сборник заданий по практикуму на ЭВМ).

18. Написать функцию, принимающую в качестве параметра квадратную матрицу 7×7 и печатающую ее элементы по спирали, начиная с центрального элемента (первое звено спирали идет вверх).

19. Написать функцию, принимающую в качестве параметра квадратную матрицу 7×7 и печатающую ее элементы по рядам, параллельным главной диагонали, начиная с верхнего правого угла, причем направление прохода этих рядов чередуется: сверху-вниз, снизу-вверх и т. д.

10 Сортировка

Задачи

1. Реализовать алгоритм сортировки пузырьком, состоящий в следующем. Делается n (размер массива) проходов, на каждом из которых последовательно просматриваются пары соседних элементов, и если их порядок неправилен, они меняются местами. Предложить и реализовать улучшения к этому алгоритму.

2. Реализовать алгоритм сортировки отбором, состоящий в следующем. В исходном массиве ищется наименьший элемент и он меняется местами с первым элементом массива. Затем процесс повторяется для подмассива, начинающегося со второго элемента, затем — с третьего, и т. д. Когда последние два элемента займут свои места, массив окажется отсортированным.

3. Реализовать алгоритм сортировки вставками, состоящий в следующем. Последовательно просматривается сортируемый массив, и каждый очередной элемент ставится на свое место в уже отсортированной части массива (от начала до текущего элемента).

4. Реализовать алгоритм блочной сортировки, состоящий в следующем. Для сортировки массива из n неотрицательных целых чисел заводится дополнительный двумерный массив размера n x 10, и сортировка состоит из последовательно чередующихся распределительного и собирательного проходов. Во время распределительного прохода последовательно просматривается исходный массив, и очередное число записывается в ту строку двумерного массива, номер которой равен значению соответствующего разряда этого числа (на первом распределительном проходе анализируется разряд единиц, на втором — десятков, на третьем — сотен и т. д.). На собирательном проходе в исходный массив записывается сначала содержимое нулевой строки двумерного массива, затем — первой, и т. д. Пары распределительный-собирательный проход повторяются до тех пор, пока не будут исчерпаны разряды максимального элемента массива.

5. Реализовать алгоритм быстрой сортировки, состоящий в следующем. Берется последний элемент массива (его значение назовем x), и заводятся две переменные, хранящие индексы первого и последнего элемента в оставшейся части массива. Значение первой из этих переменных увеличивается, а второй уменьшается до тех пор, пока не будет выполнено одно из двух условий:

а) первая переменная содержит индекс элемента, большего x, а вторая — меньшего. В этом случае значения этих элементов меняются местами, и процесс сближения индексов, содержащихся в переменных, продолжается опять до тех пор, пока не будет выполнено одно из условий а) или б);

б) индекс, содержащийся в первой переменной, больше индекса, содержащегося во второй переменной. В этом случае последний элемент (со значением x) меняется местами с тем, индекс которого содержится в первой переменной, затем каждая из двух частей массива, на которые он делится новым положением элемента со значением x, сортируется отдельно тем же алгоритмом, и массив оказывается упорядоченным.

6. Реализовать алгоритм сортировки слиянием, состоящий в следующем. Сначала тем же алгоритмом сортируются две половины исходного массива (рекурсивный вызов), затем заводится еще один вспомогательный массив той же длины, что и исходный, и в него переписываются элементы двух отсортированных частей исходного массива так, чтобы результат тоже был упорядоченным; наконец, упорядоченные элементы переписываются обратно в исходный массив.

7. Реализовать алгоритм пирамидальной сортировки, состоящий в следующем. Делается два прохода по исходному массиву. На первом проходе последовательно просматриваются элементы массива и по очереди добавляются в пирамиду. На втором проходе элементы по очереди извлекаются из пирамиды и записываются на освобождающиеся места в конце массива, в результате чего массив оказывается отсортированным.

Пирамида — это набор элементов, занимающих начальный отрезок массива и удовлетворяющих условию: каждый элемент пирамиды с индексом i не меньше элементов с индексами $2i+1$ и $2i+2$ (здесь важно, что индексы массива начинаются с 0).

Когда мы хотим добавить в пирамиду элемент с индексом j (к этому времени пирамида состоит из элементов с индексами от 0 до $j-1$), мы смотрим элемент с индексом $(j-1)/2$ (в смысле C++), и если он меньше добавлямого, меняем их местами. Далее, мы сравниваем элементы с индексами $(j-1)/2$ и $((j-1)/2-1)/2$, и меняем их местами в случае необходимости, чтобы больший элемент был ближе к началу массива, и т. д. до тех пор, пока либо не встретится больший добавляемого элемент, либо добавляемый элемент не окажется первым элементом массива.

При извлечении элемента из пирамиды мы просто меняем местами первый элемент, очевидно не меньший всех оставшихся в пирамиде, с последним элементом пирамиды, считая при этом, что пирамида сократилась на один элемент, и подправляем ее так, чтобы сохранилось свойство пирамиды. Для этого элемент с индексом 0 сравнивается с элементами с индексами 1 и 2, если среди них есть больший его, наибольший из них меняется местами с первым. Затем, если мы поменяли элемент с индексом i, мы сравниваем его новое значение с элементами с индексами $2i+1$ и $2i+2$, и если оба они не больше его, процесс прекращается, а если среди них найдется больший, наибольший из них меняется с ним местами, до тех пор, пока свойство пирамиды не восстановится.

11 Поиск

Задачи

1. Реализовать алгоритм линейного поиска как функцию с параметрами массив целых чисел, его длина и целое число n. Функция должна последовательно просматривать массив-параметр и возвращать индекс первого найденного элемента, равного n. Если указанный элемент не найден, функция возвращает -1.

2. Реализовать алгоритм двоичного поиска как функцию с параметрами массив целых чисел, его длина и целое число n. Функция должна анализировать средний элемент массива, сравнивая его с n и сужая диапазон поиска соответствующим образом, до тех пор, пока либо не будет найден элемент, равный n (в этом случае возвращается его индекс), либо интервал поиска не станет вырожденным (в этом случае понятно, что такого элемента нет, и нужно возвратить -1).

3. Реализовать алгоритм интерполяционного поиска, который отличается от двоичного тем, что вместо среднего элемента массива рассматривается элемент, полученный в результате попытки «угадать» его положение в массиве, исходя из гипотезы о равномерном распределении элементов в массиве, т. е. из гипотезы о том, что «график» элементов массива представляет собой отрезок.

4. Написать программы, позволяющие сравнить производительность различных алгоритмов поиска, следующим образом. Создать массив из 10 тысяч элементов и заполнить его случайными числами от 0 до 30 тысяч. Сделать в цикле 10 тысяч поисков и вывести число миллисекунд, затраченных на это (в качестве искомого числа использовать элемент массива со случайным индексом). Перед циклами двоичных и интерполяционных поисков отсортировать массив.

5. Написать функцию, принимающую двумерный массив целых чисел и целое значение и осуществляющую в массиве:

а) линейный поиск;

б) двоичный поиск (предполагая, что элементы массива упорядочены по строкам).

6. Написать функцию, принимающую вещественное число, массив упорядоченных вещественных чисел, его длину и определяющую пару соседних элементов массива, между которыми лежит данное число (методом, аналогичным двоичному поиску).

7. Написать функцию, принимающую отсортированный массив целых чисел, его длину и значение искомого элемента и возвращающую индекс первого элемента в массиве, имеющего искомое значение. Если в массиве такого элемента нет, нужно возвратить -1.