Задачи к главе 1

Задача № 1.1.

Функция

$$K(i,j) = \frac{(i+j-1)(i+j-2)}{2} + j$$

отображает упорядоченные пары целых на целые (*см.* табл. 1.1). Найти обратные функции I и J с таким свойством, что $I(K(i,j)) = i$ и $J(K(i,j)) = j$.

Составьте процедуру на Паскале, которая по целому $k>0$ выдает i и j — номер строчки и столбца треугольной сети, где расположено значение k.

[Retrun Гл. 1, 18]

Задача № 1.1.

Решение (Лучшее объяснение): Пусть имеется некоторое k, занимающее в сетке позицию (i, j), N — число элементов *внутри* с основанием, концы которого имеют координаты $(i, 1)$ и $(1, i)$: $N = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$,

i \ j	**1**	**2**	**3**	**4**	**n**
1	1	3	6	10	**N**
2	2	5	9		
3	4	8			
4	7			…	…

где n —число диагоналей, предшествующих основанию. Ясно, что n есть целый неотрицательный корень уравнения $N = \frac{n(n+1)}{2}$ или

$$n^2 + n - 2N = 0 \quad (1)$$

Уравнение (1) имеет целые неотрицательные корни только при N, расположенных в строчке 1.

Как узнать, число k —в 1-й строчке или нет?

Надо вычислять последовательно $N_m = 1 + 2 + \ldots + m$ ($m = 1, 2, \ldots$) до тех пор, пока при некотором m не окажется, либо

а) $N_m = k$, либо б) $N_{m-1} < k < N_m$.

В случае а) решить ур. (1) при $N = N_m$ и положить $i = 1$, $j = n$.

В случае б) $j = k - N_{m-1}$; и учитывая, что $i + j = n + 2$, где n корень уравнения (1) при $N = N_{m-1}$, получим $i = n + 2 - j$.

Пример.

Случай (а): пусть $k = 6$.

$N_1 = 1$, $N_2 = 1+2 = 3$, $N_3 = 1+2+3 = 6$.

Решаем ур-е $n^2 + n - 2N = 0$ (1)

при $N = N_3$: $n^2 + n - 12 = 0$

$$n = -\frac{1}{2} \pm \sqrt{\frac{1}{4} + 12} = -\frac{1}{2} \pm \sqrt{\frac{49}{4}} = -\frac{1}{2} \pm \frac{7}{2}; \quad n = -\frac{1}{2} + \frac{7}{2} = 3.$$

Итак, $i = 1, j = 3$.

Случай (б): пусть $k = 8$.

$N_1 = 1$, $N_2 = 1+2 = 3$, $N_3 = 1+2+3 = 6$, $N_4 = 1+2+3+4 = 10$.

$N_3 = 6 < k = 8 < N_4 = 10$. В этом случае придется решать то же самое уравнение, что и в случае (а). Оно дает $n = 3$.

Затем $j = k - N_3 = 8 - 6 = 2$, и учитывая, что $i + j = n + 2$ при $j = 2$, получаем $i = 3 + 2 - 2 = 3$.

Задача № 1.1.

Итак, можем описать процедуру, которая по целому $k>0$, выдает i, j — координаты элемента треугольной сети, где расположено значение k, следующим образом: [Run]

```
procedure I J (k : integer; var i, j : integer);
procedure couple(k : integer; var k1, i, j : integer);
var n : integer;
begin n := (round(sqrt (1+ 8*k)) – 1) div 2 ;
            if sqr (n) + n – 2*k = 0
            then if k <> k1
                  then begin {k1 не на верхней строчке}
                        j := k1 – k; i := n + 2 – j
                       end
                  else begin {k1 на верхней строчке} i := 1; j := n end
           else couple(k – 1, k1, i, j)
end {couple};
begin couple(k, k, i, j) end {I J};                    [Return 5]};
```

[Return 5] [Return 6]

Задача № 1.2

Пусть $J(w, x, y) = J(w, J(x, y))$. Какая тройка приписывается числу 1000, если

$$J(x,y) = \frac{(x + y - 1)(x + y - 2)}{2} + y$$

См. в задаче 1.1 функцию $K(i, j)$) и процедуру *I J*.

Задача № 1.2

Воспользуемся процедурой *IJ* из задачи 1.1.

