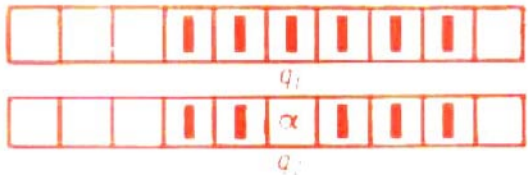
Lecciones populares
de matemáticas

LOS ALGORITMOS Y LA RESOLUCIÓN AUTOMÁTICA DE PROBLEMAS

B. A. Trajtenbrot



Editorial MIR Moscú

ПОПУЛЯРНЫЕ ЛЕКЦИИ ПО МАТЕМАТИКЕ

Б. А. ТРАХТЕНБРОТ

АЛГОРИТМЫ И МАШИННОЕ РЕШЕНИЕ ЗАДАЧ

ИЗДАТЕЛЬСТВО
«НАУКА»

LECCIONES POPULARES DE MATEMATICAS

B. A. TRAJTENBROT

LOS ALGORITMOS Y LA RESOLUCION AUTOMATICA DE PROBLEMAS

EDITORIAL MIR
MOSCU

TRADUCIDO DEL RUSO POR BERNARDO DEL RÍO SALCEDA,
CANDIDATO A DOCTOR EN CIENCIAS TECNICAS

На испанском языке

IMPRESO EN LA URSS
1977

INDICE

PREFACIO

Este libro, que es una introducción elemental a la teoría de los algoritmos, está dedicado a la explicación de uno de los conceptos esenciales de las matemáticas, al del algoritmo. En el libro se examinan cuestiones limítrofes de la lógica matemática y la teoría de las máquinas automáticas de tratamiento de la información.

El libro fue escrito a base de las conferencias de divulgación y los informes generales que dio el autor en la ciudad de Penza desde el año 1951 ante diferentes auditorios y del artículo del mismo nombre publicado en la revista „Математика в школе" («Las matemáticas en la escuela») (N°N° 4—5, 1956).

A aquellos que deseen estudiar con más profundidad estas cuestiones se les puede recomendar el libro: Б. А. Трахтенброт, „Алгоритмы и вычислительные автоматы", издательство Советское Радио, Москва, 1974 (B. A. Trajtenbrot, «Los algoritmos y los autómatas de cómputo, editorial Soviétskoye Radio, Moscú, 1974).

INTRODUCCION

En los años de postguerra las computadoras de alta velocidad han tenido un considerable desarrollo. Hoy en día se emplean para la resolución de los más variados problemas matemáticos y lógicos. La característica peculiar de estas computadoras, la que las distingue de las máquinas de calcular anteriores, consiste en que, al cumplir sus funciones, ellas, desde el momento en que se introducen los datos iniciales y el programa hasta que se imprime el resultado final, trabajan sin ninguna intervención del hombre. La productividad de las computadoras electrónicas modernas es enorme: ellas realizan cientos de miles de operaciones aritméticas en un segundo, lo que es por lo menos 100 veces más de lo que puede hacer en un solo turno un empleado de alta calificación que trabaje con un buen aritmómetro de teclas*). La esfera del empleo de las computadoras automáticas continúa ampliándose: las máquinas resuelven complejos sistemas de ecuaciones, traducen de una lengua a otra, juegan al ajedrez, etc. Las perspectivas del empleo de las computadoras automáticas en la industria son enormes, ellas pueden realizar el control de todos los procesos tecnológicos en grandes fábricas. Además, la posibilidad de un rápido y seguro tratamiento de la información y también de un análisis de datos experimentales crea la premisa para que aparezcan métodos nuevos de investigación que antes no estaban al alcance en muchas ramas de la ciencia.

Hoy, ya está completamente reconocido que las computadoras automáticas son un potente instrumento del trabajo intelectual, capaces no sólo de aligerar al hombre de este trabajo, sino de liberarlo por completo de algunas clases de un grande y tenso trabajo mental.

Al mismo tiempo los éxitos conseguidos pueden crear y crean muchas injustificadas ilusiones y pronósticos puramente fantásticos sobre la omnipotencia de estas máquinas. Particularmente se debe indicar el alboroto de propaganda que se ha levantado en parte de la prensa extranjera sobre «el cerebro gigante electrónico», sobre los autómatas capaces de revolver cualquier problema y reemplazar el trabajo creador del científico.

*) Desde el punto de vista de la ejecución de operaciones de cómputo.

Adquiere una gran actualidad y agudez, en relación con las circunstancias indicadas, la cuestión sobre las clases del trabajo intelectual que pueden cumplir las computadoras automáticas. Desde un determinado punto de vista esta cuestión se examina y soluciona en la moderna teoría de los algoritmos que es una rama importante de la lógica matemática. Es característico para la lógica matemática el estudio de la esencia de tales nociones como «proceso de cómputo», «demostración matemática», «algoritmo», etc. Ya varios años antes de la creación de las computadoras automáticas electrónicas modernas en la lógica matemática fue elaborada un concepto exacto de «algoritmo» y un esquema general de una computadora automática (la máquina de Turing), también se aclaró la estrecha relación que existe entre los algoritmos y las máquinas. Eso permitió resolver una serie de importantes teoremas que daban luz a la esencia de los procesos que se realizan en las computadoras automáticas; en particular, fue rigurosamente demostrada la existencia de tales problemas para los cuales es imposible su resolución en máquina. El presente libro está dedicado al estudio de la relación entre los algoritmos y las máquinas.

En los §§ 1—3 se explica en una serie de ejemplos lo que es algoritmo y se componen los algoritmos de resolución de problemas matemáticos y lógicos de varias clases.

En los §§ 4—5 se exponen los principios de construcción de las máquinas computadoras electrónicas y de composición de programas o sean los algoritmos adaptados para su realización en máquinas.

Los epígrafes 6—11 están dedicados a una serie de importantes casos de la teoría de los algoritmos. En calidad del concepto básico de la teoría ha sido aceptado el concepto de la máquina de Turing.

Muchas demostraciones son tan voluminosas que no permiten darlas por entero en un libro tan pequeño. Por eso, aquí hay ciertas divergencias de la rigurosidad y de la plenitud de la exposición las que, sin embargo nos parece, no sólo no molestan, sino que, al revés, favorecen a la mejor comprensión de la esencia de la cosa. Para generalizar el cuadro sobre el tema, en el § 6 se reúnen en un resumen algunas cuestiones.

Hagamos una observación más. Se llaman electrónicas a las computadoras modernas de mando automático, puesto

que sus partes principales están construidas con elementos electrónicos. El empleo de la técnica electrónica asegura un gran ahorro de tiempo necesario para realizar las operaciones que cumple la máquina. Sin embargo, la particularidad fundamental de estas máquinas, el *control automático de los procesos* que tienen lugar en ellas, no es precisamente el empleo de la técnica electrónica. Los elementos electrónicos, en un principio, podrían ser reemplazados incluso por mecanismos, o sea, podría crearse una máquina computadora mecánica de control automático capaz de resolver los mismos problemas que la electrónica (pero, claro, mucho más despacio). Así que no se puede concebir que la aparición de las computadoras de esta nueva clase es el resultado del desarrollo solamente de la técnica electrónica. Es más, la primera descripción de una máquina computadora automática (la máquina de Turing, véase el § 7) se dio en la teoría de los algoritmos ya en el año 1936 y se presentó como la descripción de un mecanismo. Las primeras máquinas construidas (1940) fueron electromecánicas.

En el presente libro al describir la construcción de las computadoras no nos concentraremos en los detalles técnicos, fundamentalmente prestaremos la atención al estudio de los principios de interacción de las diferentes partes de la computadora. Este enfoque corresponde al principal fin del libro que consiste en revelar las posibilidades matemáticas y lógicas de las computadoras y no en mostrar el aspecto técnico de la cosa.

§ 1. ALGORITMOS NUMERICOS

El concepto de algoritmo pertenece a las nociones fundamentales de la matemática. Entendemos por *algoritmo* la prescripción exacta sobre el cumplimiento de cierto sistema de operaciones en un orden determinado para la resolución de todos los problemas de algún tipo dado.

Se sobrentiende que la frase anterior no es la definición matemática exacta del concepto de *algoritmo*, esta frase más bien explica el sentido de la palabra «*algoritmo*» aclarando su significado. A pesar de todo esta explicación es comprensible y clara a cada matemático; ella refleja la concepción de algoritmo que espontáneamente se ha formado y empleado en la matemática desde los tiempos antiguos.

Los algoritmos más sencillos son las reglas con las que se cumplen una u otra de las cuatro operaciones aritméticas en el sistema de numeración decimal (el propio término de algoritmo procede del nombre del matemático uzbeko Al-Jwarizmi quien ya en el siglo IX propuso tales reglas). Por ejemplo, la acción de la suma de dos números de varias cifras se descompone en una cadena de operaciones elementales en las que, al realizar cada una de ellas, la persona que hace la cuenta trata solamente con dos cifras de los respectivos sumandos (una de ellas puede tener una marca que indica el traslado de una unidad). Estas operaciones son de dos tipos: 1) anotación de la cifra correspondiente de la suma, 2) marca sobre el traslado por encima de la cifra vecina de la izquierda; aquí la regla prescribe un orden determinado del cumplimiento de estas operaciones (de derecha a izquierda). El carácter formal de estas operaciones elementales consiste en que ellas pueden ser realizadas automáticamente con una tabla de suma de cifras dada una vez para siempre, abstrayéndose por completo del sentido de su contenido.

Con las otras tres operaciones aritméticas la cosa está en forma análoga, lo mismo pasa con el cálculo de la raíz cuadrada y con otras. El carácter formal de las respectivas prescripciones (algoritmos) parece ser que no crea ninguna duda (eso sobre todo se observa cuando los escolares aprenden las reglas para extraer la raíz cuadrada).

Veamos en calidad de ejemplo el *algoritmo de Euclides* que resuelve todos los problemas del tipo siguiente:

Hallar el máximo común divisor de dos números naturales dados a y b.

Es evidente que existen tantos diferentes problemas de este tipo como diferentes pares de números a, b.

Es sabido que la solución de cualquier problema de éstos se puede obtener mediante la composición de una sucesión disminuyente de números de los que el primero será el mayor de los dos dados, el segundo, el menor, el tercero será el resto de la división del primero por el segundo, el cuarto será el resto de la división del segundo por el tercero, etc., hasta que no se haga la división sin resto. El divisor de esta última división será el resultado que se busca.

La división se puede reducir a una sustracción repetida. Basándose en esto se podría presentar una prescripción válida para la resolución de cualesquiera de estos problemas en forma de la siguiente sucesión de indicaciones:

Indicación 1. Examina los dos números a y b. Pasa a la indicación siguiente.

Indicación 2. Compara los dos números ($a = b$, o $a < b$, o $a > b$); pasa a la indicación siguiente.

Indicación 3. Si los números examinados son iguales, cada uno de ellos da el resultado que se busca. El proceso de cómputo se para. Si no es así, pasa a la siguiente indicación.

Indicación 4. Si el primero de los números examinados es menor que el segundo, cámbialos de lugar y continúa su examen. Pasa a la siguiente indicación.

Indicación 5. Resta el segundo de los números examinados del primero y examina dos números: el sustraendo y el resto. Pasa a la indicación 2.

Después de que las cinco indicaciones se hayan cumplido hay que volver de nuevo a la segunda, pasar a la tercera, a la cuarta, a la quinta, y otra vez a la segunda, a la tercera, etc., hasta que se obtengan números iguales, o sea, hasta que se cumpla la condición que se contiene en la tercera indicación; entonces se cesa el proceso.

Es verdad que en la matemática los algoritmos no siempre se expresan de una manera tan formalista; no obstante, a nadie le vendrán dudas sobre la posibilidad de presentar de tal manera formal cualquiera de los algoritmos conocidos.

En la descripción anterior del algoritmo de Euclides figuran en calidad de operaciones elementales en las que se descompone el proceso de resolución del problema, la sustracción de dos números, la comparación de dos números y el

cambio de lugar de dos números. Ahora bien, se puede fácilmente notar que esta descomposición puede ser considerablemenre desarrollada. Por ejemplo, la misma indicación 5 sobre la sustracción de los dos números examinados puede ser desenvuelta en un sistema de indicaciones que describan el algoritmo de sustracción de dos números. Sin embargo, a causa de su gran sencillez y de la costumbre a las reglas de las operaciones aritméticas, en casos similares no se continúa detallando el algoritmo.

Los algoritmos, en concordancia con los cuales la resolución de los problemas planteados se reduce al empleo de las cuatro operaciones aritméticas, se llaman *algoritmos numéricos*. Estos juegan un gran papel en los más variados terrenos tanto de la matemática elemental como de la superior y se presentan corrientemente en forma de prescripciones textuales o de diferentes fórmulas y esquemas. Por ejemplo, el algoritmo de la resolución de un sistema de dos ecuaciones de primer grado con dos incógnitas:

$$a_1x + b_1y = c_1,$$
$$a_2x + b_2y = c_2$$

se da con las fórmulas

$$x = \frac{c_1b_2 - c_2b_1}{a_1b_2 - a_2b_1}; \qquad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1},$$

en las que están completamente expresados tanto la composición de las operaciones como el orden de su ejecución. En las fórmulas anteriores se prevé una misma cadena de operaciones para todos los problemas del tipo dado (o sea, con cualesquiera coeficientes a_1, b_1, c_1, a_2, b_2, c_2). No obstante, tiene interés el observar que, hablando en general, la cantidad de operaciones prescritas por el algoritmo no se conoce de antemano; depende de la elección concreta de las condiciones de cada problema y se aclara solamente durante el proceso de la propia resolución.

En particular, así ocurre en el caso del algoritmo de Euclides, en él el número de sustracciones que pueden hacer falta depende de la elección de uno u otro par de números a, b.

El hecho de que otras muchas operaciones se puedan reducir a las cuatro operaciones aritméticas determina la amplia divulgación de los algoritmos numéricos. Es verdad que esta reducción generalmente no es completamente exacta pero

puede ser realizada con cualquier grado de exactitud establecido de antemano. Todo esto se podría ilustrar en el ejemplo del algoritmo del cálculo de la raíz cuadrada, el que permite extraer la raíz aproximada, pero con cualquier grado de exactitud de antemano establecido, mediante una sucesión compuesta de divisiones, multiplicaciones y sustracciones. En una rama especial de la matemática moderna (el *análisis numérico*) se elaboran procedimientos análogos de reducción a operaciones aritméticas también de otras operaciones más complejas como la integración, la diferenciación, etc.

En la matemática *se considera resuelta una serie de problemas de un determinado tipo cuando se elabora el algoritmo para su resolución.* El objetivo natural de la matemática es la creación de tales algoritmos. Por ejemplo, en el álgebra hay algoritmos que por los coeficientes dados de una ecuación algebraica permiten calcular en forma completamente automática la cantidad de diferentes raíces que tiene esa ecuación (y de qué multiplicidad) y las propias raíces con cualquier grado de exactitud dado de antemano.

Si no se tiene el algoritmo para la resolución de *todos* los problemas del tipo dado, entonces, aunque a veces se consigue resolver alguno que otro problema de este tipo, pero en cada caso aparte se necesita inventar un procedimiento especial que no es utilizable para la mayoría de los demás casos.

Presentaremos un ejemplo de una clase tal de problemas del mismo tipo para la resolución de los cuales la matemática moderna no dispone de algoritmo.

Examinemos todas las posibles ecuaciones de Diofante, o sea, las ecuaciones del tipo

$$P = 0,$$

donde P es un polinomio con coeficientes enteros. De éstas serán, por ejemplo, las ecuaciones

$$x^2 + y^2 - z^2 = 0,$$

$$6x^{18} - x + 3 = 0,$$

de las que la primera tiene tres incógnitas y la segunda, una incógnita (en general se examinan ecuaciones con cualquier número de incógnitas). Una ecuación puede tener solución en números enteros y puede no tenerla. Aquí la primera de

las ecuaciones anteriores tiene solución en números enteros

$$x = 3,\ y = 4,\ z = 5;$$

pero la segunda ecuación no tiene solución en números enteros, para cualquier entero x se establece fácilmente la desigualdad

$$6x^{18} > x - 3.$$

En el año 1901, en el Congreso internacional de matemáticas en París, el famoso matemático alemán David Hilbert dio a conocer una lista de 20 difíciles problemas y agudizó la atención de los círculos matemáticos a la importancia de su resolución. Entre ellos estaba el siguiente problema (el décimo problema de Hilbert): *se exige elaborar un algoritmo que permita aclarar para cualquier ecuación de Diofante si tiene solución en números enteros*

Para el caso particular de las ecuaciones de Diofante que tienen una incógnita, tal algoritmo se conoce ya hace mucho tiempo. Está precisamente establecido que si la ecuación

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0$$

con coeficientes enteros tiene una raíz x_0 entera, pues infaliblemente a_0 se divide por x_0. En concordancia con esto se puede proponer un algoritmo así:

1) hallar todos los divisores del número a_0 (forman un número finito);

2) colocar consecutivamente en primer miembro de la ecuación los divisores hallados y calcular su valor numérico;

3) si al colocar uno de los divisores el primer miembro de la ecuación toma un valor cero, entonces ese divisor es la raíz de la ecuación; si ni para uno de los divisores se convierte primer miembro en cero, entonces la ecuación no tiene raíces en números enteros.

Este problema de Hilbert ha atraído y continúa atrayendo la atención de muchos notables matemáticos pero a pesar de eso el algoritmo necesario para el caso general de una ecuación con dos o muchas incógnitas hasta el día de hoy no se conoce. Además, ahora ya parece bastante verosímil que tal algoritmo en el futuro tampoco será nunca encontrado. El sentido exacto de este pronóstico, a primera vista pesimista, se lo aclarará al lector más tarde, en el material posterior.

En los ejemplos hasta ahora examinados ya resaltan de un modo bastante preciso las siguientes peculiaridades de los algoritmos numéricos, que también son propias de cualquier otro algoritmo. Veámoslas:

La precisión del algoritmo. Se exige que se pueda comunicar el método de cómputo a otra persona en forma de un número finito de indicaciones sobre cómo actuar en cada etapa del cálculo. En concordancia con estas indicaciones el cálculo no depende de la voluntariedad de la persona que lo hace y representa un proceso determinado que puede ser en cualquier momento repetido y cumplido con el mismo éxito por otra persona.

El amplio empleo del algoritmo. El algoritmo es la *única* prescripción que determina el proceso de cálculo que puede comenzar de *diferentes* datos iniciales y lleva en todos los casos al resultado correspondiente. Con otras palabras: el algoritmo no soluciona sólo un problema particular, sino cierta serie de problemas del mismo tipo.

§ 2. ALGORITMOS PARA LA RESOLUCION DE PROBLEMAS LOGICOS

Los problemas examinados en el epígrafe anterior han sido cogidos de la aritmética, del álgebra, de la teoría de los números y son bastante típicos de la problemática de estos terrenos de la matemática. Podríamos decir que ellos tienen un carácter matemático tradicional. Ahora estudiaremos dos clases de problemas que tienen otro carácter; con más acierto se los podría llamar problemas lógicos que problemas matemáticos a pesar de que es muy difícil proponer un criterio suficientemente exacto para distinguir tales problemas *lógicos* de los problemas corrientes *matemáticos*. Sin entrar en discusión sobre la terminología, observaremos que en los dos casos se trata de lo mismo, de elaborar un algoritmo que permita resolver cualquier problema de la clase examinada de problemas del mismo tipo con un solo método. Pero en los casos que estudiamos los algoritmos no serán numéricos.

1. *El juego a los quince.* Una tabla cuadrada está dividida en 16 casillas iguales. En 15 cualesquiera de estas casillas se pueden colocar arbitrariamente (una en cada casilla)

15 fichas numeradas desde el 1 hasta el 15. Asi se forma cierta *posición*. Llamaremos *vecinas* a dos casillas si sus límites contienen un segmento común. Dos posiciones se consideran *contiguas* si de una de ellas se puede crear otra como resultado de una *jugada*. La jugada consiste en el traslado de una ficha a la casilla vacía desde alguna de las casillas vecinas.

El número de todas las posiciones posibles es finito e igual a

$$16! = 20\,922\,789\,888\,000,$$

y en cada posición solamente es posible un número finito de jugadas (2, 3 ó 4).

Surge la serie siguiente de problemas del mismo tipo: *Aclarar para cualesquiera dos posiciones A y B si se puede pasar de una a otra por medio de una sucesión finita de jugadas.*

Si la contestación a la pregunta hecha es afirmativa, la sucesión de jugadas indicada crea una cadena de posiciones:

$$A \leftrightarrow A_1 \leftrightarrow A_2 \leftrightarrow \ldots \leftrightarrow A_n \leftrightarrow B,$$

que lleva de A a B, en la que $\leftrightarrow$ significa la jugada que traspasa la posición a una posición contigua a ella. Se puede considerar que en esta cadena no hay posiciones repetidas, ya que en el caso contrario se podría eliminar todo el trozo de la cadena que se encuentra entre las dos posiciones repetidas y obtener así otra cadena que lleve de A a B de una manera más eficaz. Entonces, si la contestación es afirmativa, el número de jugadas necesarias no será mayor que $16! - 1$. Teniendo en cuenta estas consideraciones se puede ahora formular para el problema planteado el siguiente algoritmo de resolución basado en la simple idea de la selección de todas las posibles combinaciones que tengan 1, 2, 3, ..., $16! - 1$ jugadas. Se compone la lista de las posiciones en la que se incluyen: la posición A, las posiciones contiguas a A, las posiciones contiguas a estas contiguas, etc., y así $16! - 1$ veces. Si en esta lista se encuentra B, la contestación será afirmativa; si no, negativa.

Aunque el método de resolución propuesto exige un enorme volumen de trabajo, debemos dar una contestación afirmativa a la pregunta sobre si poseemos una prescripción exacta que permita con un solo método, como resultado de un número finito de pasos, resolver cualquier problema del

tipo dado. Aquí, evidentemente, tenemos en cuenta la *posibilidad potencial de realizar* (la realización práctica depende de los medios que disponga el que haga los cálculos) el proceso descrito, que a pesar de ser muy largo es finito.

