Златопольский Дмитрий Михайлович
Кандидат технических наук, доцент кафедры информатики и прикладной математики Московского городского педагогического университета.

Задача Иосифа Флавия

В статье описываются возможные методы решения так называемой «задачи Иосифа Флавия», известной у нас в стране как «задача о считалке». Представлены два варианта решения – с применением массивов и с использованием рекуррентных соотношений. Приводятся задания для самостоятельной работы учащихся.

В основу так называемой «задачи Иосифа Флавия» положена легенда о том, что известный историк первого века н. э. Иосиф Флавий выжил и стал известным благодаря математической одарённости. В ходе Иудейской войны он в составе отряда из 41 иудейского воина был загнан римлянами в пещеру. Предпочитая самоубийство плену, воины решили выстроиться в круг и последовательно убивать каждого третьего из живых до тех пор, пока не останется ни одного человека[1]. Однако Иосиф, наряду с одним из своих единомышленников, счёл подобный конец бессмысленным – он быстро вычислил спасительные места в круге, на которые поставил себя и своего товарища [1].

Эта задача известна у нас в стране как задача о считалке. Во многих играх (прятки и др.) для того, чтобы выяснить, кому «водить», кто-либо из играющих произносит недлинный стихотворный текст – считалку. Играющий, на которого попадает последнее слово текста, выходит из круге. Кто последним останется в круге, тому и водить.

В общем виде задача формулируется так: «По кругу размещены n человек. Задан параметр расчёта k, то есть каждый k-й человек будет выбывать из круга. Требуется определить p – порядковый номер человека, который останется в круге последним». Такую задачу можно решить несколькими способами. Опишем два из них.

[1] По-видимому, предполагалось, что последний должен был убить себя сам.

Способ 1 – использование массива

Прежде всего нужно смоделировать расчёты, происходящие в круге. Для этого необходимо выбрать структуру данных, в которой будут «размещены» n человек.

С одной стороны, самая простая структура данных, допускающая использование множества значений (номеров стоящих по кругу людей) – это массив. Но, с другой стороны, у массива нет желательного в этой задаче автоматического перехода от последнего элемента к первому. Ещё одна особенность, трудно сочетаемая с массивом, – это постоянное его изменение, связанное с исключением из расчёта людей (элементов массива). Проще всего вовсе не исключать никого из круга, а просто помечать в массиве исключённых людей специальным значением, которое будем интерпретировать как «не участвует в расчёте». Для этого информацию об n человек, участвующих в расчёте, будем хранить в массиве из n элементов с именем m. Значение $m[i] = 1$ будет означать, что человек с номером i участвует в расчёте. Если $m[i] = 0$, то человек с номером i уже выбыл из расчёта. Чтобы отыскать очередного выбывающего k-го человека, будем определять s – сумму последовательно «обработанных» элементов массива. Когда s достигнет значения k, то искомый номер (человек) найден[1]. Отыскание k-го человека повторим $n - 1$ раз. Тогда номер того человека, который останется, – искомый.

Программа на языке Паскаль на основе сделанных рассуждений имеет вид:

```
program Variant_1;
{Параметры n и k – константы}
const n = …; k = …;
var m: array[1..n] of byte; i, j, s: integer;
BEGIN
{Начальное состояние массива m}
for i := 1 to n do
  m[i] := 1; {Сначала в круге все}
{Далее величина i указывает на очередного человека в круге}
i := 1;
{n – 1 раз ищем номер исключаемого человека}
for j := 1 to n – 1 do
 begin
  {Начинаем очередной расчет}
   s := m[i];
   while s < k do
    begin
     {Пропускаем очередных людей}
     i := i + 1;
     if i > n then
       {Переход от последнего человека к первому}
         i := 1;
     s := s + m[i]
    end;
```

[1] Именно поэтому для элементов массива выбраны значения 0 и 1. В результате при определении s будут учтены ещё не выбывшие участники счёта.

```
   {Очередной "выбывающий" человек найден}
   m[i] := 0 {Исключаем его}
  end;
 {Поиск номера оставшегося человека}
i := 1;
while m[i] = 0 do i := i + 1;
{и вывод этого номера}
writeln('p = ', i)
END.
```

Способ 2 – использование рекуррентных соотношений[1]

Существует также решение обсуждаемой задачи без использования массива (в результате чего уменьшается требуемый для решения объём памяти).