Для $k = 1000$ находим, что оно равно $J(36, 10)$. В то же время $10 = J(1, 4)$. Следовательно,

$$k = 1000 = J(36, 10) = J(36, J(1, 4)) = J(36, 1, 4).$$

$$k = 1000 = J(36, 10) = J(36, 1, 4)$$

$$J(1, 4) = J(1, 1, 3)$$

$$J(1, 3) = J(1, 1, 2)$$

$$J(1, 2) = J(1, 2, 1)$$

$$J(2, 1) = J(1, 1, 1)$$

$$J(1, 1) = 1$$

Задача № 1.3

Опишите простую процедуру для перенумерации предложений рекурсивного языка.

Задача № 1.3

Решение: Пусть $L \subseteq V^*$ — рекурсивный язык. Существует алгоритм A, который распознает, $x \in L$? Перечисляющую процедуру можно организовать так:

1) Последовательно генерировать цепочки x из V^*, постепенно увеличивающейся длины, причем цепочки одной длины упорядочиваются в числовом порядке (как целые по основанию $p = \#V$).

2) Каждую цепочку x пропускать через алгоритм A. Если A распознает x, то x включается в перечисление, а иначе генерируется очередная цепочка. [Run]

Задача № 1.4

Докажите, что если язык $L \subseteq V^*$ и его дополнение $\overline{L} = V^* \setminus L$ — оба рекурсивно перечислимы, то язык L — рекурсивен.

Задача № 1.4

Докажите, что если язык $L \subseteq V^*$ и его дополнение $\overline{L} = V^* \setminus L$ — оба рекурсивно перечислимы, то язык L — рекурсивен.

Рекурсивность рекурсивно перечислимого языка, дополнение которого тоже рекурсивно перечислимо

Теорема 1.1. *Пусть* $L \subseteq V^*$ *— некоторый язык, а* $\overline{L} = V^* \setminus L$ *— его дополнение.*

Если языки L *и* $\overline{L}$ *оба рекурсивно перечислимы, то язык* L *рекурсивен.*

Доказательство. Пусть язык L распознается процедурой P, а его дополнение $\overline{L}$ распознается процедурой $\overline{P}$. Достаточно показать, как построить ***алгоритм*** для распознавания L.

Построение распознающего алгоритма

Построим *алгоритм* распознавания языка L, т. е. алгоритм, отвечающий на вопрос: $x \in L$?, следующим образом:

Шаг 1: $i := 1$.

Шаг 2: Применим i шагов процедуры P к цепочке x.
Если процедура P не завершается, то перейти к шагу 3, иначе к шагу 4.

Шаг 3: Применим i шагов процедуры $\overline{P}$ к цепочке x.
Если процедура $\overline{P}$ не завершается, то $i := i + 1$ и перейти к шагу 2.

Шаг 4: При некотором i одна из этих двух процедур завершится, распознав цепочку x, так как либо $x \in L$ — и это распознает процедура P,
либо $x \notin L$ — и это распознает процедура $\overline{P}$.

Соответственно распознающий алгоритм выдает либо положительный, либо отрицательный ответ. Поскольку язык L распознается описанным алгоритмом, то L — рекурсивен. Что и требовалось доказать.

Задача № 1.5

Докажите, что если существует процедура для перечисления множества целых в монотонном порядке, то это множество рекурсивно в том смысле, что существует алгоритм определения, находится ли данное целое в этом множестве.

Задача № 1.5

Решение: Пусть P — процедура, перечисляющая элементы множества целых M в монотонном порядке.

Нам достаточно построить алгоритм A, который по заданному целому n отвечает на вопрос: $n \in M$?

Он может быть организован так:

1) Запустим процедуру P. Пусть она выдает целое m.

2) Если $n = m$, то алгоритм A завершается с ответом “Да”. Иначе

3) Если $m > n$, то алгоритм A завершается с ответом “Нет”. Иначе повторить шаг 1.

Задача № 1.6

Покажите, что все конечные множества рекурсивны.

Задача № 1.6

Решение: Пусть M — произвольное конечное множество. Нам достаточно построить алгоритм, который по заданному m, определяет, $m \in M$?

Он может быть организован так:

1) Поочередно перебирать элементы M. Пусть i — очередной элемент.
2) Если $i = m$, то алгоритм завершается с ответом “Да”. Иначе
3) Если не все элементы множества M исчерпаны, то перейти к шагу 1, иначе алгоритм завершается с ответом “Нет”.