Este algoritmo es muy poco atrayente pero tiene mucha importancia el hecho de que lo hallamos descubierto. Recuérdense que para el problema de Hilbert acerca de las ecuaciones de Diofante no se pudo encontrar *ningún* algoritmo. Al mismo tiempo observaremos que la elaboración de un algoritmo, aunque éste exija mucho trabajo al emplearlo, puede

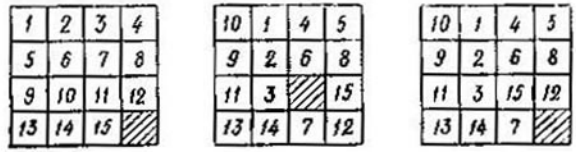


Fig. 1

dar la esperanza de su futuro perfeccionamiento o de la creación de algoritmos más adecuados. En particular en nuestro caso la cosa es precisamente así. Ahora describiremos otro algoritmo que es bastante admisible en el sentido de la facilidad de su uso. Para argumentarlo se necesita un teorema cuyo enunciado citaremos a continuación sin demostrarlo.

Llamaremos *transposición* al cambio de lugares de dos fichas en la tabla. Observemos que de cualesquiera dos posiciones A y B se puede pasar de la una a la otra por medio de traslados y transposiciones; precisamente, si en estas posiciones las casillas vacías no coinciden, entonces, al principio solamente por medio de traslados (no más de 15) se puede pasar de la posición B a la B' desde la misma casilla vacía que en A. Después, el paso desde B' a A se realiza ya sólo con la ayuda de las transposiciones (con no más de 15 transposiciones). La justedad de nuestra afirmación se hace evidente al examinar algún ejemplo. Que vaya el caso sobre las posiciones representadas en la fig. 1.

Trasladamos la ficha 15 y después la 12 con lo que obtenemos la posición B'. Continuando colocamos la ficha N° 1

en el mismo lugar que en A por medio de las transposiciones (1, 10), después realizamos las transposiciones (2, 10), (3, 4), (4, 5), (5, 9), (6, 10), (7, 10), (9, 11), (10, 11), (11, 15). Las fichas 8, 12, 13, 14, 15 se encuentran en sus lugares correspondientes y por eso no participan en las transposiciones.

Citaremos ahora sin demostración el teorema que nos hace falta.

Teorema. *Si se puede pasar de la posición B a la posición A por medio de traslados y de un número par de transposiciones, entonces de B' se puede también pasar a A solamente mediante traslados. Si se puede pasar de B a A por medio de traslados y de un número impar de transposiciones, entonces no se puede pasar de B' a A empleando sólo traslados.*

Ahora ya está claro cómo se crea el algoritmo para el problema planteado: con el procedimiento indicado anteriormente mediante traslados y transposiciones (en total no más de $15 + 15 = 30$ jugadas) pasamos de la posición B a la posición A; si al hacer esto se empleó un número par de transposiciones, la respuesta a la pregunta dada será afirmativa, en el caso contrario, negativa. En nuestro ejemplo el número de transposiciones es par lo que quiere decir que de la posición B realmente se puede pasar a la posición A.

2. *La búsqueda del camino en un laberinto.* La mitología griega narra sobre Teseo, el héroe legendario que tuvo la valentía de penetrar en el laberinto para buscar allá al monstruoso Minotauro y matarlo. Ariadna ayudó a Teseo a salir del laberinto. Ella le dio un ovillo de hilo un cabo del cual lo tenía ella sujeto. Mientras que Teseo se internaba en el laberinto el ovillo se desenrollaba. Después, enrollando el hilo Teseo volvió felizmente a la entrada.

El juguete automático «el ratón en el laberinto» creado por el matemático e ingeniero norteamericano Claude Shannon hizo recordar esta antigua leyenda hace relativamente poco tiempo. En un lugar de un laberinto especial se coloca algo que condicionalmente se llamará un trocito de queso, en otro lugar, «el ratón». El ratón comienza a errar por el laberinto dando rodeos hasta que encuentre el «queso». Si se vuelve a soltar «el ratón» desde el mismo sitio, entonces él ya llega directamente hasta la «comida» sin dar rodeos. Aquí examinaremos cierto problema similar de búsqueda del camino en un laberinto (mejor dicho, una serie de tales problemas del mismo tipo) y describiremos un algoritmo en

concordancia con el cual se deben realizar las búsquedas para conseguir el fin indicado en el problema.

Nos representamos un laberinto en forma de un sistema finito de plazoletas de las que salen corredores cada uno de los cuales une dos plazoletas (tales plazoletas se llamarán contiguas). No se excluye la existencia de plazoletas de las que se puede salir sólo por un corredor (denominemos tales

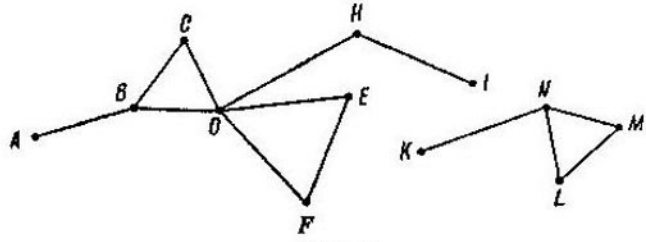


Fig. 2

plazoletas callejones sin salida). Geométricamente se puede representar un laberinto con un sistema de puntos A, B, C, . . . (que representan las plazoletas) y un conjunto de segmentos AB, BC, . . . (que representan los corredores) que unen ciertos pares de puntos (fig. 2).

Diremos que la plazoleta Y es accesible desde la plazoleta X si existe un camino que lleva desde X hasta Y por corredores y plazoletas intermedios. Hablando más exactamente esto quiere decir que bien X e Y son plazoletas contiguas, bien existe una sucesión de plazoletas X_1, X_2, X_3, ..., X_n tales que X y X_1, X_1 y X_2, X_2 y X_3, . . ., etc., y al final X_n e Y son contiguas. Por ejemplo, en la fig. 2 la plazoleta H es accesible desde el callejón A por el camino AB, BC, CD, DE, EF, FD, DH, mientras que la plazoleta K no es accesible desde A. Al mismo tiempo, si Y es en general accesible desde X, ella será también accesible por un *camino simple*, o sea, por un camino tal en el que cada plazoleta (y tanto más cada corredor) se pasan sólo una vez. En el ejemplo anterior el camino no fue simple pero *cortando* el rodeo DE, EF, FD obtendremos el camino simple AB, BC, CD, DH.

Se supone que Minotauro se encuentra en una de las plazoletas del laberinto (la designaremos con la M) y Teseo partiendo en su búsqueda desde la plazoleta A, donde lo espera Ariadna, tiene que resolver el problema siguiente: hay que aclarar si es accesible M desde A o no*). Si es que es accesible, hay que llegar hasta M por cualquier camino pero hay que regresar a donde está Ariadna por un *camino simple*. Si M no es accesible hay que volver a donde está Ariadna.

Puede haber un conjunto innumerable de diferentes laberintos, la disposición mutua de las plazoletas A y M en un laberinto dado también puede variar. Como de antemano Teseo no conoce ni la construcción del laberinto dado, ni el lugar donde se encuentra Minotauro, se supone la resolución del problema planteado en forma de un método general de búsquedas que se pueda emplear para cualquier laberinto y para cualquier disposición de las plazoletas A y M en él. Con otras palabras, se supone que la resolución será un algoritmo que solucionará cualquier problema del tipo dado.

Para crear tal algoritmo estudiaremos un método especial de búsqueda. En cualquier fase del proceso de búsqueda hay que diferenciar en concordancia con este método los corredores que Teseo no ha pasado ni una vez (condicionalmente, los verdes), los que ha pasado una vez (los amarillos) y los que ha pasado dos veces (los rojos). Entonces Teseo, al encontrarse en alguna de las plazoletas, puede llegar a una de las plazoletas contiguas por medio de uno de los dos pasos siguientes:

1. *Desenrollamiento del hilo.* Paso de la plazoleta dada por cualquier corredor verde hasta la plazoleta contigua. (En este caso el hilo de Ariadna se desenrolla a lo largo de este corredor el que después de haber pasado por él ya se considera amarillo).

2. *Enrollamiento del hilo.* Regreso de la plazoleta dada por el último corredor amarillo pasado hasta la plazoleta contigua. Entonces el hilo de Ariadna que se desenrolló antes a lo largo de este corredor se enrolla de nuevo y este corredor se anuncia rojo.

Se supone que Teseo hace algunas marcas que le permiten después distinguir los corredores rojos de los verdes; los amarillos se distinguen, puesto que en ellos está tendido el

*) Es natural considerar que $M \neq A$.

hilo de Ariadna. La elección de un paso u otro depende de la situación que observa Teseo en la plazoleta en donde se encuentra en ese momento. Esa situación puede caracterizarse con uno o varios de los siguientes indicios:

1. *Minotauro.* Se ha descubierto que Minotauro está en la plazoleta dada.

2. *Rodeo.* Por la plazoleta dada ya está extendido el hilo de Ariadna; con otras palabras, de la plazoleta dada salen por lo menos dos corredores amarillos.

3. *Calle verde.* De la plazoleta dada hay salida por lo menos a un corredor verde.

4. *Ariadna.* En la plazoleta dada se encuentra Ariadna.

5. *Caso quinto.* Carencia de todos los indicios anteriores.

Nuestro método de búsqueda puede ser ahora presentado con el esquema siguiente:

Indicio	*Paso*
1. *Minotauro*	Parada
2. *Rodeo*	Enrollamiento del hilo
3. *Calle verde*	Desenrollamiento del hilo
4. *Ariadna*	Parada
5. *Caso quinto*	Enrollamiento del hilo

Al encontrarse en alguna de las plazoletas Teseo decide el paso inmediato de la siguiente forma: él comprueba según el orden de los números, en concordancia con la columna izquierda del esquema, cuál de los indicios ennumerados tiene lugar; al ver tal indicio él (sin comprobar ya los demás) hace el paso correspondiente (o la parada) de la columna derecha. Tales pasos se hacen hasta que llegue la parada.

La utilidad del método propuesto se deduce directamente de las tres afirmaciones siguientes:

1. Desde cualquier disposición mutua de A y M en el laberinto, después de un número finito de pasos, infaliblemente se llegará a la parada que será bien en la plazoleta de Minotauro, bien en la plazoleta de Ariadna.

2. Si la parada tiene lugar en la plazoleta de Minotauro, entonces Minotauro es accesible. Más aún, en este caso resulta que el hilo de Ariadna está extendido por un camino simple que lleva desde A hasta M; enrollando el hilo, Teseo ahora podrá volver hasta donde está Ariadna.

3. Si la parada ha sucedido en la plazoleta de Ariadna, entonces Minotauro es *inaccesible.*

Antes de demostrar estas afirmaciones enseñaremos en dos ejemplos cómo funciona el método propuesto.

Ejemplo 1. Supongamos que la búsqueda de Minotauro, que se encuentra en F, comienza desde la plazoleta A del laberinto (fig. 2). Es cómodo representar el proceso de búsqueda que corresponde a nuestro método por medio del esquema de la tabla 1 (por motivo de arbitrariedad en la elección de cada corredor verde éste es uno de los posibles esquemas).

Tabla 1

Número de orden del paso	Indicio por el que se conduce Teseo	Paso	Corredor pasado	Color del corredor después de pasar por él
1	*Calle verde*	Desenrollamiento del hilo	*AB*	Amarillo
2	» »	»	*BC*	»
3	» »	»	*CD*	»
4	» »	»	*DH*	»
5	» »	»	*HJ*	»
6	*Caso quinto*	Enrollamiento del hilo	*JH*	Rojo
7	» »	»	*HD*	
8	*Calle verde*	Desenrollamiento del hilo	*DB*	Amarillo
9	*Rodeo*	Enrollamiento del hilo	*BD*	Rojo
10	*Calle verde*	Desenrollamiento del hilo	*DF*	Amarillo
11	*Minotauro*	Parada	....	.....

Vemos que en este caso Minotauro es accesible. Seleccionando en la penúltima columna aquellos corredores que se han quedado amarillos (a base de las indicaciones de la última columna) obtendremos el siguiente camino simple que lleva de A a F:

$$AB,\ BC,\ CD,\ DF.$$

Ejemplo 2. Si la búsqueda comienza desde la plazoleta K, entonces se puede representar el proceso de búsqueda con el esquema de la tabla 2.

Vemos que en este caso Minotauro es inaccesible.

Tabla 2

Número de orden del paso	Indicio por el que se conduce Teseo	Paso	Corredor pasado	Color del corredor después de pasar por él
1	*Calle verde*	Desenrollamiento del hilo	*KN*	Amarillo
2	» »	»	*NL*	»
3	» »	»	*LM*	»
4	» »	»	*MN*	»
5	*Rodeo*	Enrollamiento del hilo	*NM*	Rojo
6	*Caso quinto*	»	*ML*	»
7	» »	»	*LN*	»
8	» »	»	*NK*	»
9	*Ariadna*	Parada	....	.....

Pasemos ahora a la demostración de las afirmaciones 1—3.

Demostración de la afirmación 1. Previamente mostremos con el método de inducción, partiendo del número de pasos de Teseo, que en cualquier fase del proceso de búsqueda surge la siguiente alternativa (o sea, tiene lugar uno de los dos siguientes casos que se excluyen mutuamente):

a) En el labernito no hay corredores amarillos; en este caso Teseo, se encuentra en la plazoleta *A* (Ariadna).

b) En el laberinto hay corredores amarillos y éstos, siendo examinados en el mismo orden en que Teseo pasó por ellos, forman un camino que lleva desde *A* hasta la plazoleta en la cual se encuentra Teseo.

Junto con esto se aclarará que Teseo nunca pasa por un corredor rojo.

Las tesis expresadas son evidentes para el comienzo del proceso, cuando Teseo todavía se encuentra en la plazoleta *A* y todavía no ha pasado por ningún corredor (todos los corredores tienen el color verde). Supongamos ahora que las alternativas indicadas son justas después del $(n-1)$-ésimo paso y mostremos que ellas entonces serán justas también después del n-ésimo paso (claro está, sólo si el $(n-1)$-ésimo paso todavía no ha llevado a la parada).

Supongamos que después del $(n-1)$-ésimo paso tiene lugar el caso *a*. Entonces el paso inmediato puede ser el

avance por un corredor verde desde A hasta cierta plazoleta contigua K (después del n-ésimo paso surge el caso b con el único corredor amarillo AK) o la parada en la plazoleta A (después del n-ésimo paso se conserva el caso a).

Supongamos ahora que después del $(n-1)$-ésimo paso tiene lugar el caso b con s corredores amarillos que forman el camino $AA_1, A_1A_2, \ldots, A_{s-1}K$. En dependencia del indicio que toma Teseo para determinar su conducta en la elección del inmediato n-ésimo paso, después del n-ésimo paso se tienen las posibilidades siguientes:

1. *Minotauro.* Tiene lugar la parada en la plazoleta K con la conservación de los corredores amarillos anteriores (caso b después del n-ésimo paso).

2. *Rodeo.* Teseo enrolla el hilo, o sea, se retira por el corredor amarillo KA_{s-1} que ahora ya se hace rojo. El camino amarillo se acorta en un corredor; si el número s de corredores en el camino anterior era mayor que 1, entonces después del n-ésimo paso tendrá lugar el caso b con $(s-1)$ corredores amarillos; si $s = 1$, entonces tendrá lugar el caso a.

3. *Calle verde.* Teseo desenrolla el hilo, es decir, avanza por un corredor verde que ahora ya se hace amarillo. Tiene lugar el caso b con $s+1$ corredores amarillos.

4. *Ariadna.* Teseo no se conducirá por este indicio, pues, si incluso él volviese a la plazoleta de Ariadna yendo por el camino amarillo

$$AA_1, A_1A_2, \ldots, A_{s-1}K$$

(o sea, si $K = A$), en concordancia con el método de búsqueda aceptado, él debería conducirse por el indicio *rodeo*.

5. Al no revelarse ninguno de los primeros cuatro indicios tiene lugar el enrollamiento del hilo que lleva, lo mismo que en el caso del rodeo, al caso a (cuando $s = 1$) o al caso b (cuando $s > 1$)

De esta manera ha sido por completo establecida la alternativa de que se hablaba anteriormente. También se ha aclarado que Teseo por cada corredor pasa no más de dos veces (por uno rojo no pasa nunca). Además, como el número de todos los corredores del laberinto es finito, la sucesión de los pasos también tiene que ser finita. Esta sucesión puede acabarse solamente con la parada de Teseo en la plazoleta de Minotauro o en la plazoleta de Ariadna.

Demostración de la afirmación 2. En el caso de la parada en la plazoleta de Minotauro el hecho de la posibilidad de acceso es evidente y el hilo de Ariadna estará extendido por el camino amarillo la existencia del cual se ha establecido anteriormente. La carencia de rodeos en el hilo está garantizada ya que cada vez que Teseo, al errar por el laberinto, ve un rodeo, vuelve por él y así lo suprime.

Demostración de la afirmación 3. Observemos que en el caso de parada en la plazoleta de Ariadna

1) cada corredor del laberinto o ha sido pasado dos veces (corredor rojo), o no ha sido pasado ninguna (corredor verde); diciéndolo con otras palabras, todo el hilo está enrollado (en el laberinto no hay corredores amarillos)*);

2) todos los corredores que salen de A son rojos, pues, en concordancia con la condición admitida, el indicio *Ariadna* se toma en cuenta solamente en el caso de que los tres indicios que le preceden en el esquema, entre los cuales se encuentra la *calle verde*, no tengan lugar.

Supongamos ahora a la inversa que Minotauro es accesible y que sea AA_1, A_1A_2, . . ., A_nM el camino que lleva de A a M. El primer corredor en este camino es rojo pues él sale de A, el último es verde puesto que Teseo no ha encontrado todavía a Minotauro. Que sea A_iA_{i+1} el primer corredor verde en esta sucesión. Así que en A_i convergen corredores verdes y rojos. Examinemos ahora el último paso de Teseo por la plazoleta A_i; éste, evidentemente, tuvo lugar por uno de los corredores rojos adyacentes a A_i y solamente pudo ocurrir al enrollar el hilo ya que fue un paso *repetido* por ese corredor. Su motivo, por consiguiente, fue: bien el haber un rodeo, bien el quinto caso, o sea, la carencia de todos los cuatro indicios. No obstante, el último no puede tener lugar puesto que de A_i sale el corredor verde A_iA_{i-1}. Tenemos por eso que admitir que el último paso por A_i estuvo relacionado con la aparición de un rodeo. Ahora bien, esto inmediatamente nos lleva a una contradicción con la que se termina la demostración de la afirmación 3. Esto es así, puesto que si en A_i hubiese aparecido un rodeo, eso significaría que de ella salen por lo menos dos corredores amarillos; al hacer el paso siguiente Teseo habría convertido

*) Se propone al lector que haga la demostración de estos hechos.

uno de estos corredores de amarillo a rojo, el corredor amarillo que quedase obligatoriamente tendría que haber sido más tarde pasado de nuevo ya que no tienen que quedar corredores amarillos, entonces pasará otra vez por la plazoleta A_i. Esto contradice a la admisión de que se examina el último paso por A_i.

Haremos ahora la siguiente observación. En la prescripción propuesta por nosotros para la búsqueda del camino se admite cierta eventualidad. En las condiciones de la *calle verde* el paso inmediato no se determina de un modo unívoco, puesto que de la plazoleta pueden haber salidas a varios corredores verdes y nuestra prescripción no establece cuál de ellos se debe elegir, o diciéndolo más exactamente, admite la elección arbitraria de uno de ellos. Con esto se perturba la calidad de precisión sobre la cual en el epígrafe anterior se decía que era propia de todos los algoritmos. No obstante, se puede fácilmente eliminar este elemento de casualidad (y así mismo convertir la predicción examinada en algoritmo) aunque sea con el acuerdo de que cuando haya varios corredores verdes siempre se elige uno de ellos a base de cierta regla; por ejemplo, al entrar en cada plazoleta Teseo la recorre en la dirección de las agujas del reloj hasta que aparece el primer corredor verde por el cual continúa su camino, o a base de algún otro acuerdo. El estudio de tales prescripciones en las que de antemano se prevén actos de elección casual tiene un gran interés teórico y práctico, sobre todo en la moderna teoría de los juegos. Nosotros no vamos a tocar esta cuestión, estudiaremos sólo los procesos de estricto carácter determinativo.

Al concluir, es provechoso comparar los dos problemas de este epígrafe y revelar cierta relación entre ellos. A pesar de su aparente diferencia estos problemas en su esencia son muy semejantes, mejor dicho, el primero es un caso particular del segundo. Efectivamente, si cada posición del juego «15» la comparamos con una plazoleta y cada paso de una posición a la contigua, con un corredor que une dos plazoletas, entonces el primer problema de este epígrafe se convierte en el problema de búsqueda del camino en un laberinto de forma particular que tenga 16! plazoletas cada una de las cuales se comunique por corredores con dos, tres o cuatro plazoletas contiguas. Y entonces el algoritmo de búsqueda del camino que hemos propuesto se puede emplear en

el juego «15». Se comprende con facilidad que él sencillamente representa cierta variante perfeccionada del algoritmo de selección; el perfeccionamiento consiste en que al seleccionar las posiciones y los pasos no se realizan más de dos repeticiones.

Para un laberinto de caso particular (que corresponde al juego «15») se ha logrado encontrar un algoritmo más sencillo.

Al mismo tiempo, será completamente natural suponer que para el caso general, en el cual el algoritmo se puede aplicar a cualquier laberinto, él no puede ser nada más que una cierta clase de selección. Por eso seguramente no debe esperarse la creación de un algoritmo más sencillo que el que hemos propuesto.

§ 3. EL PROBLEMA DE LAS PALABRAS

El problema de las palabras representa una generalización ampliada del juego «15» y de las búsquedas de Teseo. Si el juego «15» se reduce a búsquedas en un laberinto especial que tiene una forma dada *fija* y *finita*, y las búsquedas de Teseo se pueden realizar en un laberinto *finito cualquiera*, el problema de las palabras es en cierto sentido un problema de búsqueda en un laberinto *infinito*. El problema de las palabras surgió en unos apartados del álgebra moderna denominados *teoría de los sistemas asociativos y teoría de los grupos*, sin embargo, su importancia sale de los límites de estas teorías especiales. Diferentes variantes de este problema fueron investigados con éxito por los eminentes matemáticos soviéticos Andréi Andréievich Márkov (nacido en 1903) y Piotr Serguéievich Nóvikov (1901—1975) y por sus discípulos.

Para comenzar introduzcamos ciertos conceptos preliminares:

Llamaremos *alfabeto* a cualquier sistema finito de signos diferentes entre sí que se denominan letras de este alfabeto. Se puede poner el ejemplo del alfabeto $\{\alpha, \text{ы}, z, ?\}$ compuesto de la letra griega α, de la rusa ы, de la latina z y del signo de interrogación. Se denomina *palabra* en un alfabeto dado a cualquier sucesión de letras de este alfabeto. Por ejemplo, *abaa* y *bbac* son palabras en el alfabeto $\{a, b, c\}$.