Зададимся вопросом – если кто-то решил задачу для некоторых n и k, то не поможет ли это нам решить задачу для пары $(n + 1)$ и k?

Пусть, например, при $n = 100$ и $k = 20$ известно, что $p = 10$. Представим себе расчет при $n = 101$, $k = 20$. На первом шаге расчёта будет исключен человек с номером $k = 20$ (так как $k < n$). После этого в круге останется на одного человека меньше (то есть 100 человек), и расчёт продолжится с двадцать первого человека. Но нам уже известно, что при таких исходных данных последним останется человек под номером 10. Следовательно, в исходной нумерации номер этого человека вычисляется как сумма $20 + 10 = 30$.

Если же $k > n$, то тогда номер выбывающего первым вычисляется по формуле $k \bmod n$. И здесь надо прибавить к полученному номеру предыдущий ответ p.

В обоих случаях результат сложения может выйти за пределы n, тогда следует сделать уточнение – вычесть из результата n.

Отдельного обсуждения требует случай, когда k кратно n ($k \bmod n = 0$). В этом случае номер первого выбывающего равен n (убедитесь в этом самостоятельно!). Но тогда сумма предыдущего ответа p и n явно выйдет за пределы n, и после уточнения, которое упоминалось чуть выше, мы получим всё тот же ответ p для предшествующего значения n. Это означает, что и в рассматриваемом случае мы можем использовать рекуррентную зависимость «нового» ответа от «старого»: $p = k \bmod n + p$.

Учитывая все сказанное, мы можем так оформить программу:

```
program Variant_2;
var n, p, k, m: integer;
BEGIN
```

[1] Рекуррентные соотношения – формулы, выражающие значение очередного члена последовательности чисел через значения одного или нескольких предыдущих членов.

```
{Ввод исходных данных}
write('Задайте значения n и k ');
readln(n, k);
m := 1;  {См. ниже о переменной m}
p := 1;   {Ответ для n = 1 (m = 1)}
{Последовательно решаем задачу для числа людей от 2 до n}
for m := 2 to n do {Здесь число людей обозначается
                    через m - переменную цикла}
 begin
  {Рекуррентная формула: зная предыдущий ответ p, находим ответ
  для числа людей m, на 1 большего}
  p := k mod m + p; {k mod m - номер первого выбывшего}
  {Уточнение на случай выхода p за пределы m}
  if p > m then p := p - m
 end;
{Вывод номера оставшегося человека}
writeln('p = ', p)
END.
```

Задания для самостоятельной работы

1. Разработав любой из описанных вариантов программы, определите номер человека, который останется в круге последним при исходных данных задачи, представленных в таблице:

n	*k*
100	37
200	133
300	500
1000000	739
1000001	739

2. Задача «Кот и мыши»

Коту снится, что его окружили тринадцать мышей. Двенадцать из них серые, а одна белая. Слышит кот, как кто-то говорит ему: «Мурлыка, ты можешь съедать третью мышку, считая их по часовой стрелке. Если ты начнёшь счёт с белой, то какая мышь будет съедена последней? Или вот ещё тебе задача – последней ты должен съесть белую мышку. Подумай, с какой начать счёт».

Помогите коту решить эти две задачи.

3. Подумайте над решением обсуждаемой в статье задачи для частного случая – когда $k = 2$. Обращаю внимание на то, что при этом нет необходимости разрабатывать программу – задача может быть решена путём рассуждений.

Указание. Сначала рассмотрите вариант, когда значение n есть степень двойки, а затем исследуйте общий случай.

Литература

1. *Дагене В.А., Григас Г.А, Аугутис К.Ф.* 100 задач по программированию. – М.: Просвещение, 1993.

2. *Кордемский Б.А.* Математическая смекалка. – М.: Юнисам, МДС, 1994.