Si la palabra L es parte de la palabra M lo trataremos como la *entrada* de la palabra L en la palabra M. Por ejemplo, en la palabra *abcbc bab* hay dos entradas de la palabra *bcb*, una de ellas comenzando de la segunda letra, la otra, de la cuarta. Vamos a estudiar las transformaciones de unas palabras en otras por medio de ciertas sustituciones admisibles que se expresan en forma de

$$P - Q \text{ o } P \to Q,$$

donde P y Q son dos palabras en un mismo alfabeto.

El empleo de una *sustitución orientada* $P \to Q$ a la palabra R es posible en el caso cuando en ella hay aunque sea una sola entrada de la *parte izquierda* P; ello consiste en la sustitución de una entrada tal cualquiera por la correspondiente parte derecha Q. El empleo de la *sustitución no orientada* $P - Q$ permite tanto el cambio de la entrada de la parte izquierda por la derecha, como el cambio de la entrada de la parte derecha por la izquierda. Comenzando desde este lugar estudiaremos sobre todo las sustituciones no orientadas y allí, donde esto no trae confusión, las llamaremos sencillamente sustituciones.

Ejemplo 1. La sustitución *ab — bcb* se puede aplicar a la palabra *abcbcbab* de cuatro maneras; el cambio de cada una de las dos entradas de *bcb* trae las palabras:

$$a\,\underline{ab}\,cb,\ ab,\ abc\,\underline{ab}\,ab\ ,$$

y el cambio de cada una de las dos entradas de *ab* trae las palabras:

$$\underline{bcb}\ cbc\ bab,\ abc\ bcb\ \underline{bcb}.$$

Empero, esta sustitución no es aplicable a la palabra *bacb*.

Acordaremos llamar *cálculo asociativo* al conjunto de todas las palabras formadas con un cierto alfabeto junto con algún sistema finito de sustituciones admisibles.

La indicación del alfabeto correspondiente y del sistema de sustituciones es suficiente para determinar un cálculo asociativo.

Si la palabra R puede ser transformada en la palabra S por medio de una sola sustitución admisible, entonces S también puede ser transformada en R de la misma manera; en este caso denominaremos a S y R palabras *contiguas*. Llamaremos *cadena deductiva*, que lleva de R_1 a R_n, a la

secuencia de palabras

$$R_1, R_2, \ldots, R_{n-1}, R_n$$

tales que R_1 y R_2 son contiguas, R_2 y R_3 son contiguas, ...R_{n-1} y R_n son contiguas. Si existe una cadena deductiva que lleva de la palabra R a la palabra S, entonces es evidente que existirá también una cadena deductiva que lleva de S a R; en tal caso denominaremos estas palabras *equivalentes* y designaremos eso así: $R \sim S$. También estará claro que si $S \sim R$ y $R \sim T$, entonces $S \sim T$. En lo sucesivo necesitaremos además aplicar el teorema siguiente:

Teorema. *Que sean $P \sim Q$; entonces si en alguna palabra R existe la entrada de P, como resultado de la sustitución de ella con Q se recibirá una palabra equivalente a R.*

Demostración. Para el teorema es más evidente representar la palabra R con la designación SPT, en la que S es la parte de la palabra R anterior a la entrada de P, y T es la parte de la palabra que la sigue; entonces la representación de la palabra ya transformada será SQT. Debido a la equivalencia $P \sim Q$ existe una cadena deductiva

$$P, P_1P_2, \ldots, P_m, Q.$$

En este caso, como se ve fácilmente, la secuencia de palabras

$$SPT, SP_1T, SP_2T, \ldots, SP_mT, SQT$$

es una cadena deductiva que lleva de SPT (o sea, de R) a SQT (o sea, a la palabra transformada). El teorema está demostrado.

Ejemplo 2. Veamos el cálculo asociativo que fue estudiado por G. S. Tseitin.

El alfabeto se compone de

$$\{a, b, c, d, e\}.$$

En el sistema de sustituciones admisibles entran:

$$ac \to ca$$
$$ad \to da$$
$$bc \to cb$$
$$bd \to db$$
$$abac \to abace$$
$$eca \to ae$$
$$edb \to be$$

En este cálculo a la palabra *abcde* se le puede aplicar sólo la tercera sustitución y ella tiene sólo una palabra contigua *acbde*. Al continuar tiene lugar la equivalencia

$$abcde \sim cadedb,$$

lo que se deduce de la cadena deductiva siguiente:

$$\underline{abcde}, \underline{acbde}, \underline{cabde}, \underline{cadbe}, cadedb.$$

A la palabra *aaabb*, por ejemplo, no le es aplicable ninguna de las sustituciones y por eso no tiene palabras contiguas; además, no existen palabras diferentes de *aaabb* que le sean equivalentes.

En cada cálculo asociativo surge su *problema* especial *de la equivalencia de las palabras*. Este consiste en lo siguiente:

Para cualesquiera dos palabras en un cálculo dado hay que comprobar si son equivalentes o no.

Como en cualquier cálculo se encontrará una cantidad innumerable de diferentes palabras, pues de hecho aquí hay una serie infinita de problemas del mismo tipo. La solución se representa en forma de un algoritmo que distinga la equivalencia o la no equivalencia de cualquier par de palabras.

Puede darse la impresión de que el problema de las palabras representa un rompecabezas sin sentido y, por consiguiente, que la búsqueda de tal algoritmo no tiene ningún interés especial práctico o teórico. En realidad eso no es así; se puede mostrar que este problema tiene un origen natural y, además, una gran importancia teórica y práctica que por completo justifica los esfuerzos dirigidos a la creación del algoritmo correspondiente. No obstante, en esta etapa de nuestra exposición nos abstendremos por el momento a discutir esta cuestión en su esencia y pasaremos al examen de ciertos hechos concretos.

Ante todo indicaremos la relación entre el problema de la equivalencia de las palabras y el problema de Teseo. Si se construyese para cada palabra su «plazoleta» y para cada par de palabras contiguas un corredor que uniese las plazoletas correspondientes, entonces el cálculo asociativo se presentaría ante nosotros en forma de un laberinto con un número infinito de plazoletas y corredores en el que de cada plazoleta sale siempre un número finito de corredores (es posible que haya también tales plazoletas de las cuales no

sale ni un solo corredor; véase la palabra *aaabb* en el ejemplo 2). Aquí la cadena deductiva que lleva de una palabra cualquiera R a la palabra Q representará el camino en el laberinto que lleva de una plazoleta a otra; vale decir que la equivalencia de las palabras corresponde en este caso a la accesión mutua de una plazoleta desde otra. Por último, al darle esta interpretación el mismo problema de las palabras se convierte en el problema de búsqueda del camino en un laberinto infinito.

Para aclarar mejor la especificidad de las dificultades que aquí surgen, veamos preliminarmente un *problema de palabras limitado* que consiste en lo siguiente:

Para cualesquiera dos palabras R y T en los límites de un cálculo asociativo dado hay que determinar si se puede transformar una en otra por medio de la aplicación sucesiva no más de k veces de sustituciones admisibles (k es un número arbitrario, pero fijado y natural).

Con un planteamiento tal el problema del algoritmo se resuelve fácilmente: se puede precisamente emplear el algoritmo de selección que ya conocemos. Con este algoritmo se examina la lista de todas las palabras comenzando por la palabra R, pasando por todas sus palabras contiguas, después, por las contiguas de las contiguas y así k veces. La contestación a la pregunta dada será positiva o negativa en dependencia de si aparece o no la palabra T en esa lista.

Empero, si volvemos al problema de las palabras ilimitado, veremos allí un fenómeno completamente diferente. Como la longitud de la cadena deductiva que lleva de R a T (si tal existe) puede resultar de cualquier gran dimensión, pues, hablando en general, de antemano no se sabe cuándo se debe contar acabado el proceso de selección. Supongamos, por ejemplo, que hemos continuado ya el proceso de selección hasta $10^{20} = 100\,000\,000\,000\,000\,000\,000$ y ya tenemos la lista de todas las palabras que se pueden obtener de R con ayuda del empleo reiterado de las sustituciones, el número total de las cuales no pasa de 10^{20}. Supongamos también que en esta lista la palabra T no ha aparecido. ¿Nos da eso algún fundamento para deducir que las palabras R y T no son equivalentes? Está claro que no, puesto que no se excluye la posibilidad de que R y T sean equivalentes pero la cadena deductiva mínima que las une sea todavía más larga.

Para obtener los resultados deseables, aquí habrá que renunciar a la selección simple, aquí es necesario utilizar otras ideas basadas en el análisis del propio procedimiento de transformación de unas palabras en otras por medio de las sustituciones admisibles. Haremos la prueba, por ejemplo, de aclarar si las palabras *abaacd* y *acbdad* son equivalentes en los cálculos de Tseitin (véase el ejemplo 2). La contestación negativa a esto se deduce de los siguientes razonamientos: en cada una de las sustituciones admisibles de este cálculo las partes izquierda y derecha contienen un mismo número de entradas de la letra *a* (o no contienen esta letra); por eso, en cualquier cadena deductiva todas las palabras tienen que contener el mismo número de entradas de la letra *a*. Como en las dos palabras propuestas el número de entradas de la letra *a* no es el mismo, estas palabras no son equivalentes.

El hallazgo de semejantes *invariantes deductivas*, o sea, de las propiedades que no cambian para todas las palabras de una cadena deductiva permite en ciertos casos encontrar los algoritmos resolutivos que se buscan.

Ejemplo 3.

El alfabeto es

$$\{a,\ b,\ c,\ d,\ e\}.$$

El sistema de sustituciones admisibles es

$$\begin{array}{llll} ab - ba; & ae - ea; & be - eb; & \partial e - e\partial; \\ ac - ca; & bc - cb; & c\partial - \partial c; & \\ a\partial - \partial a: & b\partial - \partial b; & ce - ec. & \end{array}$$

Estas sustituciones admisibles no cambian la cantidad de entradas de cada letra en la palabra, cambia sólo el orden de las letras en la palabra. Se ve palpablemente que dos palabras son equivalentes en aquel, y sólo en aquel caso en el cual en ellas es igual el número de entradas de cada letra. El algoritmo de la indagación de la equivalencia es por eso muy simple y se reduce al cálculo del número de entradas de las letras en cada una de estas palabras y a la comparación de estos números.

Más adelante examinaremos con detalle un ejemplo más complicado, pero de antemano convendremos en la siguiente generalización de los conceptos «palabra» y «sustitución admisible». Además de las palabras corrientes en el alfabeto

dado precisamente veremos también las *palabras vacías* que no contienen ni una letra y las designaremos por Λ. Al mismo tiempo admitiremos las sustituciones del tipo siguiente:

$$P - \Lambda.$$

Al hacer esto, el cambio de la parte izquierda por una palabra vacía significa simplemente que de la palabra transformada se excluye la entrada de la palabra P. El cambio de la parte derecha por la izquierda significa que entre dos letras cualesquiera de la palabra transformada o al principio de ella, o al final de ella se coloca la palabra P.

Ejemplo 4. Se da un cálculo asociativo en el alfabeto $\{a, b, c\}$ con el sistema de sustituciones:

$$b - acc$$
$$ca - accc$$
$$aa - \Lambda$$
$$bb - \Lambda$$
$$cccc - \Lambda.$$

Hay que hallar el algoritmo resolutivo para el problema de la equivalencia de las palabras en este cálculo.

Crearemos un algoritmo auxiliar, el algoritmo de reducción (o transformación) que indica para cualquier palabra su palabra equivalente pero de un tipo especial, o sea, la palabra reducida. Para eso examinaremos un sistema ordenado de sustituciones orientadas:

$$b \to acc$$
$$ca \to accc$$
$$aa \to \Lambda$$
$$cccc \to \Lambda$$

y nos pondremos de acuerdo en que al aplicar el algoritmo a cualquier palabra, él funciona de la manera siguiente: se elige de esta lista la primera, por orden, sustitución orientada aplicable a la palabra R; al haber varias entradas de su parte izquierda en la palabra R se aplica la sustitución a la primera entrada que se encuentra más a la izquierda; para obtener de esta misma manera la palabra R', de nuevo se elige la primera sustitución de la lista que le sea aplicable, y

su parte izquierda se aplica a la entrada que se encuentra más a la izquierda; si después de un número finito de tales pasos se obtiene la palabra S a la que ya no le es aplicable ni una de estas sustituciones, entonces se considera que el algoritmo es aplicable a la palabra R y puede transformarla en la palabra S*).

Mostraremos que el algoritmo de reducción es aplicable a cualquiera que sea palabra R y la transforma en una de las ocho siguientes palabras (palabras ya reducidas):

$$\Lambda,\ c,\ cc,\ ccc,\ a,\ ac,\ acc,\ accc.$$

En efecto, si en la palabra R hay entradas de la letra b, entonces al principio se aplicará la primera sustitución que *excluye* b y la cambia por acc. Esto tendrá lugar hasta que no queda ni una b. A continuación funcionará la segunda sustitución que traslada la letra a a la izquierda de la letra c hasta que no quede ni una letra a que esté inmediatamente a la derecha de la letra c, diciéndolo con otras palabras, hasta que todas las a resulten delante de todas las c. Por último, comenzará el proceso de exclusión de cada dos a vecinas y de cada cuatro c vecinas hasta que no aparezca la palabra que contenga no más de una a y no más de tres c. Pueden haber sólo ocho palabras tales que son las indicadas anteriormente.

Es evidente que cada palabra es equivalente a su palabra reducida, por eso dos palabras son equivalentes en el caso y sólo en el caso de que les corresponde una misma palabra reducida o bien dos palabras reducidas equivalentes. Más adelante demostraremos que cada dos de todas las ocho palabras reducidas no son equivalentes entre sí. De esto se deducirá que dos palabras son equivalentes en el caso y sólo en el caso de que les corresponda una misma palabra reducida. Al mismo tiempo será creado también un algoritmo para el problema de las palabras planteado. Este consistirá en la aplicación del algoritmo de reducción a cada una de las dos palabras estudiadas y en la comparación de las palabras reducidas que se han obtenido.

*) Los algoritmos para la transformación de palabras en cierto alfabeto que se dan de la manera indicada por medio de algún sistema ordenado de sustituciones orientadas se llaman normales. La teoría de los algoritmos normales que tiene un gran interés teórico y práctico, ha sido creada por el miembro correspondiente de la Academia de ciencias de la URSS A. A. Márkov y está expuesta en su libro «La teoría de los algoritmos».

Supongamos, por ejemplo, que se dan las palabras *cacb* y *bb*. Encontramos las palabras reducidas:

1)*cacb*, *cacacc*, *acccacc*, *acccacccccc*, *accacccccccc*,

acacccccccccc, *aacccccccccccccc*, *ccccccccccccc*, *ccccccccc*, *cccccc*, *cc*;

2)*bb*, *accb*, *accacc*, *acacccccc*, *aaccccccccc*, *cccccccc*, *cccc*, $\wedge$.

La conclusión es que las palabras *cacb* y *bb* no son equivalentes puesto que han sido obtenidas dos palabras reducidas diferentes: *cc* y $\wedge$.

Demostración de la no equivalencia mutua de las ocho palabras indicadas. Advertiremos ante todo que si tenemos una cadena deductiva que lleva de alguna palabra R que no contiene la letra *b* a la palabra S que tampoco la contiene, se puede sacar de ella una cadena deductiva que lleve de R a S tal que en las palabras intermedias de la cadena no se encuentre la letra *b*. Efectivamente, si en todas las palabras de la cadena deductiva dada cambiamos cada entrada de la letra *b* por la palabra *acc*, obtendremos una sucesión de palabras en la que cada dos palabras vecinas serán contiguas (en el sentido de su composición) o sencillamente iguales. Si se eliminan ahora las palabras que se repiten (situadas una al lado de otra) y que están de más, se obtendrá la cadena deductiva necesaria. En las cadenas deductivas de este tipo no participa la sustitución *b—acc*.

Continuando, en cada una de las sustituciones admisibles que quedan, el número de entradas de la letra *a* en la parte izquierda y en la parte derecha son simultáneamente par o simultáneamente impar. Una afirmación análoga también sería justa para la letra *c*. Esto quiere decir que la paridad del número de entradas de la letra *a* (o de la letra *c*) es una invariante deductiva para las cadenas deductivas del tipo indicado. En el acto seguido de aquí se deduce que ni una de las cuatro palabras indicadas que contienen una entrada de la letra *a* no es equivalente ni a una de las cuatro palabras predichas que no contienen tales entradas en absoluto. Asimismo ni una de las palabras indicadas que contienen una o tres entradas de la letra *c* no son equivalentes a alguna palabra que contenga dos entradas tales o que no contenga ninguna. Ahora nos queda convencernos de la no equivalen-

cia de los siguientes pares de palabras:

$$\Lambda,\ cc;\ c,\ ccc;\ a,\ acc;\ ac,\ accc.$$

Si tuviese lugar la equivalencia aunque sea en uno de los tres primeros pares, entonces, como consecuencia del teorema de este epígrafe, tendría también lugar la equivalencia del cuarto. Es por eso suficiente establecer la no equivalencia del par *ac*, *acc*; lo que haremos ahora.

Introduciremos los términos siguientes.

Denominaremos *índice* de una *entrada de la letra a* en la palabra *R* al número de todas las entradas de la letra *c* que se encuentran a la derecha de esta entrada de la letra *a*. *Indice de la palabra R* se llama a la suma de los índices de todas las entradas de la letra *a**). Cada una de las sustituciones $aa \to \Lambda$ y $cccc \to \Lambda$ no cambia la paridad del índice de la palabra. Efectivamente, al sustituir *aa* en lugar de una palabra vacía, el índice de la palabra aumenta en la suma de los índices iguales entre sí de estas dos entradas de la letra *a*, o sea, en un número par; al cambiar la entrada *aa* por una palabra vacía, el índice de la palabra disminuye en un número par.

Al sustituir *cccc*, los índices de ciertas entradas de *a* aumentan en cuatro, los índices de otras no cambian; en total, el índice de la palabra aumenta en un número par. Análogicamente ocurre al tachar *cccc*. Es evidente que la sustitución $b \to acc$ no cambia la paridad del índice de la palabra.

Y por último, mostremos que la sustitución $ca \to acc$ cambia la paridad del índice de la palabra. Comparemos las palabras

$$PcaQ \text{ y } PacccQ;$$

el índice de cada entrada de *a* en la parte *P* de la palabra *R* cambia justo en 2, pero el índice de la entrada de *a* en *Q* no cambia. El índice de la única entrada de *a* entre *P* y *Q* cambia exactamente en 3. En total el índice de la palabra cambia en un número impar.

Las dos palabras *ac* y *accc* tienen índices de una misma paridad, por eso, si son equivalentes, en la cadena deductiva

*) Por ejemplo, en la palabra *acbca* el índice de la primera entrada de la izquierda de la letra *a* es cero (a la derecha no hay entradas de la letra *c*); la segunda entrada de la izquierda de la letra *a* tiene el índice 2. El índice de la palabra es 2

correspondiente (se puede contar que en ella no se encontrará la letra *b*, véase consideraciones anteriores) puede solamente haber un número par de sustituciones *ca* — *accc*.

Empero, tal conjetura, como se verá en las reflexiones siguientes, lleva a contradicción. Cada aplicación de la sustitución *ca* — *accc* cambia el número de entradas de la letra *c* en 2, por eso, un número par de sus aplicaciones cambia el número de entradas de la letra *c* en un número múltiple de cuatro. Es evidente que la sustitución *cccc* — Λ cambia el número de entradas de la letra *c* justo en cuatro, y la sustitución *aa* — Λ en absoluto no cambia este número. Haciendo el resumen de todo lo dicho debemos concluir que si $ac \sim \sim accc$, entonces la cantidad de entradas de la letra *c* en ellos se diferencia en un número múltiple de cuatro lo que de hecho no es así. O sea, las palabras *ac* y *accc* no son equivalentes. Junto con esto hemos acabado la fundamentación del algoritmo propuesto.

La resolución del problema de las palabras dada ampliamente en el ejemplo 4 para un cálculo asociativo concreto caracteriza en mucho los conceptos y métodos que también surgen al investigar el problema de las palabras para otros cálculos. Nos queda todavía aclarar el sentido del problema de las palabras y su importancia para el álgebra moderna. Lo más práctico es hacer esto en un ejemplo concreto.

Examinemos un cuadrado (fig. 3, I). Veamos en él tres *autocoincidencias*, o sea, transformaciones que hacen cambiar el cuadrado a una forma igual a sí mismo:

1) simetría (reflejo especular) respecto al eje vertical que pasa por el centro del cuadrado *O*;

2) simetría respecto al eje horizontal que pasa por el centro del cuadrado *O*;

3) giro de 90° en la dirección de las agujas del reloj alrededor del centro *O*.

A estas transformaciones las llamaremos elementales y las designaremos con las letras *a*, *b*, *c* correspondientemente. En la fig. 3 (III — IV — V) se muestra cómo cambia la posición de los vértices del cuadrado (II) al realizar cada una de estas transformaciones elementales.

Observaremos que al realizar dos o varias cualesquiera autocoincidencias del cuadrado, como resultado tiene lugar también la autocoincidencia del cuadrado. Aceptaremos la definición de uso general según la cual multiplicar dos trans-

formaciones dadas (en particular, dos autocoincidencias del cuadrado) quiere decir realizarlas sucesivamente una después de la otra. Acordaremos también conservar el sistema corriente de signos para la operación de multiplicación y el término de *producto* para la transformación que aparezca como resultado. Por ejemplo, el producto *cc* es el resultado

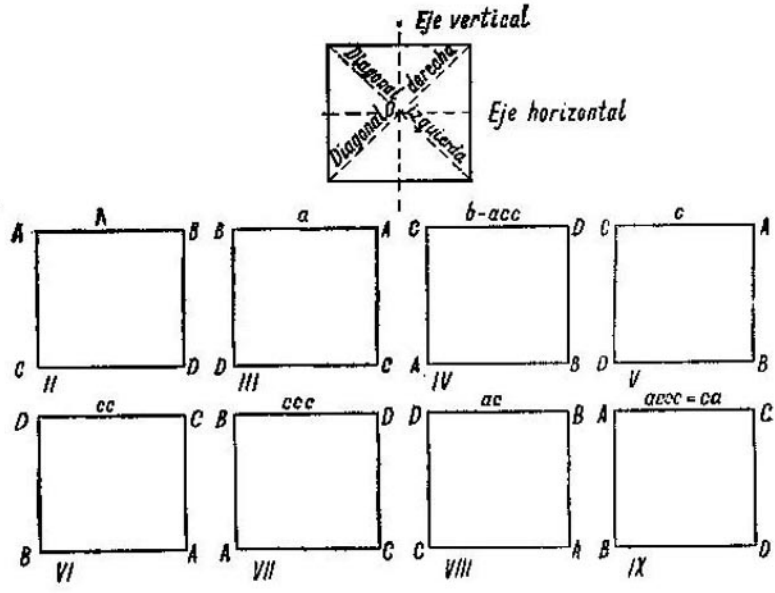


Fig. 3

de dos giros sucesivos de 90°, o sea, un giro de 180°; el producto *ac* es el resultado de la simetría respecto al eje vertical y del seguido giro de 90° lo que es equivalente a la simetría respecto a la diagonal izquierda (véase fig. 3, I). El producto (*ac*) (*cc*) de los dos productos anteriores pues es equivalente a la simetría respecto a la diagonal derecha (véase fig. 3, I).

Nuestra multiplicación no es conmutativa; en la fig. 3 (VIII — IX) están representadas dos posiciones diferentes de los vértices del cuadrado que resultan con las autocoinciden-

cias *ac*, *ca* del cuadrado inicial (II). Sin embargo, la familiariza con la multiplicación aritmética la propiedad asociativa (combinatoria): cualesquiera que sean las transformaciones de p, q, r, tiene lugar la identidad $(pq)\,r = p\,(qr)$. Debido a esto en los productos se puede desapreciar la colocación de los paréntesis; así, por ejemplo, $(ac)\,(cc)$ y $(((ac)\,c)\,c)$ dan la misma autocoincidencia del cuadrado que es la simetría respecto a la diagonal izquierda.

El objeto de nuestras investigaciones será el conjunto Ω compuesto de las transformaciones elementales de a, b, c y de todas las autocoincidencias del cuadrado que pueden ser presentadas como el producto de un número finito (pero arbitrario) de transformaciones elementales. Teniendo en cuenta la propiedad asociativa de la multiplicación al apuntar los símbolos, de los elementos en Ω se pueden omitir los paréntesis y limitarse a la anotación correspondiente al orden de las letras que representan las respectivas autocoincidencias elementales, por ejemplo, *abb*, *cabb*, *accc*, etc. Esto quiere decir que cualquier producto se anotará como palabra en el alfabeto $\{a, b, c\}$.

De la propiedad asociativa de la multiplicación se deduce también que si a la palabra P se le añade a la derecha la palabra Q de tal forma que resulte una sola palabra PQ, entonces ella representará el producto de las autocoincidencias expresadas con las palabras P y Q, respectivamente. Por ejemplo, la palabra *abccab* representa el producto de las autocoincidencias expresadas con las palabras *abc* y *cab*.

Es evidente que existe una cantidad innumerable de diferentes anotaciones gráficas de palabras en el alfabeto $\{a, b, c\}$; no obstante, las anotaciones gráficas de diferentes palabras pueden representar en Ω una misma autocoincidencia. En este caso es natural considerar las palabras iguales y designar esta igualdad en la forma corriente. El lector fácilmente apreciará que las siguientes igualdades son legítimas:

$$b = acc, \tag{1}$$

$$ca = accc. \tag{2}$$

Para eso es suficiente comparar la posición de los vértices del cuadrado que aparecen como resultado de las transformaciones de las partes izquierda y derecha de estas igualdades. Además, se ve fácilmente que cada una de las pala-

bras *aa*, *bb*, *cccc* da una misma autocoincidencia, precisamente la llamada transformación idéntica en la que todos los vértices se quedan en los lugares anteriores. Puesto que esta transformación no cambia nada, es conveniente representarla también como una palabra vacía Λ. Así, pues, también tienen lugar las igualdades

$$aa = \Lambda, \tag{3}$$

$$bb = \Lambda, \tag{4}$$

$$cccc = \Lambda. \tag{5}$$

La comparación de las igualdades (1) — (5) con las sustituciones admisibles del cálculo asociativo del ejemplo 4 sugiere la siguiente proposición que establece la relación entre este cálculo y el sistema examinado de transformaciones del cuadrado:

Dos productos de autocoincidencias elementales del cuadrado prefijan una misma transformación en el caso y sólo en el caso cuando las palabras que los representan son equivalentes en el cálculo del ejemplo 4.

En efecto, de las igualdades (1) — (5) se deduce que al aplicar cada vez cualquier sustitución admisible a cualquier palabra S, ésta se transforma en una palabra igual. Por ejemplo, aplicando la sustitución *ca* — *accc* a la palabra *bcac*, obtenemos la palabra *bacccc*; pero es que debido a la calidad asociativa de la multiplicación podemos anotar: *bcac* = *b* (*ca*) *c* y *bacccc* = *b* (*accc*) *c*; los segundos miembros son iguales como productos correspondientes de iguales factores lo que quiere decir que los primeros miembros también son iguales entre sí. Concluyendo, *cualesquiera dos palabras contiguas son iguales.*

Ahora ya es fácil comprender que la equivalencia de dos palabras en nuestro cálculo asociativo lleva tras sí su igualdad (o sea, la igualdad de autocoincidencias que ellas prefijan). Realmente, si $S \sim T$, entonces en la cadena correspondiente cualesquiera dos elementos contiguos son iguales, entonces vale decir que también $S = T$.

Tiene lugar también la afirmación inversa: *si las palabras son iguales, entonces son equivalentes.* Efectivamente, si dos palabras son iguales, entonces también son iguales sus

correspondientes palabras reducidas (esto se infiere de la afirmación directa). Al mismo tiempo se puede directamente comprobar que todas las ocho palabras reducidas dan por parejas diferentes autocoincidencias (véase la fig. 3, II — IX, en donde se representan las posiciones de los vértices del cuadrado (fig. 3, II) con las autocoincidencias que corresponden a las ocho palabras reducidas). Por eso, si dos palabras son iguales, les corresponde una misma palabra reducida, eso quiere decir, según lo demostrado anteriormente, que son equivalentes.

Así, pues, la equivalencia formal de dos palabras en nuestro cálculo recibe un sentido geométrico concreto y el discernimiento de la equivalencia de dos palabras toma el sentido de solución de un problema geométrico concreto. Al mismo tiempo el algoritmo descrito se presenta ante nosotros como un método general de resolución de cualquier problema geométrico del tipo dado.

Análogamente ocurre también con otros cálculos en los que la equivalencia formal también admite una interpretación geométrica, algebraica o de otra clase. Se puede decir sin exageración que en cualquier terreno de las matemáticas hay teoremas que pueden ser formulados después de cierta preparación en forma de afirmación sobre la equivalencia de dos palabras en cierto cálculo. En este pequeño libro no hay la posibilidad de examinar este círculo de cuestiones; ciertas aclaraciones más se darán al paso de las explicaciones siguientes (véase el § 6).

Observemos además, que partiendo de la interpretación geométrica que hemos dado al problema de las palabras en el cálculo estudiado, se puede ahora crear un algoritmo de manera inmediata e incluso algo más sencilla. Precisamente, es suficiente para cada uno de los dos productos propuestos realizar de hecho la sucesión de las autocoincidencias correspondientes (aunque sea en un dibujo) y comparar los resultados.

Ejercicio. Solucionar el problema de las palabras para el cálculo asociativo dado en el alfabeto $\{a, b\}$ con las sustituciones admisibles:

$$aaa = bb,$$

$$bbbb = \Lambda.$$

§ 4. MAQUINA DE CALCULAR CON MANDO AUTOMATICO

La creación de un algoritmo para los problemas de cierto tipo dado (y sobre todo un algoritmo «bueno», de un cómodo empleo), en los casos cuando eso se consigue, está relacionada, en general, con finos y complicados razonamientos que exigen una alta calificación y gran inventiva. No obstante, desde el momento cuando el algoritmo ya ha sido creado el proceso de resolución de los problemas correspondientes se hace tal que lo puede cumplir exactamente una persona que incluso no tenga ni el mínimo concepto de la esencia del propio problema. Se exige solamente que esta persona sea capaz de cumplir las sencillas y poco numerosas operaciones elementales de las cuales se compone el proceso y, además, que escrupulosamente y sin objeción se atenga a la prescripción (algoritmo) propuesta. Tal persona, actuando, como se dice, maquinalmente, podría solucionar con éxito cualquier problema del tipo examinado. La expresión «maquinalmente» se emplea aquí para subrayar la calidad de precisión del algoritmo; no obstante, el desarrollo actual de la ciencia y de la técnica le da a esta expresión un sentido directo. Precisamente, en el lugar de esta persona hipotética que resuelve un problema sin comprender su sentido (o sin querer saberlo), realmente se puede colocar una máquina que cumpla ese mismo proceso. Así es la máquina computadora de control automático de hoy en día.

Nuestra tarea inmediata consiste en aclarar los principios fundamentales de la construcción y del funcionamiento de semejantes máquinas. Para eso, preliminarmente, otra vez volveremos a examinar el proceso algorítmico que realiza el hombre al hacer cálculos.

Ateniéndose al algoritmo, el hombre que hace cálculos realiza un proceso donde tienen lugar la entrada, el almacenamiento, el tratamiento y la salida de ciertos datos (cierta *información*). Corrientemente el que hace los cálculos apunta (representa) estos datos en el papel por medio de cifras, letras u otros símbolos. El conjunto de todos estos símbolos se llama *alfabeto*. Por ejemplo, en el álgebra se emplea un alfa-

beto en el que además de las letras corrientes entran cifras, signos de las operaciones algebraicas, paréntesis, etc.

Es característico para un proceso de cálculo que se realiza con la participación del hombre que haya en él las tres siguientes etapas (véase el esquema de la fig. 4, *a*).

1. *El almacenamiento de la información* corrientemente se realiza anotando todos los datos en *hojas de papel*. En los datos entran también las instrucciones (el esquema del algoritmo) para la resolución del problema. Observemos que en realidad el calculador no apunta absolutamente todo en el

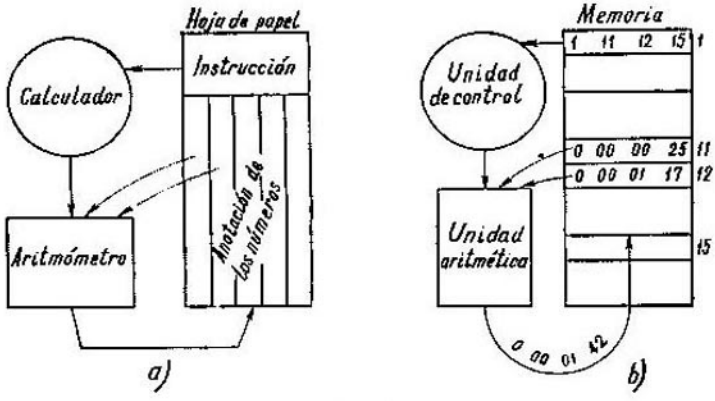


Fig. 4

papel; algunas cosas sencillamente las recuerda (las conserva no en la hoja de papel sino en su memoria) y ciertos datos los saca de diferentes manuales y tablas. Empero, esto no debe atenuar la idea básica que consiste en que se presupone que en el proceso de cálculo haya tales medios que aseguren la conservación de todos los datos necesarios. Así que en nuestro esquema al decir *una hoja de papel* debemos entender el conjunto de todos los medios que garantizan el almacenamiento de datos.

2. *El tratamiento de la información* presupone que el calculador es capaz de cumplir cada una de las operaciones elementales previstas en el algoritmo y que puede también realizarlas por medio de mecanismos especiales, por ejemplo, las operaciones aritméticas con números pueden hacerse en un aritmómetro. Cada operación de por sí consiste en lo

siguiente: el calculador en concordancia con la instrucción saca ciertos datos (por ejemplo, números) de unas determinadas columnas de la hoja de papel, introduce éstos en el mecanismo de tratamiento (en el aritmómetro) y el resultado de este tratamiento lo coloca en un lugar completamente determinado de la hoja.

3. *El control del proceso*, o sea, la toma de decisión para realizar en la etapa dada del proceso una u otra operación y para preparar su cumplimiento se ejecuta por el mismo calculador de acuerdo con la instrucción.

¿De qué dispositivos está compuesta una computadora y cómo ellos interaccionan entre sí? La contestación a esta pregunta sale de que en la computadora deben de ocurrir los mismos procesos que acaban de ser descritos pero sin la participación del calculador.

Ante todo, para representar la información, la máquina también tiene que tener un *alfabeto* determinado; empero, en lugar de la representación gráfica corriente de los símbolos que se distinguen uno de otro por su imagen, en la máquina los diversos símbolos del alfabeto se expresan con ciertos estados físicos diferentes uno del otro, por ejemplo, con diferentes tensiones eléctricas o con diferentes estados de imantación.

Una serie entera de razonamientos justifica la preferencia del empleo de un alfabeto de dos símbolos (el alfabeto *binario*) denominados condicionalmente 0 y 1. Así, por ejemplo, este alfabeto se realiza a modo físico muy fácilmente en los circuitos eléctricos en forma de dos estados: alta tensión (o corriente eléctrica) y baja tensión (o falta de corriente). Tendremos también que contar con lo que las operaciones lógicas elementales se efectúan con variables que pueden tomar dos valores: «verdadero» y «falso». No obstante, la elección de un alfabeto u otro y del procedimiento de representación en él de los datos necesarios está todavía muy lejos de ser decisivo para la comprensión de la construcción y del funcionamiento de la computadora. Por eso, limitándonos a la observación de que en las computadoras actuales se emplea con preferencia el alfabeto binario y el sistema de numeración binario (en lugar del corriente decimal), a continuación no vamos a meternos en estos detalles.

Así, pues, la información que llega a la computadora lo mismo que la que se elabora en ella durante el proceso de su

funcionamiento se representa en forma de ciertos parámetros físicos. En los casos que nos interesan todos los datos que componen la información estarán cifrados con números. En particular, el mismo algoritmo por el cual debe guiarse la computadora en su funcionamiento también estará cifrado en forma de un conjunto de números. Los algoritmos preparados especialmente para las computadoras corrientemente se llaman *programas*. Entre la información que trata la computadora, la parte más importante es el programa.

Siguiendo, en concordancia con el esquema de la fig. 4, *a*, en la computadora hay dispositivos (órganos) que cumplen las funciones de almacenamiento de la información, de su tratamiento, de control del proceso (véase el esquema de la fig. 4, *b*).

1. *El dispositivo de memoria* cumple la función de la hoja de papel. En él se graban, en el lenguaje condicional de la computadora, todos los datos necesarios incluyendo el programa. Es poco probable que a alguien le quepa duda de la posibilidad de realizar un órgano físico que cumpla tales funciones. Efectivamente, esta función la puede cumplir una cinta magnética en la que los datos cifrados se graban y de la cual ellos se extraen en forma semejante a cómo se hace en un magnetófono. El dispositivo de memoria (la *memoria de la computadora*) está compuesto de un conjunto be células numeradas con los números naturales: 1, 2, 3, . . . Estos números se llaman *direcciones* de las células de memoria. Cada célula conserva o puede recibir para guardarla una comunicación cifrada; en las computadoras cualquier comunicación tal está representada, como ya dijimos, en forma de cierto número.

En las computadoras electrónicas reales también se emplean, además de la cinta magnética, otros medios de «recordación» que son precisamente los tubos de haz electrónica, el principio del funcionamiento de las cuales en cierto modo recuerda el funcionamiento de los tubos catódicos de televisión, los tambores magnéticos y otros. Las funciones de almacenamiento de datos están divididas entre estos órganos de la forma más conveniente. Sin embargo, no vamos a hacer diferencia entre las variadas partes componentes de la memoria, y sin perjuicio para la comprensión del material partiremos de la suposición de que haya un órgano de memoria único que podría ser de cinta magnética.

2. *La unidad aritmética* juega el mismo papel que un aritmómetro corriente a pesar de que los principios físicos en los que está basada su construcción son muy diferentes de los que se emplean en un aritmómetro corriente. El tratamiento de los datos que se introducen en ella para obtener el resultado necesario (por ejemplo, la suma de números) se realiza por medio de la transformación en un dispositivo electrónico de las señales eléctricas de entrada que representan los datos iniciales en señales eléctricas que representan los datos de salida. Los datos de entrada llegan a la unidad aritmética de las células de memoria, en donde estaban almacenadas, y la señal de salida parte de ella a la célula donde se va a conservar. Esto en forma de esquema se representa en la fig. 4, *b* en que los números de las células 11 y 12 se suman y el resultado se manda a la célula 15. Para que esta operación se realice en la computadora en cierto tiempo es necesario que al comienzo de este tiempo se formen las conexiones de las células 11 y 12 con la unidad aritmética y de la unidad aritmética con la célula 15; también es preciso que el aritmómetro se prepare para la operación necesaria (en este caso para la adición). Todo esto entra ya en la competencia de la unidad de control.

3. *La unidad de control* está destinada a cumplir las funciones que en el esquema de la fig. 4, *a* hace el mismo calculador. Precisamente el *dispositivo de control* en cada etapa del funcionamiento de la máquina crea las condiciones para la realización de la operación siguiente del proceso. Al hacer esto actúa como una central telefónica automática y une a los «abonados» (dispositivos y células de la computadora) que participan en cada una de las operaciones. Diciéndolo metafóricamente, la unidad de control mira el programa y en concordancia con él manda las órdenes sobre el funcionamiento de las unidades de la computadora que tienen que garantizar la operación siguiente.

Para hacer una descripción más exacta del cuadro que surge aquí, indicaremos que cada modelo de computadora se caracteriza por un determinado sistema de instrucciones (órdenes) que puede captar para su cumplimiento. Todo programa que se introduce en la computadora representa una determinada combinación de instrucciones y ciertos números auxiliares (parámetros) que se colocan en las células de memoria. Por ejemplo, en la БЭСМ (BESM), Gran computadora

electrónica de la Academia de ciencias de la URSS, se ha aceptado el denominado sistema de instrucciones con tres direcciones, cada una de las cuales es una sucesión de cuatro números:

$$\alpha\beta\gamma\delta,$$

de los cuales el primero indica el *número* de orden de la operación prescrita; los dos siguientes, las *direcciones* de las dos células con el contenido de las cuales se hará la operación y el último, la dirección de la célula en que se debe introducir el resultado (en total tres direcciones).

Prácticamente cada instrucción se apunta en una célula en forma de un solo número cuyas cifras están divididas en cuatro grupos que tienen su correspondiente destino. Por ejemplo, en la fig. 4, *b* en la célula 1 del dispositivo de memoria está anotado el número 1 11 12 15 que representa la siguiente orden cifrada:

«*Sumar* (*operación* N° 1) *los números de las células* 11 *y* 12 *y mandar el resultado a la célula* 15».

(Aquí ha sido aceptada la división en grupos de dos dígitos de derecha a izquierda. Para precisión en adelante nos atendremos también a esta forma de anotación de las instrucciones.) Corrientemente un sistema de instrucciones cuenta con varias decenas de ellas. Aquí indicaremos sólo las más utilizables.

1. *Instrucciones aritméticas*:

a) 1 β γ δ — significa *sumar* el número que está en la célula β con el número de γ y mandar la suma a δ;

b) 2 β γ δ, *restar* del número de β el número de γ y mandar la diferencia a δ;

c) 3 β γ δ, *multiplicar* el número de β por el número de γ y mandar el producto a δ;

d) 4 β γ δ, *dividir* el número de β por el número de γ y mandar el cociente a δ.

2. *Instrucciones de salto*:

c) 5 00 00 δ — significa pasar a la instrucción que se encuentra en la célula δ (transmisión *incondicional* del control del funcionamiento del programa);

f) 5 01 γ δ, pasar a la instrucción que se encuentra en la célula δ si en la célula γ hay un número positivo;

g) 5 02 γ δ, pasar a la instrucción que se encuentra en δ si en la célula γ hay un número negativo.

3. *Instrucción de ruptura* (parada del proceso de computación):

0 00 00 00.

Además de las instrucciones indicadas hay también instrucciones llamadas *operaciones lógicas* y otras en las que aquí no nos vamos a parar. Las instrucciones indicadas son suficientes para componer con ellas los programas más variados. En el epígrafe siguiente serán examinados algunos ejemplos.

Las instrucciones *f* — *g* se llaman *condicionales*; ellas se aceptan para su ejecución solamente si se cumple la condición indicada, en el caso contrario la unidad de control las deja pasar sin cumplirlas.

Corrientemente la computadora cumple las instrucciones en el orden en que ellas se encuentran en las células de memoria. Las desviaciones de este orden pueden ocurrir sólo como resultado del cumplimiento de una instrucción de salto (incondicional o condicional si se cumple la condición correspondiente).

El funcionamiento de la computadora se realiza en *tiempos*; en el curso de cada uno de ellos se cumple una instrucción inmediata. Al principio de cada tiempo de una de las células de memoria llega a la unidad de control el número-instrucción que se contiene en esa célula. En cuanto la instrucción llega allí el control realiza automáticamente las conexiones correspondientes y así asegura el cumplimiento de la operación de turno del proceso. Después de eso a la unidad de control llega otra instrucción y la computadora cumple la operación siguiente, etc., hasta la parada de la máquina con la obtención de los resultados.

La posibilidad de la realización técnica de tal unidad de control no representa nada inesperado; en efecto, de esta unidad no se exige más de lo que sabe hacer una central telefónica automática en donde el número de llamada se conecta por medio del envío de una señal eléctrica correspondiente.

Tales son los tres dispositivos principales de la computadora. Es verdad que en ella hay también una serie más de órganos importantes, en particular, para la introducción de los datos, para la salida de los resultados que ella elabora.

No obstante, estos dispositivos no tienen importancia al estudiar los principios del funcionamiento de la computadora y al aclarar sus posibilidades lógicas y matemáticas. Por eso, en la continuación de nuestro estudio podemos no aprovecharlos suponiendo que la entrada de la información a la computadora y la salida de la información resultante se realiza directamente en el dispositivo de memoria.

§ 5. PROGRAMAS
(los algoritmos de máquina)

En este epígrafe veremos ejemplos de programas compuestos para una computadora electrónica con un sistema de instrucciones de tres direcciones. Estos programas resultan de los algoritmos antes vistos después de su tratamiento en el que se tiene en cuenta que ahora se van a ejecutar no por una persona, sino por una computadora automática que tiene el sistema de instrucciones en cuestión. El estudio de los ejemplos propuestos está llamado a aclarar el sentido real de la expresión que se emplea corrientemente: «*la computadora en su funcionamiento se guía por el programa que ha sido introducido en ella*». Precisamente será aclarado de qué manera se regula uno u otro orden de entrada de las instrucciones a la unidad de control y cómo se consigue por medio de un número relativamente pequeño de instrucciones asegurar el cumplimiento de cadenas de operaciones, con frecuencia muy largas, necesarias para la resolución de una u otra variante del problema tratado.

En lo sucesivo tenemos en cuenta que las operaciones aritméticas de adición, sustracción, multiplicación y división se indican en las instrucciones con los números 1, 2, 3, 4, correspondientemente (véase el § 4).

Ejemplo 1. Componemos el programa de la resolución del sistema de ecuaciones

$$ax + by = c,$$
$$dx + ey = f.$$

Para precisar aceptamos que los coeficientes a, b, c, d, e, f han sido introducidos sucesivamente en células de memoria comenzando desde el número 51:

Dirección	*Contenido*
51	*a*
52	*b*
53	*c*
54	*d*
55	*e*
56	*f*

Continuando, para los resultados intermedios y finales de los cálculos dedicamos las células 31—50.

Como se deduce de las fórmulas

$$x = \frac{ce - fb}{ae - bd}, \qquad y = \frac{af - cd}{ae - bd},$$

para obtener el resultado hay que hacer sucesivamente 6 multiplicaciones, tres restas y dos divisiones. En concordancia con esto el programa que proponemos está compuesto de 12 instrucciones anotadas en las células 1—12:

Dirección	*Contenido (instrucciones)*
1	3 53 55 31
2	3 56 52 32
3	3 51 56 33
4	3 53 54 34
5	3 51 55 35
6	3 52 54 36
7	2 31 32 37
8	2 33 34 38
9	2 35 36 39
10	4 37 39 40
11	4 38 39 41
12	0 00 00 00

Las instrucciones llegan a la unidad de control y se cumplen en el orden de acrecentamiento de sus direcciones. Después de cumplir la última instrucción, se han recibido en las células 31—41 y se encuentran en ellas los números siguientes:

Dirección	*Contenido*
31	*ce*
32	*fb*
33	*af*
34	*cd*
35	*ae*
36	*bd*
37	*ce—fb*
38	*af — cd*
39	*ae — bd*

40 $\frac{ce - fb}{ae - bd}$

41 $\frac{af - cd,}{ae - bd}$

que representan los resultados intermedios del cálculo y el resultado final (en las células 40, 41).

Ejemplo 2. Hay que encontrar las soluciónes de n sistemas de ecuaciones dados:

$$\begin{cases} a_i x + b_i y = c_i \\ d_i x + e_i y = f_i \end{cases} \quad i = 1,2, \ldots n.$$

El algoritmo resolutivo representa la repetición n veces del algoritmo de resolución de un sistema tal. Se podría componer el programa correspondiente para la computadora por el modelo anterior de tal manera que los $6n$ coeficientes estuviesen emplazados en $6n$ células de memoria y que el programa tuviese $11n + 1$ instrucciones. Después que el primer ciclo de 11 instrucciones elabore la solución del primer sistema, el segundo ciclo de 11 instrucciones elabora la solución del segundo sistema y esto se continúa... n veces. La instrucción $(11n + 1)$ será la instrucción de ruptura.

Sin embargo, tan considerable aumento del volumen del programa no es conveniente y se puede evitarlo. Para eso observaremos que cada ciclo posterior de 11 instrucciones puede ser obtenido del anterior cambiando las direcciones en cada instrucción. Eso es así: si los $6n$ coeficientes se emplazan de uno en uno en células que están una detrás de otra, comenzando, por ejemplo, por la N° 51, pues, en las primeras 6 instrucciones las direcciones de los multiplicadores deben ser aumentados cada una en 6 para obtener las instrucciones correspondientes del segundo ciclo.

Siguiendo, es necesario aumentar en dos cada una de las últimas direcciones de la décima y undécima instrucciones para que la solución del sistema siguiente quede anotada en las dos células que siguen a las células donde se guarda la solución del sistema anterior. Este cambio de las direcciones puede ser realizado empleando las *instrucciones* llamadas *de cambio de direcciones* de las que en nuestro caso habrá 8. Para eso emplazamos en las células 25 y 26 los parámetros:

N° de la célula	*Contenido*
25	0 06 06 00
26	0 00 00 02,

y en las células 12—18 ocho instrucciones de cambio de dirección:

Nº de la célula	*Contenido*
12	1 01 25 01
13	1 02 25 02
14	1 03 25 03
15	1 04 25 04
16	1 05 25 05
17	1 06 25 06
18	1 10 26 10
19	1 11 26 11

Después de que se cumplan las instrucciones de las células 1—19, como en el caso anterior, en las células 40—41 se recibirá la solución del primer sistema de ecuaciones y al mismo tiempo en las células 1—6 y 10—11 ya se encontrarán las siguientes instrucciones con las direcciones cambiadas:

Nº de la célula	*Contenido*
1	3 59 61 31
2	3 62 58 32
3	3 57 62 33
4	3 59 60 34
5	3 57 61 35
6	3 58 60 36
10	4 37 39 42
11	4 38 39 43

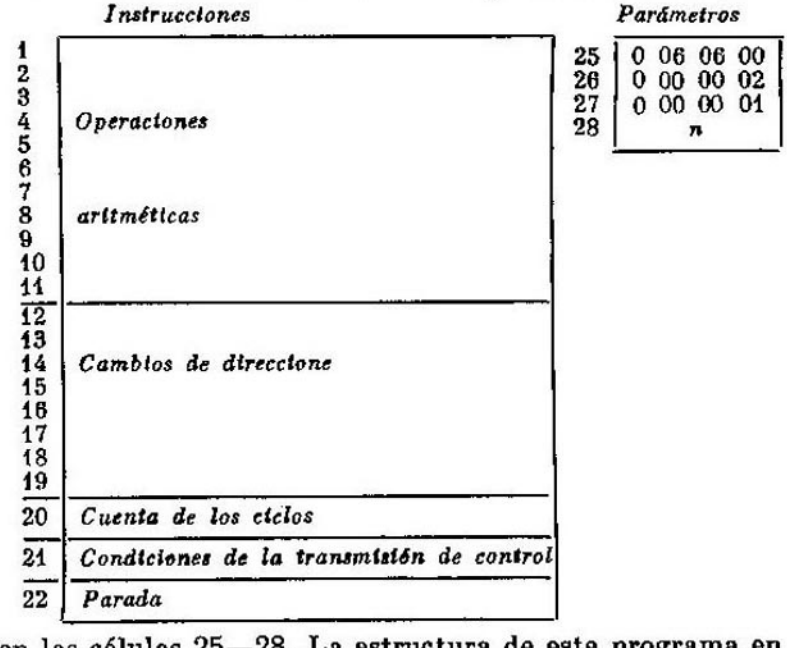
Por eso, si ahora por segunda vez se efectúan las instrucciones de las células 1—19, pues, como resultado de este segundo ciclo de funcionamiento de la computadora se encontrará la solución del segundo sistema de ecuaciones que se mandará a las células 42—43; además, se realiza el cambio de direcciones consecutivo en las instrucciones 1—6 y 10—11 lo que crea las condiciones para el tercer ciclo análogo de funcionamiento, etc.

¿Cómo se podrá asegurar semejante cumplimiento en ciclo de las 19 instrucciones un número de veces igual al número de sistemas de ecuaciones dados, pero de tal manera que después de que se encuentren las soluciones de todos los sistemas, el funcionamiento de la computadora se interrumpa? Para eso en las células 27 y 28 emplazaremos además los parámetros 0 00 00 01 y n (n es el número de todos los sistemas dados) y a las 19 instrucciones anteriores les añadiremos las tres siguientes:

Nº de la célula	*Contenido*
20	2 28 27 28
21	5 01 28 01
22	0 00 00 00

La instrucción 20 disminuye en una unidad el contenido de la célula 28 después de cada cumplimiento del ciclo de instrucciones 1—19. La instrucción 21 es la transmisión condicional del control del programa a la instrucción 1; ésta se cumple mientras que en la célula 28 se conserva un número positivo y así se asegura el paso al ciclo siguiente de instrucciones 1—19 para la resolución del sistema de ecuaciones siguiente. Si en la célula 28 hubiese un cero, lo que ocurre exactamente después de los *n* ciclos de las 1—19 instrucciones, entonces la transmisión del control del programa no se realizará y se cumplirá la instrucción siguiente, la 22, que interrumpirá el funcionamiento de la computadora.

De todo lo dicho se aclara que para el problema planteado de la resolución de *n* sistemas de ecuaciones es válido el programa compuesto de las instrucciones 1—22 indicadas anteriormente teniendo en cuenta los parámetros introducidos



en las células 25—28. La estructura de este programa en su conjunto se puede representar en forma del esquema siguiente:

Ejemplo 3. Veamos ahora un programa para encontrar el máximo común divisor de dos números a y b. Acordaremos dedicar las células 12 y 13 para los datos iniciales a y b, correspondientemente, las células 14 y 15 para los cálculos intermedios y que el resultado final después de la interrupción del funcionamiento de la computadora se contenga en la célula 15. El programa que se propone a continuación está compuesto conforme al procedimiento del algoritmo de Euclides que se trató en el § 1.

Nº de la célula	*Contenido* (instrucción)	*Aclaración*
01	1 12 05 15	Envío del número de la célula 12 a la 15
02	2 12 13 14	La diferencia entre los números de la 12 y 13 se manda a la 14
03	5 02 14 06	Paso a la instrucción 06 a condición de que en la 14 haya un número negativo
04	5 01 14 09	Paso a la instrucción 09 a condición de que en la 14 haya un número positivo
05	0 00 00 00	Parada
06	1 13 05 12	Envío del número de la 13 a la 12
07	1 15 05 13	Envío del número de la 15 a la 13
08	5 00 00 01	Salto incondicional a la instrucción 01
09	1 13 05 12	Envío del número de la 13 a la 12
10	1 14 05 13	Envío del número de la 14 a la 13
11	5 00 00 01	Salto incondicional a la instrucción 01

Después de los dos primeros tiempos en las células 12—15 se emplazarán los números siguientes:

N° de la célula	*Contenido*
12	A
13	b
14	$a - b$
15	a

Si $a - b = 0$ (o sea, $a = b$), entonces las instrucciones 03 y 04 de transmisión condicional del mando del programa se

dejan pasar y se cumple la instrucción 05 que interrumpe el funcionamiento de la computadora. En este momento en la célula 15 ya se contiene el resultado necesario (compare esto con la indicación 3 del § 1).

Si $a - b < 0$ (o sea, $a < b$), entonces la instrucción 03 traspasa el mando del programa a la instrucción 06 que junto con la instrucción 07 que le sigue realiza el cambio de lugares de los números a y b en las células 12 y 13 (compárelo con la indicación 4). Después la instrucción 08 asegura el salto incondicional a la 01 y así comienza el segundo ciclo de funcionamiento de la computadora.

Y si $a - b > 0$ (es decir, $a > b$), entonces se deja pasar la instrucción 03, la instrucción 04 traspasa el mando del programa a la instrucción 09 la que junto con la siguiente instrucción 10 traslada a las células 12 y 13 el sustraendo y el resto anterior, o sea, b y $a - b$, correspondientemente, (compárelo con la instrucción 4). Después la instrucción 11 asegura el salto incondicional a la instrucción 01 y de esta manera comienza el segundo ciclo de funcionamiento de la computadora.

La sucesión de ciclos de funcionamiento de la computadora generará en las células 12 y 13 la sucesión de pares de números:

$$(a_1, b_1), (a_2, b_2), \ldots, (a_i, b_i), (a_{i+1}, b_{i+1}), \ldots$$

y en la célula 15 la sucesión de números

$$a_1, a_2, a_3, \ldots, a_i, a_{i+1}, \ldots$$

hasta que no aparezca el primer par de números iguales a_k, b_k. Entonces la instrucción 05 interrumpirá el funcionamiento de la computadora y en la célula 15 quedará emplazado el resultado que se busca a_k.

En los ejemplos examinados están ya reflejados con suficiente claridad dos principios fundamentales del funcionamiento de las computadoras automáticas:

1. Corrientemente las instrucciones del programa se cumplen en la computadora según el orden en que se han anotado en las células de memoria.

No obstante, la computadora es apta para cambiar automáticamente el curso del proceso de cálculo en dependencia de los resultados intermedios que se obtienen durante éste.

Eso se consigue por medio de la introducción de las instrucciones condicionales.

2. Teniendo un programa de un volumen relativamente pequeño la computadora es apta para cumplir sucesiones de cálculos bastante largas debido a que ella misma puede transformar y repetir muchas veces todo el programa o algunas partes suyas. Tal posibilidad se garantiza porque el programa, que está codificado con números, se emplaza en el mismo dispositivo de memoria donde también se conservan números corrientes y, en consecuencia, la computadora puede también realizar operaciones con números condicionales que son los códigos de las instrucciones. Por ejemplo, la computadora puede cambiar las direcciones que hay en una serie de instrucciones.

Estos mismos principios caracterizan el funcionamiento de la computadora al resolver problemas que no tienen un carácter evidente de cálculo aritmético. Por ejemplo, se puede programar el algoritmo de Teseo (la búsqueda en un laberinto) o los conocidos algoritmos para el problema de las palabras en ciertos cálculos asociativos y realizar los procesos correspondientes en la máquina. Es verdad que para eso hace falta que la máquina pueda realizar ciertas operaciones elementales complementarias, además de las aritméticas, y las transmisiones del mando que participaban en los problemas aritméticos examinados anteriormente. En las computadoras electrónicas reales estos pocos tipos de operaciones simples se pueden cumplir (están previstas las instrucciones correspondientes). Por eso, para que una misma computadora pase a hacer otra clase necesaria de tareas es bastante cambiar el programa.

No sólo en las matemáticas sino también en otros variadísimas esferas de la actividad humana se encuentran procesos que se ejecutan según una prescripción formal rigurosamente determinada (es decir, conforme a un algoritmo) y que también se les puede programar. Por ejemplo, en la contabilidad y en la planificación el análisis de los datos recibidos, su tratamiento, la composición de los balances económicos para obtener resultados óptimos se forman de una larga sucesión de operaciones elementales de varios tipos que se realizan en rígida concordancia con instrucciones y esquemas especiales. En otros casos a pesar de que por ahora todavía no existen algoritmos exactos y acabados

pero se los podría crear y llevar su descripción formal hasta la perfección. Eso en particular se refiere al problema de traducción de una lengua a otra. Con un tratamiento formal suficiente y una clasificación de las reglas principales de la gramática y del estilo, así como de los procedimientos del empleo del diccionario, se puede crear un algoritmo completamente satisfactorio para la traducción, digamos, de un texto científico u oficial (para algunas lenguas, tales algoritmos ya han sido creados).

En relación con esto es también interesante recordar que en muchos juegos a sus reglas les son propios ciertos rasgos de la prescripción formal de las acciones. Esta circunstancia permite plantear la creación de algoritmos para llevar el juego con éxito. Un tal algoritmo basado en cierta táctica del juego le debe indicar al jugador la única (la mejor desde el punto de vista de esta táctica) jugada que él debe hacer en cada posición. Por ejemplo, en el ajedrez se puede introducir un sistema de valores para las figuras en el que el rey se evalúe en una cantidad muy grande de puntos, la reina, en menos, la torre, menos aún, etc., y el peón tendrá la evaluación mínima. Además de eso, de una manera determinada se apreciarán las ventajas de posición (la disposición de las figuras más cerca del centro del tablero, su movilidad, etc.). La diferencia entre la suma de puntos para las figuras blancas y la suma para las negras caracteriza en el sentido de la táctica dada la superioridad material y de posición de las blancas sobre las negras en cada momento del juego. El algoritmo más sencillo del juego consiste en el análisis de todas las jugadas que son posibles en la situación creada y en la elección entre ellas de la que lleva a la superioridad mayor desde el punto de vista del sistema de valores establecido. Es mejor pero más complicado el algoritmo que revisa todas las combinaciones posibles de tres o, digamos, cinco jugadas inmediatas y a base de esto hace la elección de la jugada óptima*).

*) En teoría existe un algoritmo que es el mejor puesto que con él siempre cuando sea posible se gana. Para los problemas de ajedrez del tipo: *se da mate a las negras en N jugadas* este algoritmo prescribe a las blancas lo siguiente: revisar todas las series posibles de N jugadas $A_1, B_2, A_3, B_4, \ldots, B_{n-1}, A_n$ (aquí $A_1, A_3, \ldots$ son las jugadas de las blancas y $B_2, B_4, \ldots$ son las de las negras) y elegir para sí una sucesión de jugadas $A_1, A_3, \ldots$ tal con las que

De todo lo dicho se aclara la gran variedad de clases de trabajo intelectual que se hacen o pueden hacerse a base de determinados algoritmos. En todos estos casos es posible en principio programar los algoritmos y transmitir el cumplimiento del trabajo correspondiente a una computadora de mando automático. En particular, ahora ya hay programas compuestos para la traducción de una lengua a otra y para el juego de ajedrez que se emplean con éxito en las computadoras, por ejemplo, en la БЭСМ (BESM) de la Academia de Ciencias de la URSS.

§ 6. LA NECESIDAD DE PRECISAR EL CONCEPTO DE ALGORITMO

Lo explicado anteriormente indica la profunda relación que existe entre los algoritmos y las computadoras automáticas.

Es evidente que cualquier proceso cuyas partes por separado se realizan consecutivamente, en una computadora automática puede ser descrito con un algoritmo. Por otro lado, todos los algoritmos conocidos hasta ahora y también los que se pueden prever teniendo en cuenta el estado actual de la ciencia *en principio* son realizables en computadoras automáticas.

La última tesis exige cierta aclaración. Como ya se señaló, el proceso del empleo de un algoritmo al resolver unos u otros problemas del tipo dado puede durar mucho tiempo, y la anotación de los datos con que el algoritmo opera puede ser extremadamente voluminosa. Por otra parte, la unidad de memoria (la memoria) de las computadoras actuales tiene una capacidad limitada (ya que el número de células es finito y el lugar para cada una de ellas, limitado). Por eso habrán algoritmos que en ciertas condiciones pueden resultar prácticamente irrealizables.

Eso se puede ilustrar con el ejemplo del algoritmo de Euclides. El simple problema del cálculo del máximo común

siempre se salga ganando independientemente de cuáles sean B_2, B_4, . . . Empero, este algoritmo es tan grande que su empleo, incluso en las veloces computadoras electrónicas, es prácticamente irrealizable.

divisor de dos números puede resultar irrealizable para un calculador que lo resuelve «a mano» si para este problema se necesita más papel y tinta que él tiene a su disposición. Por analogía, el algoritmo de Euclides en algún problema concreto dado puede resultar irrealizable para una computadora si él exige un volumen de memoria mayor que el de que esa máquina dispone.

Como ya se dijo, en los casos semejantes el proceso de empleo del algoritmo se considera como un *proceso potencialmente realizable* que lleva después de un número finito (aunque sea muy grande) de pasos al resultado buscado.

Al hablar sobre las posibilidades de realizar los algoritmos en las computadoras, se tiene en cuenta la posibilidad potencial de aumento ilimitado del volumen de memoria en la máquina.

La relación observada entre el concepto de algoritmo y el concepto de computadora automática con una potencial memoria ilimitada permite aclarar mejor el sentido de cada uno de ellos. Cualquier precisión de uno de estos conceptos representa simultáneamente la precisión del otro.

Empero, a pesar de que se ha subrayado la generalidad de estos conceptos, a ninguno de ellos todavía no lo hemos definido exactamente. La definición matemática exacta del concepto de algoritmo (y junto con ella la definición exacta del concepto de computadora automática) fue elaborada en la ciencia en los años treinta de nuestro siglo. ¿Por qué en el curso de siglos los matemáticos se contentaron con un concepto vago de algoritmo sin mostrar ninguna inquietud? ¿Por qué hace sólo relativamente poco tiempo surgió una aguda necesidad de elaborar una estricta definición matemática de este concepto tal que pudiese servir de objeto de investigación matemática?

Hasta hace poco el término de algoritmo se encontraba en las matemáticas sólo en relación con la composición (creación) de algoritmos concretos, cuando *la afirmación de la existencia del algoritmo para los problemas del tipo dado iba acompañada de su descripción real.* En estas circunstancias a la persona a quien se le comunicaba el sistema de reglas formales le quedaba sólo tomarlas en consideración y en el proceso de su aplicación convencerse de que ellas automáticamente llevan al resultado necesario. Por eso, no surgía la cuestión de la estricta definición del concepto «algoritmo»

y se podía contentarse con una vaga representación de algoritmo pero al mismo tiempo suficientemente afín y comprensible para cada matemático. No obstante, en el curso del desarrollo de las matemáticas comenzaron a acumularse tales hechos que cambiaron por completo esta situación. El motivo de esto fue el deseo natural de los matemáticos de crear algoritmos cada vez más y más poderosos que pudiesen resolver problemas de clases más amplias (problemas de un tipo muy general). Ahora pasaremos al examen de estos hechos.

Recordemos el algoritmo para la extracción de la raíz cuadrada que se describe en todos los manuales escolares. Se puede plantear un fin más general: crear un algoritmo general para la extracción de la raíz de cualquier índice de cualquier número dado. Es natural esperar que sea más difícil crear tal algoritmo, no obstante, es muy atractiva la perspectiva de poder utilizarlo. Se puede ir aún más adelante. Extraer la n-ésima raíz del número a quiere decir resolver la ecuación $x^n - a = 0$ (calcular la raíz de la ecuación). Por eso, se puede formular un problema aún más general.

Crear un algoritmo que permita para cualquier ecuación del tipo

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0 \quad (*)$$

(n es un número natural arbitrario) *encontrar todas sus raíces**).

Está claro que es todavía más difícil crear este algoritmo. En su esencia el contenido fundamental de la sección del álgebra superior llamada álgebra de los polinomios es la creación y la argumentación de este algoritmo. Pero también su importancia es grandísima.

Los ejemplos citados en cierta medida caracterizan la aspiración de los matemáticos de crear algoritmos todavía más poderosos que resuelvan problemas de clases aún más amplias (problemas de tipo muy general.) Se sobreentiende que en este aspecto el problema de la resolución de cualquier ecuación del tipo (*) no representa todavía el límite más allá del cual no se puede pasar. Es más, si queremos ser consecuentes en nuestra impetuosa aspiración de ampliar

*) Diciéndolo más exactamente, para cualquier número natural k encontrar la aproximación decimal de las raíces por exceso y por defecto con una exactitud de $\frac{1}{10^k}$.

la clase de problemas para los que es deseable tener un algoritmo resolutivo único, inevitablemente tendremos que llegar al planteamiento del siguiente problema:

Crear un algoritmo que permita resolver cualquier problema matemático.

Este planteamiento es ya tan general que realmente puede ser apreciado como un insolente desafío a todas las matemáticas. Además, uno se puede meter con este enunciado aunque sea porque no está completamente claro lo que se supone bajo el término de «cualquier problema matemático». Al mismo tiempo, la gran fuerza atrayente de semejante problema, su seducción, no necesitan una propaganda especial.

Este problema tiene su historia. Ya el gran matemático y filósofo alemán Leibniz (1646—1716) soñaba con la creación de un método general que permitiese resolver eficazmente cualquier problema. A pesar de que no consiguió encontrar tal algoritmo general, consideraba que llegaría el tiempo cuando éste sería creado y entonces cualquier discusión entre matemáticos se decidiría automáticamente, con papel y lápiz, de acuerdo con este algoritmo general.

Más tarde el mismo problema obtuvo una determinada precisión en forma de uno de los más importantes problemas de la lógica matemática, a saber, el *problema de la distinción de la posibilidad de deducción*. Al no tener posibilidad de hacer en este pequeño libro una extensión completa y exacta de la cuestión nos limitaremos a su descripción en términos generales.

Es conocido que el método axiomático en las matemáticas consiste en que todas las proposiciones de la teoría dada se obtienen de varias sentencias (axiomas) que se aceptan en esa teoría sin demostración. Antes de todas las otras teorías matemáticas se creó la geometría axiomática, empero, en las matemáticas actuales casi todas sus teorías están construidas sobre una base axiomática. En la lógica matemática se describe un lenguaje especial de fórmulas que permite expresar cualquier sentencia de la teoría matemática con fórmulas completamente definidas.

Empleando la terminología que se ha introducido anteriormente al examinar los cálculos asociativos, se puede decir que cualquier fórmula tal representa una palabra en cierto alfabeto especial que contiene, junto con los signos

empleados en las matemáticas, como son las letras, paréntesis, etc., otros signos especiales que representan operaciones lógicas, por ejemplo, la negación, la reunión (suma lógica), etc. Sin embargo, la principal analogía con el cálculo asociativo consiste en que el mismo proceso de deducción lógica de la conclusión S partiendo de la premisa R puede ser descrito en forma de un proceso de transformaciones formales de palabras bastante similares a las sustituciones admitidas en el cálculo asociativo. Esto permite hablar de un cálculo lógico en el que se da un sistema de transformaciones admitidas que representan los actos elementales de razonamiento lógico de los que se compone cualquier conclusión lógica y formal de cualquier grado de complejidad. Ejemplo de una tal transformación admisible es la exclusión en una fórmula de dos signos de negación que están juntos; digamos, «no es no bonito» puede ser transformado en «bonito» (es interesante comparar esta transformación admisible con la sustitución $aa \rightarrow \Lambda$ en el cálculo asociativo del ejemplo 4).

La cuestión sobre la posibilidad de deducción lógica de la proposición S de la premisa R en el cálculo lógico indicado se convierte en la indagación de la existencia de una cadena deductiva que lleva desde la palabra que representa la premisa R hasta la palabra que representa la proposición S. Ahora podemos formular el problema de la distinción de la posibilidad de deducción así:

para cualesquiera dos palabras (fórmulas) R y S en un cálculo lógico determinar si existe o no una cadena deductiva que lleva de R a S.

La solución se entiende en el sentido de un algoritmo que dé contestación a cualquier pregunta de este tipo (con cualesquiera que sean R y S).

Ahora ya no es difícil apreciar que la creación de un algoritmo tal permitiría obtener un método resolutivo general para el tratamiento automático de los más variados problemas de todas las teorías matemáticas construidas a base de axiomas. En efecto, la justedad de tal o cual proposición S (por ejemplo, el enunciado de algún teorema) se entiende en esta teoría como la posibilidad de deducción lógica de un sistema de axiomas tomado en calidad de premisa R. Entonces aplicando el algoritmo de distinción de la posibilidad de deducción se podría establecer si es

justa o no la afirmación S en la teoría examinada. Además, en el caso de una contestación afirmativa eficazmente se podría encontrar en el cálculo lógico la cadena deductiva correspondiente y por ella restablecer la sucesión de razonamientos que componen la demostración de la afirmación estudiada. Este presunto algoritmo resolvería con un único método efectivo casi todos los problemas matemáticos formulados pero no solucionados hasta el día de hoy. Esta circunstancia hace comprensible no sólo lo seductivo que sería la creación de un tal «algoritmo general», lo que también quiere decir una correspondiente «computadora omnipotente», sino al mismo tiempo la dificultad de su creación.

Efectivamente, a pesar de los largos y obstinados esfuerzos de muchos eminentes especialistas, se han quedado insuperadas las dificultades ligadas a esta creación. Más aún, pronto se descubrieron dificultades semejantes al intentar crear algoritmos de un tipo bastante más particular. Por ejemplo, entre ellos también se encuentra el problema de Hilbert acerca de las ecuaciones de Diofante (véase el § 1) y otra serie de problemas de las que se hablará posteriormente.

Como resultado de numerosos y vanos intentos de crear tales algoritmos se hizo evidente que se choca con unas dificultades de un carácter trascendental. Entonces surgió la sospecha de que *no para toda clase de problemas es posible crear un algoritmo resolutivo.*

Es evidente que la afirmación sobre la insolubilidad algorítmica de cierta clase de problemas, o sea, sobre la imposibilidad de encontrar el algoritmo resolutivo correspondiente, no es tan sólo el reconocimiento de que no conocemos este algoritmo y de que todavía nadie lo ha encontrado. Tal afirmación representa simultáneamente el pronóstico de que en todo el futuro el algoritmo nunca y por nadie será encontrado (con otras palabras, no existe) y necesita su argumentación por medio de alguna demostración matemática. Sin embargo no tiene sentido pensar en una demostración semejante mientras que falte una definición exacta del concepto de algoritmo, pues en el caso contrario no está claro la no existencia de qué se va a demostrar. Es útil recordar aquí que en la historia de las matemáticas antes también se conocían tales problemas que durante mucho tiempo no se sometían a resolución y, como después fue

establecido, son insolubles con los medios con que antes intentaban solucionarlos. Como ejemplo indicaremos el problema de la trisección de un ángulo y la resolución de ecuaciones en radicales.

Del curso escolar se sabe cómo se traza la bisectriz de un ángulo con ayuda de un compás y una regla. Los antiguos griegos ya se plantearon un problema semejante sobre la trisección de un ángulo con ayuda de un compás y una regla. No obstante, fue demostrado que es imposible realizar la trisección de un ángulo cualquiera con estos medios.

También se sabe del curso escolar que las raíces de una ecuación de segundo grado se expresan con sus coeficientes por medio de fórmulas en las que figuran signos de operaciones aritméticas y el signo radical (de segundo grado). Para las ecuaciones de tercero y cuarto grados también se han elaborado fórmulas con radicales que, es verdad, son considerablemente más complicadas y, además, las radicales en ellas son «de muchos pisos». No obstante, las búsquedas de semejantes fórmulas con radicales para las ecuaciones de grados mayores que el cuarto, que continuaron hasta el comienzo del siglo XIX, resultaron vanos, hasta que por fin fue establecido el siguiente notable resultado:

Para ninguna n mayor o igual a cinco es posible indicar la fórmula que exprese las raíces de cualquier ecuación de n-ésimo grado con sus coeficientes por medio de radicales.

En ambos casos la propia demostración de la imposibilidad resultó posible porque habían dos definiciones exactas que dan contestación a las preguntas: «¿qué quiere decir construcción con ayuda de un compás y una regla?» y «¿qué quiere decir resolución de una ecuación en radicales?». A propósito, observaremos que en estas dos definiciones se precisa el sentido de ciertos algoritmos *especiales* que son precisamente el algoritmo de resolución de ecuaciones *en radicales* (y no en general un algoritmo de resolución de ecuaciones) y el algoritmo de trisección de cualquier ángulo con ayuda de un compás y una regla (y no un algoritmo de trisección en general). Hasta hace poco tiempo nosotros no teníamos una definición exacta para el concepto *general* de *algoritmo* y por eso la elaboración de una tal definición se ha hecho una de las tareas más importantes de la matemática moderna. Es muy importante subrayar que la elaboración de la definición del concepto de algoritmo (como tam-

bién cualquiera otra definición matemática) no se puede examinar como un acuerdo arbitrario de los matemáticos para comprender igualmente el término de *algoritmo*. Al formular esta definición hubo que superar dificultades que consistían en que la definición propuesta debía reflejar correctamente la esencia del concepto que, aunque en forma vaga, de hecho ya se tenía y que hemos ilustrado con muchos ejemplos. Con este fin, a partir de los años treinta del siglo XX se emprendió una serie de investigaciones para revelar todos aquellos medios que realmente se utilizan para crear algoritmos. La tarea consistía en dar, sobre esta base, una definición del concepto de *algoritmo* que fuese perfecta no sólo desde el punto de vista de exactitud formal sino también, lo más importante, desde el punto de vista de su correspondencia real al sentido del concepto definido. Aquí cada investigador partía de diferentes consideraciones técnicas y lógicas y como consecuencia fueron elaboradas varias definiciones del concepto de *algoritmo*. No obstante, después se aclaró que todas estas definiciones son equivalentes entre sí y, por consiguiente, determinan el mismo concepto, éste es el concepto exacto actual de *algoritmo*.

Todos los procedimientos de precisión del concepto de *algoritmo*, a pesar de toda su diferencia y variedad, en su esencia siempre llevaban y llevan a un mismo resultado. Esta circunstancia tiene una gran importancia cognoscitiva y es precisamente un testimonio del acierto de la definición elaborada.

Tiene un interés particular desde el punto de vista de la matemática de máquina, la definición del algoritmo en la que la esencia de este concepto se descubre a base del examen de los procesos que se realizan en la computadora.

Para tal definición matemática rigurosa hay que representar el mecanismo del funcionamiento de la computadora en forma de cierto esquema estándar, simple al máximo en cuanto a su estructura lógica y a la vez tan exacto que pueda servir de objeto para la investigación matemática. Esto fue hecho por primera vez por el matemático inglés Turing, guien propuso la máquina computadora más general y al mismo tiempo de la más simple concepción. Se debe observar que la máquina de Turing fue descrita en el año 1937, o sea, antes de la creación de las computadoras de hoy. En esto Turing partía de la idea general de asemejar el

funcionamiento de la máquina al trabajo de un calculador que opera en concordancia con cierta rigurosa prescripción. En la explicación que aquí damos sobre esta cuestión ya se emplean los principios generales del funcionamiento de las computadores electrónicas existentes.

§ 7. LA MAQUINA DE TURING

Las particularidades distintivas de la máquina de Turing en comparación con las computadoras electrónicas descritas en los §§ 4—5 consisten en lo siguiente:

1. En la máquina de Turing la descomposición del proceso en operaciones elementales simples se ha llevado, en cierto sentido, hasta el límite de las posibilidades. Así, por ejemplo, la operación de suma que figura en la computadora electrónica como una sola operación elemental, aquí se descompone en una cadena de operaciones todavía más simples.

Ni que decir tiene que esto alarga considerablemente el proceso que se realiza en la máquina de Turing, pero al mismo tiempo su estructura lógica se simplifica mucho y toma una forma estándar muy conveniente para las investigaciones teóricas.

2. En la máquina de Turing parte de la memoria *) se representa en forma de una cinta dividida en células e ilimitada por sus dos lados.

Es evidente que ninguna máquina creada o por crear puede tener una memoria infinita (una cinta ilimitada) y en ese sentido la máquina de Turing sólo representa un esquema idealizado que refleja la posibilidad potencial del aumento del volumen de memoria.

Esta idealización se justifica con la relación, citada anteriormente, entre el concepto de algoritmo y el concepto de computadora con una memoria potencial ilimitada.

Pasemos ahora a una descripción detallada de la máquina de Turing.

(*) Que es precisamente la llamada memoria exterior.

1. La máquina tiene un número finito de signos (símbolos)

$$s_1, s_2, \ldots, s_k,$$

que forman el llamado *alfabeto exterior* en el que se cifran los datos introducidos en la máquina y también los que se elaboran en ella. Para la generalidad del estudio posterior será conveniente aceptar que entre los signos del alfabeto exterior se encuentra el *signo vacío* (para precisión, que sea el s_1), el envío del cual (la anotación del cual) a cualquier célula de la cinta (de la memoria) extingue (borra) el signo que había antes en ella y deja la célula vacía. Diremos que en una célula vacía está depositado un signo vacío.

En cualquier etapa del funcionamiento de la máquina en cada célula puede haber sólo un signo. Cada comunicación que se guarda en la cinta está representada con un conjunto finito de signos del alfabeto exterior diferentes del signo vacío y colocados de uno en uno en ciertas células de la cinta. En el comienzo del funcionamiento de la máquina se introducen en la cinta los datos iniciales (la información inicial); el funcionamiento de la máquina se realiza en tiempos consecutivos, en el curso de los cuales se efectúa la transformación de la información inicial en información intermedia (al final de cada tiempo todo el conjunto de signos almacenados en la cinta forma la información intermedia correspondiente). Se puede introducir en la cinta, en calidad de información inicial, cualquier sistema finito de signos del alfabeto exterior (cualquier palabra en este alfabeto) que se coloque por las células de manera arbitraria. Ahora bien, en dependencia de la información inicial $\mathfrak{A}$ introducida son posibles dos casos:

a) después de un número finito de tiempos el funcionamiento de la máquina se interrumpe y ella da una señal de parada; en la cinta queda representada cierta información $\mathfrak{B}$. En este caso se dice que la máquina es utilizable para la información inicial $\mathfrak{A}$ y la transforma en la información resultante $\mathfrak{B}$;

b) la interrupción y la señal de parada nunca aparecen. En este caso se dice que la máquina no es utilizable para la información inicial $\mathfrak{A}$.

Se dice que la *máquina resuelve cierta clase de problemas* si ella siempre es utilizable para la información que en

un código determinado representa las condiciones de cada cual problema de este tipo y la transforma en la información que representa en el mismo código la solución de este problema.

2. El sistema de instrucciones con tres direcciones que se emplea en muchas computadoras electrónicas reales está acondicionado a la existencia de operaciones elementales en las que simultáneamente participa el contenido de tres células de memoria.

No obstante, en algunas computadoras electrónicas se emplea un sistema de instrucciones que tienen una sola dirección lo que está relacionado con el hecho de que en cada tiempo participa sólo una célula de memoria. (Llamaremos a ésta la *célula observada* en la etapa dada.) Así, por ejemplo, una instrucción con tres direcciones de suma de los números que están en las células β, γ y envío del resultado a la célula δ en condiciones adecuadas puede ser sustituido por tres instrucciones consecutivas: *a*) llamada (hacia el sumador) al número de la célula β, *b*) llamada al número de la célula γ, *c*) envío del resultado a la célula δ.

En la máquina de Turing el sistema de operaciones elementales y junto con él el sistema de instrucciones con una sola dirección están aún más simplificados: en cada tiempo aparte la instrucción prescribe sólo la sustitución del único signo is que está en la célula observada por algún otro signo s_j. Si $j = i$, eso quiere decir que el contenido de la célula observada no cambia; si $j = 1$, eso quiere decir que si en la célula observada estaba depositado algún signo, entonces él se extingue. La simplificación siguiente consiste en que cuando la máquina pasa de un tiempo al tiempo inmediato, la dirección de la célula observada puede cambiar en no más de una unidad, o sea, se contemplará la célula vecina de la izquierda, la vecina de la derecha o la misma célula del tiempo anterior.

La idea de esta simplificación consiste en que el contenido de alguna célula, necesario para el proceso, se busca comprobando todas las células una detrás de otra hasta que no se encuentre la que hace falta. Esto, naturalmente, alarga mucho el proceso pero al mismo tiempo da la siguiente oportunidad: en las instrucciones del programa en lugar de direcciones arbitrarias de las células observadas puede limitarse al empleo de sólo tres direcciones estándar que se representan con los siguientes signos:

D significa observar la célula vecina de la derecha,
I, observar la célula vecina de la izquierda,
M, continuar observando la misma célula anterior.

3. Para el tratamiento de la información numérica que se conserva en la memoria, la computadora descrita en los §§ 4—5 tiene una unidad aritmética $\mathfrak{A}$, que puede permanecer en uno del número finito de estados: de sumar, de restar, etc.

Para realizar cualquier operación, por determinados canales a la unidad llegan no sólo los números con los que se ejecuta la operación sino también señales que preparan la unidad para la operación correspondiente, o sea, la pasan al estado correspondiente (véase la fig. 4, *b*). En la máquina de Turing el tratamiento de la información se ejecuta en la *unidad lógica* que también puede estar en uno del número finito de estados; que sean

$$q_1, q_2, \ldots, q_m$$

signos especiales introducidos para designar estos estados. La unidad tiene dos canales de entrada: por uno de ellos en cada etapa del funcionamiento de la máquina (en cada tiempo) entra el signo desde la célula observada, por el otro, el signo q_l del estado que se prescribe a la unidad en el tiempo dado. Por el canal de salida la unidad envía a la célula observada el signo «tratado» s_j correspondiente que es la función unívoca de las señales s_j, q_l que se mandaron a la entrada. Las instrucciones que determinan el funcionamiento de la máquina en cada tiempo se designan:

$$Dq_l,\ Iq_l,\ Mq_l\ (l = 1, 2, \ldots, m),$$

donde el primer signo sustituye la dirección de la célula observada (lo que se explicó anteriormente) y el segundo prescribe a la unidad lógica el estado necesario. Los signos D, I, M, q_1, q_2, . . ., q_m componen el alfabeto interior de la máquina.

La particularidad específica de la máquina de Turing consiste en que la unidad lógica tiene también la tarea de elaborar en cada tiempo dado la instrucción que llegará a la unidad de control al principio del tiempo próximo. Así, pues, la unidad lógica además del canal para la salida del signo s_j tiene dos canales más para enviar dos signos de la instrucción siguiente. El esquema correspondiente se

presenta en la fig. 5. Aquí es importante que los *tres signos de salida* s_j, P, q_l *) dependen exclusivamente de qué *par de entrada* de los signos s_i, q_n ha sido enviada en ese mismo tiempo a la entrada de la unidad. Eso quiere decir que la unidad lógica realiza una función que confronta cada par de signos s_i, q_n (en total hay $k \cdot m$ pares de éstos) con los

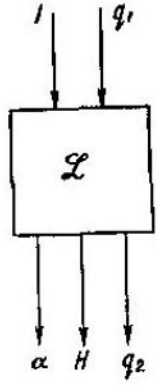


Fig. 5

tres signos s_j, P, q_l. Es cómodo representar esta función, que llamaremos *función lógica* de la máquina, en forma de una tabla rectangular cuyas columnas están numeradas con los signos de los estados y las líneas, con los signos del alfabeto exterior; en cada célula de la tabla están anotados los tres signos de salida correspondientes. Llamaremos a esta tabla el *esquema funcional* de la máquina; en la fig. 6 se presenta un ejemplo de tal esquema.

De la descripción hecha está claro que el *funcionamiento* de la máquina de Turing se determina por completo con la función lógica que realiza la unidad lógica. En otras palabras, dos máquinas de Turing que tengan un esquema funcional común, si nosotros nos interesamos sólo por cómo ellas trabajan, serán indistinguibles. Por otra parte, la estruc-

*) Con P se entiende cualquier de los tres signos D, I, M.

tura de la máquina, la composición de cada uno de sus órganos y su interacción se pueden dar en forma del *esquema estructural* común para todas las máquinas de Turing (véase la fig. 7).

En el esquema indicado se ve la división de la memoria en exterior e interior. La memoria exterior está representa-

	q_1	q_2	q_3	q_4	q_5
Λ	$\Lambda D q_4$	$\Lambda I q_3$	$\Lambda D q_1$	$\Lambda M q_3$	$\Lambda M q_5$
$\|$	$\alpha M q_2$	$\beta M q_1$	$\| D q_1$	$\| I q_1$	$\| M q_5$
α	$\alpha I q_1$	$\alpha D q_2$	$\| I q_3$	$\Lambda D q_4$	$\alpha M q_5$
β	$\beta I q_1$	$\beta D q_2$	$\Lambda I q_3$	$\| D q_4$	$\beta M q_5$

Fig. 6

da con las células de la cinta infinita, destinadas a almacenar la información codificada con símbolos del alfabeto exterior; la memoria interior, con dos células para depositar

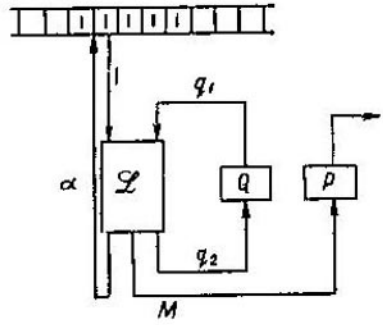


Fig. 7

la instrucción inmediata: la célula Q guarda el signo del estado y la célula P, la señal de desplazamiento de la cinta. En estas dos células tiene lugar la demora de los signos P, q_i, recibidos en la salida de la unidad lógica en el tiempo

dado de funcionamiento, hasta el comienzo del siguiente tiempo cuando ellos llegan a la unidad de control. Las funciones de la unidad de control ahora están extraordinariamente simplificadas y en esencia consisten solamente en asegurar el desplazamiento de la cinta en no más de una célula en concordancia con el signo P que ha llegado. El signo del estado de la unidad lógica de hecho se podría mandar de la célula Q directamente a $\mathfrak{L}$ formando así la línea llamada de retroalimentación por la cual al bloque $\mathfrak{L}$ llega el signo q_i elaborado allí mismo en el tiempo anterior.

El funcionamiento de la máquina de Turing transcurre de la manera siguiente. Antes de ponerla en marcha se anota en la cinta la información inicial (en la fig. 7 ésta es una sucesión de cinco rayitas) y en «el campo visual» de la máquina se establece cierta célula inicial (en la figura es la célula que contiene la cuarta rayita de la derecha); en las células P y Q se introducen los signos del estado inicial y del desplazamiento inicial (supongamos q_1 y M). El proceso posterior transcurre ya automáticamente y se determina unívocamente por el esquema funcional de la máquina. Veamos, por ejemplo, lo que ocurre en el caso cuando haya sido dado el esquema funcional de la fig. 6.

Primer tiempo. Se observa el signo | (rayita) de la célula inicial (desplazamiento M) con el estado q_1. El resultado es los tres signos de salida $\alpha M q_2$, es decir, el signo | ha sido sustituido por el signo α y en las células P y Q se ha depositado hasta el tiempo siguiente la instrucción sucesiva Mq_2.

Segundo tiempo. Se observa el signo α de la misma célula (desplazamiento M) con el estado q_2. Los tres signos de salida son $\alpha D q_2$, o sea, se deja como antes el signo α al pasar a la instrucción Dq_2.

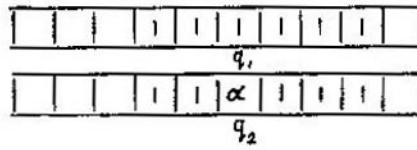
Tercer tiempo. Se observa la rayita de la célula vecina de la derecha (desplazamiento D) con el estado q_2. Como resultado: el signo | se sustituye por el signo β y se pasa a la instrucción Mq_1, etc.

Como se deduce de la última columna del esquema funcional de la fig. 6, la parada de la máquina tendrá lugar sólo bajo la condición de que en cierta etapa del proceso surja el estado q_5. En efecto, cualquiera que sea el signo observado, él no será sustituido por otro y la máquina continuará observándolo (desplazamiento M) con el mismo estado q_5. Esto es el *estado de parada* que signaliza la obten-

ción del resultado y el final del proceso en el caso cuando la máquina *sea utilizable* para la información que se introduce antes de su puesta en marcha.

Un hombre calculador también puede emplear el esquema funcional; para él este esquema representa cierto procedimiento estándar de prefijar el algoritmo de transformación de los datos iniciales, anotados en el alfabeto exterior, al resultado correspondiente, anotado en el mismo alfabeto. Realmente eso es lo que hemos hecho antes con el esquema funcional de la fig. 6 al tratar la palabra de cinco rayitas considerando que la máquina de Turing hubiese hecho lo mismo. A continuación, en casos semejantes para más claridad emplearemos las llamadas *configuraciones*. Entenderemos bajo el término de *configuración* k-ésima la representación de la cinta de la máquina con la información que aparec en ella al comienzo del k-ésimo tiempo, además, aquí debajo de la célula observada se anota el signo del estado de la unidad que se envía a la unidad lógica $\mathfrak{L}$ al comienzo de este tiempo. Así que en la configuración k-ésima se indica evidentemente el par de signos de entrada y, por consiguiente, dirigiéndose al esquema funcional, se pueden determinar los tres signos de salida y asimismo la $(k + 1)$-ésima configuración.

En el ejemplo anterior las primera y segunda configuraciones son:



con los pares de entrada $| q_1$, αq_2, correspondientemente. El paso de la primera configuración a la segunda está relacionado con los tres signos de salida $\alpha M q_2$ que corresponden por el esquema de la fig. 6 al par de entrada $| q_1$.

Acordaremos también aceptar la anotación simplificada de los esquemas funcionales la cual hace el esquema más claro y cómodo para apuntar las configuraciones. Renunciaremos precisamente a la anotación completa de los tres signos de salida $s_j P q_l$ y omitiremos los signos s_j y q_l si no

se diferencian de los signos de entrada correspondientes; también se omitirá el signo M que indica la falta de movimiento de la cinta. Esto, en particular, permite suprimir por completo la columna que corresponde al estado de parada. En la fig. 8 se representa la anotación simplificada del esquema de la fig. 6. En esta nueva anotación el estado de parada lleva el signo «!». En la columna q_1 de la fig. 8 se ve más evidentemente que de la fig. 6 lo que al llegar al

	q_1	q_2	q_3	q_4
Λ	Dq_4	Iq_3	Dq_1	!
1	αq_4	Iq_3	Dq_1	!
α	I	D	I I	Λ D
β	I	D	Λ I	1 D

Fig. 8

estado q_1, observando el signo α, la máquina comienza una serie de desplazamientos hacia la izquierda a través de todos los signos α y β contiguos, quedándose en el estado q_1 y sin cambiar el contenido de las células observadas hasta que en su campo visual aparezca la primera rayita o la primera célula vacía; sólo al darse estas condiciones la máquina saldrá del estado q_1.

A continuación, el signo «!» siempre se empleará para designar el estado de parada.

§ 8. REALIZACION DE ALGORITMOS EN LA MAQUINA DE TURING

En este epígrafe, en una serie de ejemplos mostraremos cómo se construyen máquinas de Turing que realicen algunos algoritmos aritméticos sencillos y cómo transcurre en la máquina el proceso de realización de estos algoritmos; en concordancia con el contenido del epígrafe anterior, bajo el término de construcción de la máquina entenderemos la composición del esquema funcional de la unidad lógica el que

también representa cierta forma estándar de anotación del algoritmo. Además, se explicarán también ciertos razonamientos más generales sobre los métodos de construcción de las máquinas de Turing (de los esquemas funcionales).

I. Algoritmo de paso de n a n+1 en el sistema de numeración decimal

Hay que resolver el siguiente problema:

Se da la anotación decimal del número n (o sea, la representación del número natural *n* en el sistema de numeración decimal); *se exige indicar la anotación decimal del número* $n + 1$.

	q_0	q_1
0	1 !	
1	2 !	
2	3 !	
3	4 !	
4	5 !	
5	6 !	
6	7 !	
7	8 !	
8	9 !	
9	0 /	
Λ	1 !	
1	1	Λ1q_0

Fig. 9

Para eso se toma el alfabeto exterior compuesto de diez cifras 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 y del signo vacío Λ. La máquina puede estar sólo en dos estados: q_0 (el estado de trabajo) y ! (la parada). El número dado *n* lo mismo que el número resultante $n + 1$ se anotarán en el sistema decimal con la particularidad de que las cifras se emplazarán una en cada célula (las células van consecutivamente una detrás de otra sin blanco). El esquema funcional correspondiente se da en la fig. 9 en forma de la parte de la tabla indicada

en ella, que resulta si no se toma en cuenta en esa tabla la última línea y la última columna (el sentido de la tabla ampliada será aclarado algo más tarde). Supongamos que al comienzo del funcionamiento en el campo visual está la cifra del orden de unidades del número n y que la máquina se encuentra en el estado q_0; si esa cifra es diferente de 9, entonces la máquina se interrumpirá inmediatamente después del primer tiempo de su funcionamiento en que tiene

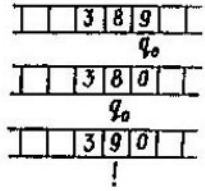


Fig. 10

lugar el cambio de esa cifra por otra en concordancia con el esquema. Si la última cifra es 9, entonces la máquina la sustituye por cero y hace un desplazamiento a la izquierda (hacia el orden vecino más alto) y continúa quedándose en el estado de trabajo (así se asegura el traslado de la unidad a órdenes superiores). Si el número termina con k nueves, entonces acabará su funcionamiento exactamente después del $k + 1$-ésimo tiempo. En la fig. 10 están anotadas las configuraciones correspondientes para el número $n = 389$.

Aclaremos ahora el sentido de la tabla ampliada que viene dada en la fig. 9. Ella prefija el esquema funcional de una máquina que tiene un estado más: q_1; además de eso en su alfabeto exterior tiene un signo más que es precisamente «la rayita». Si al principio de su funcionamiento la máquina está puesta en el estado q_0 y en la cinta no hay rayitas, entonces su actuación transcurrirá exactamente tal y como la de la máquina del ejemplo anterior. Eso es evidente puesto que en las condiciones indicadas la última línea y la última columna de la tabla no toman ninguna participación en el funcionamiento descrito. Esto, en particular, quiere decir que la máquina dada puede también ser empleada para la realización del algoritmo anterior.

No obstante, esta máquina es apta para hacer alguna otra cosa más y precisamente por eso nos hemos puesto a examinarla.

Supongamos que en la cinta se da la anotación decimal del número n y en varias células consecutivas situadas a la derecha de esta anotación están apuntadas rayitas de una en una en cada célula. Veamos cómo va a actuar la máquina con este esquema funcional si al comienzo de su trabajo en el campo visual se establece la rayita extrema de la derecha y la propia máquina se encuentra en el estado q_1. En el primer tiempo (el par de entrada es q_1 |) se borra esta rayita; también tiene lugar un desplazamiento a la izquierda y el paso al estado q_0 (los tres signos de salida son $\wedge$ $1q_0$). En los tiempos siguientes la máquina continúa los desplazamientos a la izquierda estando en el estado q_0 a través de todas las rayitas hasta la primera cifra del orden de las unidades. Comenzando desde este momento todo transcurre ya como en el algoritmo anterior, o sea, tiene lugar la transformación de la anotación del número n en la del número $n + 1$ y el proceso se termina.

Resumiendo, la máquina disminuye en una unidad el número de rayitas y en anotación decimal realiza el paso del número n al $n + 1$. Acordaremos llamar a este proceso el *paso controlado* de la anotación decimal de n a la anotación decimal de $n + 1$.

En la fig. 11 están anotadas las configuraciones para un conjunto de cinco rayitas y para $n = 389$.

II. Algoritmo de conversión al sistema de numeración decimal

Crearemos un esquema funcional de una máquina (algoritmo) que resuelva problemas del tipo siguiente:

Se da cierta cantidad finita de rayitas anotadas en células que van una detrás de otra sin blancos entre ellas (llamaremos a éstas *conjunto de rayitas*); *hay que apuntar en el sistema decimal el número de rayitas.*

Dicho brevemente: hay que contar las rayitas del conjunto.

Tal esquema se representa en la fig. 12. Para convencerse de que este esquema realmente describe la máquina (el algoritmo) necesaria, es útil compararlo con el esquema de

la tabla ampliada de la fig. 9. La columna q_0 del esquema de la fig. 12 se diferencia de la columna q_0 en el esquema de la fig. 9 sólo en que en lugar del estado «!» en él en todos los sitios figura el nuevo estado q_2; la diferencia entre las columnas q_1 para el funcionamiento del esquema de la fig. 9 no tiene una importancia esencial. Por eso, si en la cinta

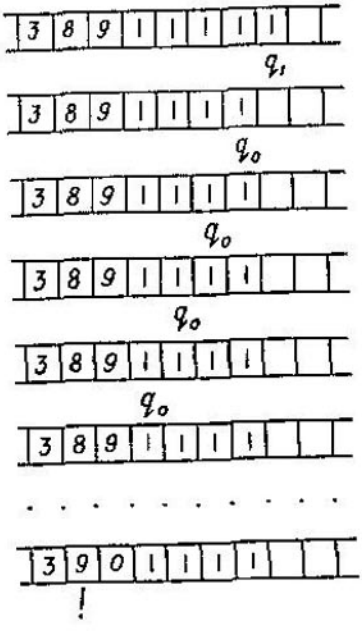


Fig. 11

	q_0	q_1	q_2
0	$1\ q_2$	!	D
1	$2\ q_2$	!	D
2	$3\ q_2$	!	D
3	$4\ q_2$	!	D
4	$5\ q_2$	!	D
5	$6\ q_2$	!	D
6	$7\ q_2$	!	D
7	$8\ q_2$	!	D
8	$9\ q_2$	!	D
9	$0\ 1$	!	D
Λ	$1\ q_2$	!	Iq_1
1	I	$\Lambda I q_0$	D

Fig. 12

se dan la anotación decimal del número n y a la derecha de ésta un conjunto de rayitas, y si en el campo visual de la máquina, como antes, se coloca la rayita que está más a la derecha y la propia máquina está puesta en el estado q_1, entonces en la máquina al principio tendrá lugar el mismo proceso que el del esquema de la fig. 9; se borrará precisamente la rayita del conjunto y la anotación del número n será sustituida por la del número $n + 1$. Sin embargo, mientras que, conforme al esquema de la fig. 9, en esta etapa del proceso aparece el estado !, o sea, el proceso se interrumpe,

aquí, de acuerdo con el esquema de la fig. 12, aparece el estado q_2 y el proceso continúa. En particular, si la primera configuración se toma como en la fig. 11, la octava configuración resulta ya tal como la de la fig. 13. En la columna del estado q_2 se puede ver cómo continuará el proceso: comenzará una serie de desplazamientos hacia la derecha a través de todas las cifras y todas las rayitas hasta que

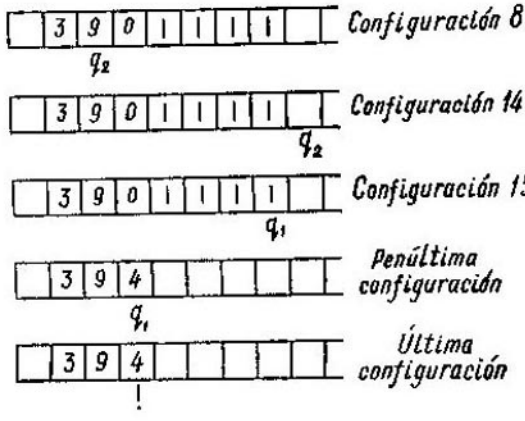


Fig. 13

se llegue a la primera célula vacía con el par de entrada $\wedge\ q_2$ (véase la configuración 14 en la fig. 13), después sigue un desplazamiento hacia la izquierda con el paso simultáneo al estado q_1 (la configuración 15 de la fig. 13). De esta manera, en el campo visual de la máquina de nuevo aparece la rayita que está situada más a la derecha del conjunto con el estado q_1. Así acaba un ciclo de funcionamiento y comienza el segundo, análogo al primero. Como resultado del segundo ciclo se borrará una rayita más y la anotación del número $n + 1$ se sustituirá por la anotación del número $n + 2$. Si en el conjunto al principio había k rayitas, después de k ciclos de funcionamiento se borrarán todas ellas y en lugar de la anotación inicial del número n aparecerá la anotación del número $n + k$. Al concluir el k-ésimo ciclo la máquina de nuevo llegará al estado q_1 pero en su campo visual ya no habrá una rayita (ya todas estarán borra-

das) sino que estará la primera cifra, es decir, la de las unidades, de la anotación del número $n + k$ (la penúltima configuración de la fig. 13). Como se ve en el esquema de la fig. 12, en este caso el funcionamiento se interrumpe (última configuración de la fig. 13).

De todo lo dicho se deduce que si al comienzo del funcionamiento de la máquina en la cinta están anotados la cifra 0 y un conjunto de k rayitas, pues la máquina borrará todas las rayitas y en lugar del cero aparecerá la anotación decimal del número $0 + k$, o sea, el número k. De hecho, al comienzo del funcionamiento se puede pasar sin cero, puesto que si en lugar del cero figura el signo Λ, entonces con los estados q_0 y q_1 la máquina se conduce de tal modo como si fuese un cero (véase el esquema de la fig. 12). Así, pues, el esquema propuesto de la fig. 12 realmente describe un algoritmo de conversión de un conjunto de rayitas a la anotación decimal de su número.

Ejercicio. Componer por analogía con el I un esquema funcional de una máquina (de un algoritmo) que realice el paso de la anotación decimal del número n a la anotación decimal del número $n - 1$ (siempre que $n \geqslant 1$). Continuando, por analogía con el II componer un esquema funcional de una máquina para pasar del sistema decimal, o sea, para la conversión de la anotación decimal de cualquier número n a un conjunto de n rayitas.

Veremos algunos ejemplos más de máquinas de Turing para resolver problemas aritméticos. En estos problemas tanto los datos iniciales (las condiciones del problema) como el resultado serán números naturales. Acordaremos conisderar que cada número natural se introduce en la máquina en forma de un conjunto del mismo número de rayitas. Si en el problema figuran varios números naturales, entonces separaremos los conjuntos de rayitas que los representan con algún signo especial, por ejemplo, con un asterisco *. Este signo también entra en el alfabeto exterior de la máquina.

III. Algoritmo de adición

En la cinta se introducen dos números, por ejemplo,

Como resultado se debe obtener su suma, en este caso

Observemos que no se pueden reducir las acciones de la máquina sencillamente a borrar el asterisco, porque entonces surgiría una célula vacía entre las rayitas y, en consecuencia, no se podría considerar el resto de las rayitas como la representación de un número natural. El funcionamiento

	q_0	q_1	q_2
1	$\Lambda D q_2$	I	D
Λ	D	$D q_0$	$1 q_1$
*	Λ !	I	D

Fig. 14

de la máquina de acuerdo con el esquema funcional supuesto (véase la fig. 14) transcurrirá de la siguiente manera.

Condiciones iniciales: en el campo visual está emplazada la rayitá extrema de la izquierda y la máquina se encuentra en el estado q_0 (la configuración 1 de la fig. 15).

Primer tiempo. Se borra la rayita observada, desplazamiento a la derecha (en el campo visual aparece la rayita siguiente) y paso al estado q_2 (configuración 2).

Como se deduce de la columna q_2, los tiempos siguientes se reducen a desplazamientos hacia la derecha a través de todas las rayitas y del asterisco hasta que no se llegue a la primera célula vacía (configuración 12); entonces (el par de entrada es $\Lambda\ q_2$) en esta célula vacía se inscribe una rayita y la máquina pasa al estado q_1 (configuración 13). Con el estado q_1 tienen lugar desplazamientos hacia la derecha a través de todas las rayitas y del asterisco hasta la primera célula vacía de la izquierda (configuración 24); entonces (el par de entrada es $\Lambda\ q_1$) ocurre un movimiento a la derecha, en el campo visual se establece la primera de las rayitas que quedan a la izquierda del asterisco y la máquina pasa al estado q_0 (configuración 25). Como conse-

cuencia de este ciclo una rayita del sumando de la izquierda resulta traspasada al de la derecha. Si a la izquierda del asterisco al principio había k rayitas, después de k ciclos todas se traspasarán a su parte derecha. En el $(k + 1)$-ésimo

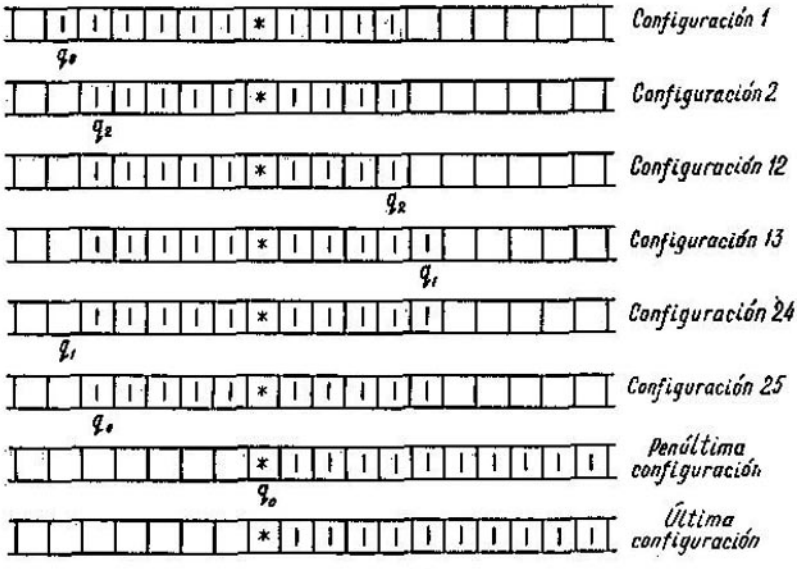


Fig. 15

desplazamiento a la derecha, en el campo visual de la máquina con el estado q_0 ya no se encontrará una rayita (ya no hay rayitas a la izquierda del asterisco) sino que el mismo asterisco (la penúltima configuración). Entonces (el par de entrada es $*q_0$) se borra el asterisco y la máquina se para (última configuración). Junto con esto ya se obtiene la suma que se busca.

IV. Algoritmo de la adición repetida y de la multiplicación

Veamos qué cambios hay que hacer en el esquema de la fig. 14 para que después de la introducción inicial en la cinta del par de números m, n, por ejemplo,

la máquina dé a un proceso infinito que consiste en que el número de la izquierda m se suma al de la derecha y después se adiciona a la suma obtenida $n + m$, después, de nuevo se adiciona a la suma $n + 2m$ y así se continúe sin fin. Es evidente que para esto hace falta que el sumando izquierdo no desaparezca por completo después de la primera suma,

	q_0	q_1	q_2	q_3
1	$\alpha D q_2$	I	D	
$\wedge$	D	$D q_0$	$1 q_1$	$D q_0$
*	q_3	I	D	I
α	D	$D q_0$	$1 q_1$	$1 I$

Fig. 16

sino al revés, que se lo pueda restablecer después de cado suma para adicionarlo de nuevo al número representada a la derecha del asterisco. Eso se puede conseguir haciendo, por ejemplo, que no se borren las rayitas del conjunto izquierdo y que se sustituyan temporalmente por algún signo o marca. En la fig. 16 se representa un esquema en el que la letra α hace el papel de tal marca. En concordancia con esto la entrada del signo vacío $\wedge$ en la primera línea del esquema 14 corresponde a la entrada del signo α en la primera línea del esquema 16; además, en el esquema de la fig. 16 a la letra α se le dedica una línea más (la cuarta) en la cual las tres primeras células contienen los mismos datos que las correspondientes de la línea de $\wedge$ en la fig. 14.

Y continuando, para que el proceso no se interrumpa después de la primera suma, es necesario que en el esquema de la fig. 16, en lugar del signo «1» que aparece en el esquema de la fig. 14 con el par de entrada $*q_0$, se contenga el signo de otro estado que garantice la continuación del proceso. En nuestro caso, tal será el estado q_3 introducido complementariamente. Para su signo habrá que dedicar su columna correspondiente.

En el esquema de la fig. 16 no hay signo «l» (parada) por eso el proceso que se describe no tiene fin. Ahora el lector sin gran dificultad comprobará que la máquina correspondiente precisamente realiza una ilimitada cantidad de veces la adición del número representado a la izquierda del asterisco al número de su derecha. Si a la derecha del asterisco al comienzo del funcionamiento no hubiesen rayitas (o sea, el segundo número fuese cero), entonces a la derecha del asterisco aparecerían m rayitas, después $2m$ rayitas, después $3m$ rayitas, etc., y así sin fin.

Ejercicio. Componer un esquema funcional para el algoritmo de la multiplicación.

Indicación. Tomar como base el esquema anterior y modificarlo de tal manera que el proceso de repetición de la suma no continúe de un modo ilimitado, sino que se cumpla tantas veces como rayitas haya en el multiplicador (después de cada ciclo de suma se borra una rayita del conjunto que representa el multiplicador).

V. El algoritmo de Euclides

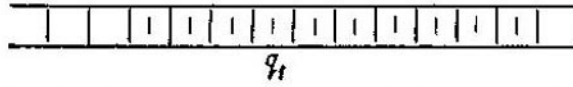
Examinemos ahora cómo se verá en una máquina de Turing el algoritmo de Euclides para el cálculo del máximo común divisor de los números a y b. Ya hemos descrito este algoritmo dos veces: la primera vez en forma de prescripción de palabra y la segunda, en forma de un programa para una computadora de control automático. Esta vez presentaremos el algoritmo en forma de un esquema funcional de una máquina de Turing y observaremos el proceso de cálculo en la máquina. Este proceso se compone de *ciclos de comparación* y *ciclos de sustracción* que se turnan. Ellos corresponden a las *operaciones elementales de comparación* y *de sustracción* de la computadora. El esquema funcional correspondiente está representado en la fig. 8; su alfabeto exterior se compone de cuatro signos

$$\Lambda\ ,\ |\ ,\ \alpha,\ \beta.$$

Los números naturales se representarán como antes con sus correspondientes conjuntos de rayitas. Para evitar detalles que no están ligados a la esencia de la cosa y que solamente complicarían nuestro examen, acordaremos emplazar los conjuntos de rayitas que representan los dos números

dados en la cinta uno inmediatamente detrás del otro sin separarlos con el asterisco y, además, tendremos en cuenta que al comienzo del proceso en el campo visual de la máquina está colocada la rayita más de la derecha del conjunto del primer número. Después de un detallado análisis que será hecho posteriormente, el lector, en calidad de ejercicio, podrá fácilmente modificar el esquema funcional propuesto de tal manera que garantice un funcionamiento correcto de la máquina también al plantear las condiciones del problema de otro modo (por ejemplo, si los conjuntos están separados con un asterisco y en el campo visual de la máquina se establece alguna célula vacía). Observemos además, que las letras α, β desempeñarán el papel de marcas temporales que hace un calculador (corrientemente en forma de tildes o signos marginales) para recordar algunas circunstancias que surgen en el curso del cálculo.

Acompañaremos la descripción ulterior con la ilustración en un ejemplo para el caso de $a = 4$, $b = 6$ por medio de configuraciones. La primera configuración tendrá la forma:



En el ciclo de comparaciones participan sólo los estados q_1, q_2; en el ciclo de sustracción participan q_3 y q_4.

Seguiremos ahora el proceso con más detalle. Al principio la máquina compara los números representados en la cinta para establecer cuál de ellos es mayor. Al hacer esto la máquina se comporta de la misma manera que obraría una persona al comparar dos largas series de unidades que son difícil de contemplarlas completas. Es decir, el hombre marca de alguna manera cada unidad alternativamente en las dos series (por ejemplo, con algún signo), al acabarse así una de las series, se aclara cuál de ellas se compone de un número mayor de unidades.

La máquina sustituye la rayita del primer número con el símbolo α, después sustituye la rayita del segundo con el símbolo β, después de nuevo vuelve a las rayitas del primer número y sustituye una más con el símbolo α, después sustituye otra rayita más del segundo número con el símbolo β, etc.

En los primeros cuatro tiempos en la cinta se crean las configuraciones representadas en la fig. 17, o sea, al final del cuarto tiempo la máquina ya ha marcado una rayita de cada número y ahora comienza el desplazamiento hacia la izquierda en busca de la rayita más cercana, todavía no marcada del número izquierdo. Después de varios tiempos más en la cinta surge la configuración I de la fig. 18; el primer número ya se ha acabado y el segundo todavía no.

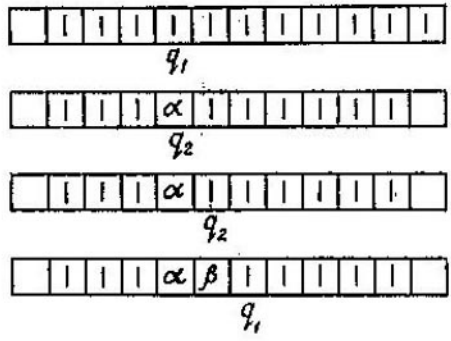


Fig. 17

Al buscar rayitas a la izquierda no aparecerá ninguna y así se llegará y la configuración II; el ciclo de comparación ya se ha realizado solamente con la participación de los estados q_1 y q_2. El tiempo siguiente da ya la configuración III.

Como se ve en la columna del estado q_4 del esquema representado en la fig. 8, ahora comenzará un movimiento hacia la derecha con la sustitución de todas las α por signos vacíos $\wedge$ (es decir, borrando todas las α) y con la sustitución de todas las β por rayitas. Después de que la última β de la derecha sea sustituida por una rayita, en la cinta aparecerá la configuración IV de la fig. 18 y a continuación, la configuración V. Así, después del ciclo de comparación tiene lugar el ciclo en el que el primer número se sustrae del segundo; como resultado de este ciclo el número menor a se borra y el número mayor b se divide en a, $b - a$; aquí se observa la última rayita del primero de estos números y la máquina de nuevo llega al estado q_1. Esto quiere decir que l problema inicial para los números a, b ha sido reducido

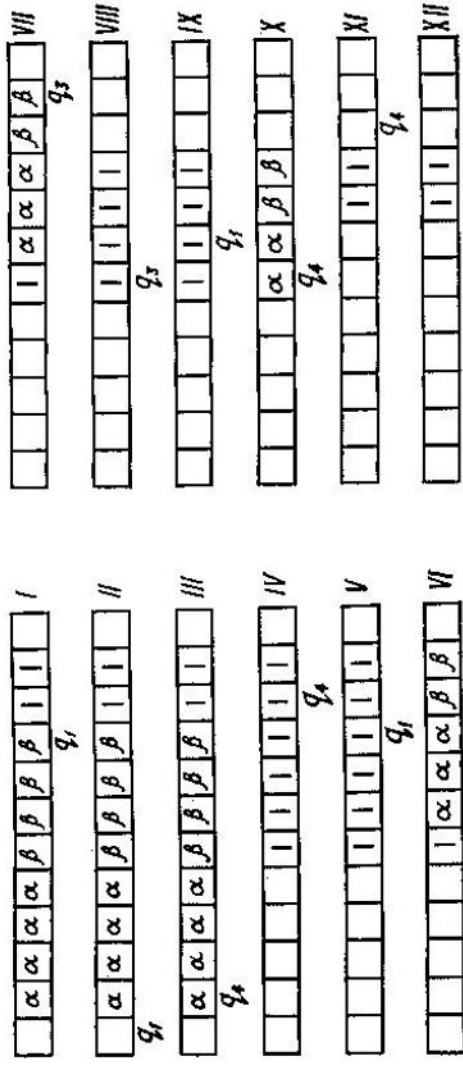

I
II
III
IV
V
VI
VII
VIII
IIII
IX
X
XI
XII
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
q_{10}
q_{11}
q_{12}
q_{13}
α
β
i

Fig. 18

al mismo problema pero para los números a, $b - a$. Precisamente el algoritmo de Euclides, como ya sabemos, está basado en esto.

Ni que decir tiene que más adelante aparecerá de nuevo el ciclo de comparación. Ahora, empero, termina al acabar el segundo número (de la derecha) que es el que esta vez resulta el menor. Lo último se revela después que la máquina al sustituir tres rayitas del primer número no encuentre ya rayitas en el segundo, o sea, surge la configuración VI.

El tiempo siguiente genera la configuración VII y con él comienza el ciclo de sustracción del segundo número del primero, es decir, se borran todas las β y se sustituyen todas las α por rayitas. Después de la sustitución de la última α de la izquierda por una rayita aparece en la cinta la configuración VIII; luego la IX con lo que acaba el ciclo de sustracción y comienza el siguiente ciclo de comparación, etc. Este proceso continúa hasta que el problema se reduzca al caso de dos números iguales entre sí (en nuestro ejemplo eso ya se ha alcanzado). Entonces comienza el último ciclo de comparación que debe llevar a la terminación resultativa del proceso. En efecto, después de que se ha obtenido la configuración X, por medio de la sustracción se genera la configuración XI, y, al fin, la configuración resultativa XII.

VI. Combinación de algoritmos

Resulta oportuno, para componer esquemas funcionales nuevos, el empleo de otros esquemas que fueron creados anteriormente; eso es posible en los casos en los que se examina un algoritmo que en cierto sentido es una combinación de algoritmos estudiados anteriormente. Aclararemos esto con un ejemplo. Supongamos que haya que componer un esquema funcional de algoritmo que transforme un par de números a, b dados con sus correspondientes conjuntos de rayitas, en su máximo común divisor anotado en el sistema de numeración decimal. Este algoritmo puede ser obtenido como resultado de *composición*, o sea, de la aplicación consecutiva de dos algoritmos examinados anteriormente: primero el V y después el II. Se puede también obtener su esquema funcional representado en la fig. 19, por medio de una combinación adecuada de los esquemas de la fig. 8 y de la fig. 12. Lo primero que debemos hacer para eso es

cambiar los signos de los estados del esquema en la fig. 12 por p_0, p_1, p_2 para poder diferenciarlos de los estados de la fig. 8; en el esquema de la fig. 8 debemos cambiar el signo de parada «!» por p_2. Después reunimos estos esquemas corregidos en un solo esquema, tal como se indica en la fig. 19.

	q_1	q_2	q_3	q_4	p_0	p_1	p_2
0					1 p_2	!	D
1					2 p_2	!	D
2					3 p_2	!	D
3					4 p_2	!	D
4					5 p_2	!	D
5					6 p_2	!	D
6					7 p_2	!	D
7					8 p_2	!	D
8					9 p_2	!	D
9					0 1	!	D
Λ	Dq_4	Iq_3	Dq_1	p_2	1 p_2	!	Ip_1
1	αq_2	βq_2	Dq_1	Iq_1	I	ΛIp_0	D
α	I	D	1 I	Λ D			
β	I	D	ΛI	1 D			

Fig. 19

(Las células que han quedado sin llenar, corresponden a los pares de entrada que no participarán en el proceso que nos interesa; en estas células se pueden inscribir cualesquiera tres datos de salida.) Ahora ya no es difícil comprobar que, en concordancia con el esquema de la fig. 19, al principio transcurrirá el proceso de transformación de los dos conjuntos de rayitas dados hasta que en la cinta aparezca el conjunto que representa el máximo común divisor. Empero, en este momento en lugar del signo de parada ! (véase, por ejemplo, la configuración XII en la fig. 18) aparecerá ahora el estado p_2 y el proceso continuará asegurando la transformación posterior de este conjunto a la anotación del número en el sistema decimal.

Se entiende fácilmente que este método se puede extender al caso de composición con cualquier número finito de algoritmos.

De esto, en particular, se deduce que al examinar algoritmos numéricos se puede considerar que los números naturales se presentan con conjuntos de rayitas ya que si se compone con esta suposición el correspondiente esquema $\mathfrak{A}$ se puede fácilmente pasar de él al esquema $\mathfrak{B}$ aplicable al sistema de numeración decimal. Para eso es suficiente hacer el esquema $\mathfrak{B}$ a base del método indicado anteriormente, por medio de la combinación de tres algoritmos: el algoritmo de conversión del sistema decimal, el algoritmo $\mathfrak{A}$ y el algoritmo de conversión al sistema decimal*).

Otro procedimiento de combinación de algoritmos es la reiterada repetición del empleo de un mismo algoritmo hasta que se cumpla cierta condición indicada anticipadamente. Por ejemplo, el algoritmo de conversión al sistema decimal se reduce al repetido empleo del algoritmo del paso controlado de n a $n + 1$ hasta que aparezcan todas las rayitas borradas. A base del esquema funcional del algoritmo dado (si es que ya se ha elaborado) y de la condición presentada se puede crear un esquema de un algoritmo cuyo funcionamiento se repita; empero, este método es más complicado que el del caso de composición y aquí al comentarlo no vamos a entrar en detalles.

De los ejemplos vistos se hace suficientemente claro cómo elaborar esquemas funcionales para otros algoritmos también y, en particular, para no numéricos. Indicaremos el plan general de elaboración de un esquema funcional para el algoritmo de reducción de palabras (véase el ejemplo 4 del § 3). Primero se elaboran los esquemas $\mathfrak{A}_1$, $\mathfrak{A}_2$, $\mathfrak{A}_3$, $\mathfrak{A}_4$ que realizan las sustituciones orientadas:

$$b \rightarrow acc$$
$$ca \rightarrow accc$$
$$aa \rightarrow \Lambda$$
$$cccc \rightarrow \Lambda,$$

*) Aquí sin quererlo surge la comparación con las computadoras electrónicas que funcionan en el sistema de numeración binario. Ellas tienen un dispositivo para la conversión de los datos iniciales del sistema decimal al binario y un dispositivo que pasa el resultado final de nuevo al sistema decimal.

correspondientemente. Por ejemplo, la máquina del esquema $\mathfrak{A}_4$ transforma cualquier palabra que esté en el alfabeto $\{a, b, c\}$ grabada en la cinta, en una palabra obtenida borrando las cuatro letras c que se encuentren juntas (una al lado de otra) en el extremo de la izquierda; si no hay tales, entonces la palabra se queda como era. Después se componen los esquemas $\overline{\mathfrak{A}}_1$, $\overline{\mathfrak{A}}_2$, $\overline{\mathfrak{A}}_3$, $\overline{\mathfrak{A}}_4$ para la repetición de los algoritmos $\mathfrak{A}_1$, $\mathfrak{A}_2$, $\mathfrak{A}_3$, $\mathfrak{A}_4$, correspondientemente. Así por ejemplo, la máquina del esquema $\overline{\mathfrak{A}}$ borra cuatro letras c, después otras cuatro, etc., hasta que se cumpla la condición: se ha obtenido una palabra en la que no hay cuatro letras c anotadas una junto a otra. Y, al fin, el esquema tratado para el algoritmo de reducción representa una composición de los esquemas $\overline{\mathfrak{A}}_1$, $\overline{\mathfrak{A}}_2$, $\overline{\mathfrak{A}}_3$, $\overline{\mathfrak{A}}_4$.

§ 9. HIPOTESIS BASICA DE LA TEORIA DE LOS ALGORITMOS

El examen de los ejemplos anteriores da la impresión de que el proceso que tiene lugar en una máquina de Turing es como el rodaje lento de una película de cine en la cual se muestra el proceso de cómputo que cumple el hombre de acuerdo con cierto algoritmo. Al mismo tiempo estos ejemplos nos dan la idea de presentar por medio de esquemas funcionales de máquinas de Turing también otros conocidos algoritmos que corrientemente se dan de otra manera, por ejemplo, en forma de una prescripción de palabra o de algunas fórmulas especiales. Ya en esta etapa de nuestro estudio parece muy verosímil que esto se debe poder lograr también en otros casos. ¿Es eso realmente así? ¿En qué grado es general el concepto de máquina de Turing y de esquema funcional de Turing? ¿Se puede considerar que el procedimiento de presentación de algoritmos por medio de esquemas funcionales es universal en el sentido de que cualquier algoritmo puede ser dado de esa manera? A estas preguntas la teoría contemporánea de los algoirtmos propone una contestación en forma de la siguiente hipótesis:

Hipótesis básica de la teoría de los algoritmos. *Cualquier algoritmo puede ser presentado por medio de un*

esquema funcional de Turing y realizado en la correspondiente máquina de Turing.

Veremos aquí dos cuestiones que surgen con motivo de la definición de la hipótesis:

1. ¿En qué consiste la importancia de esta hipótesis para la teoría de los algoritmos?

2. ¿En qué consiste la fundamentación de la hipótesis?

Ante todo pongamos atención en la siguiente particularidad característica de la definición citada de la hipótesis. En esta definición por una parte se habla sobre cualquier algoritmo, o sea, sobre el concepto general de algoritmo que, como ya se ha subrayado varias veces, no es un concepto matemático exacto; por otra parte, en esta misma definición se trata de un concepto matemático tan exacto como el esquema funcional de Turing. La importancia de la hipótesis consiste justamente en que precisa el concepto general pero vago de «cualquier algoritmo» por medio del concepto matemático de esquema funcional de Turing que es más especial y completamente exacto (y de su realización en una máquina de Turing); así, la teoría de los algoritmos anuncia en calidad de objeto de su investigación cualquier esquema funcional de Turing (o máquinas de Turing). Al mismo tiempo ya se hacen comprensibles los planteamientos de tales cuestiones como las de existencia o no existencia de un algoritmo resolutivo para los problemas de uno o de otro tipo. Precisamente ahora esto debe comprenderse como la cuestión sobre la existencia o la no existencia de una máquina de Turing (del esquema funcional) que tenga las propiedades necesarias.

Resumiendo, la hipótesis formulada justifica la aceptación de la definición básica de la teoría contemporánea de los algoritmos de acuerdo con la que el concepto vago de algoritmo se identifica con el concepto exacto de esquema funcional de la máquina de Turing.

¿Y con todo, en qué consiste la fundamentación de esta hipótesis tan importante?

Antes observemos que no se puede tratar de demostrar esta hipótesis como generalmente se demuestran los teoremas en las matemáticas. En efecto, la definición de la hipótesis no tiene carácter de teorema, puesto que ella representa la afirmación sobre el *concepto general de algoritmo* que no es un concepto matemático exacto y, en conse-

cuencia, no puede ser objeto de razonamientos matemáticos estrictos.

El convencimiento de la justedad de la hipótesis está basado sobre todo en la experiencia. Todos los algoritmos conocidos que fueron creados en el curso de muchos milenios de la historia de las matemáticas pueden ser presentados por medio de esquemas funcionales de Turing. Es verdad que el contenido de la hipótesis no está dirigido solamente al pasado y no se limita a constatar el hecho de que para todos los algoritmos conocidos ha resultado posible componer sus esquemas funcionales. El contenido de la hipótesis tiene también un carácter completamente claro de pronóstico para el futuro: siempre que en el futuro se reconzca alguna prescripción como algoritmo, independientemente de la forma y los medios con que esta prescripción inicialmente se exprese, también se la podrá presentar con un esquema funcional de la máquina de Turing.

En este sentido la hipótesis básica puede compararse con una ley física, por ejemplo, con la ley de conservación de la energía basándose en cual también se hacen pronósticos para el futuro. La gran experiencia práctica del pasado se reconoce como suficiente fundamento para similares pronósticos.

Hay también otras consideraciones que confirman la justedad de la hipótesis básica.

En el epígrafe anterior se indicaron dos procedimientos de composición de algoritmos complicados con otros algoritmos iniciales más sencillos. Estos procedimientos son la combinación de algoritmos y la repetición de un algoritmo. Se podría continuar la lista de estos procedimientos. No obstante, todos los procedimientos semejantes conocidos y también todos aquellos que se pueden esperar, teniendo en cuenta el estado de desarrollo actual de la ciencia, resultan tales que, si para los algoritmos iniciales es posible su representación por medio de esquemas funcionales, entonces ella es posible también para los algoritmos resultativos que son más complicados. En particular, para el caso de combinación de algoritmos se mostró cómo se construye a base de esquemas funcionales dados un esquema nuevo. Recordemos, además, la siguiente circunstancia que se vio de paso en el § 6. Cuando en la ciencia surgió una aguda necesidad de elaborar el concepto exacto de *algoritmo*,

muchos matemáticos comenzaron investigaciones con el fin de encontrar cierta forma general de presentación de algoritmos que fuese la suficientemente estricta para que se pudiese hacer un objeto de estudios matemáticos y lo suficiente universal para que se les pudiese dar tal forma a todos los algoritmos que se puedan imaginar. Además de los esquemas funcionales de las máquinas de Turing, fueron propuestos otros métodos de precisión de este concepto. Por ejemplo, A. A. Márkov llegó al concepto de *algoritmo normal* (el que en este libro se ilustró de pasada en el algoritmo de reducción para el cálculo asociativo del ejemplo 4 en el § 3); Gödel y Kleene llegaron a la noción de *algoritmo recurrente* (*función recurrente*), etc. Empero, después todas estas precisiones resultaron equivalentes. Este hecho no se puede considerar casual; esto es un argumento más a favor de la hipótesis formulada.

Al concluir, advirtamos además que en la misma teoría de los algoritmos la hipótesis básica no se emplea; en esta teoría al demostrar teoremas no se hace ninguna clase de referencias a la hipótesis básica. Así, una persona que no conociese esta hipótesis o que no reconociese convincentes los argumentos que hemos expuesto a favor de su justedad, no sentiría por eso ninguna dificultad formal en el estudio o empleo de la teoría contemporánea de los algoritmos. No obstante, para esta persona lo que nosotros llamamos *teoría de los algoritmos* sería nada más que la teoría de los esquemas funcionales de las máquinas de Turing, en esencia, sería solamente la *teoría de ciertos algoritmos especiales*.

El autor de estas líneas plenamente comparte la convicción de la justedad de la hipótesis básica y la apreciación que se deduce de ella de la teoría contemporánea de los algoritmos como una teoría general que define la propia naturaleza de las cosas y no sólo como una teoría de una clase artificialmente seleccionada de «algoritmos de Turing» especiales.

§ 10. LA MAQUINA UNIVERSAL DE TURING

Hasta ahora nos ateníamos al punto de vista de que diversos algoritmos se realizan en diferentes máquinas de Turing que se distinguen entre sí por sus esquemas funcio-

nales. Sin embargo se puede componer una máquina de Turing *universal*, en cierto sentido capaz de cumplir cualquier algoritmo, lo que quiere decir, capaz de cumplir el trabajo de cualquier máquina de Turing.

Para aclararse mejor cómo se haría esto nos representaremos el experimento siguiente. Supongamos que en la cinta de la máquina se introduce la información inicial $\mathfrak{A}$ y que a cierta persona se le propone indicar cómo va a tratar la máquina esta información y qué será lo que elaborará como resultado. Si esta persona conoce los principios del funcionamiento de las máquinas de Turing, será suficiente comunicarle, además de esta información inicial $\mathfrak{A}$, también el esquema funcional de la máquina. Entonces esta persona *copiando* el funcionamiento de la máquina y anotando las configuraciones necesarias, como lo hemos hecho al examinar el algoritmo de Euclides, podría obtener el mismo resultado que ella. Eso significa precisamente que tal persona es capaz de cumplir la función de cualquier máquina de Turing si se le presenta su esquema funcional. El propio proceso de imitación de las acciones de la máquina en concordancia con su esquema funcional puede ser reglamentado en forma de una prescripción exacta que se pueda comunicar a una persona que no tenga ni la menor idea de lo que son las máquinas de Turing. Si a una persona que tenga una tal prescripción, la que es natural denominar *algoritmo de imitación*, se le suministra un esquema funcional de alguna máquina de Turing y, además, cierta configuración inicial representada en la cinta, pues entonces ella se encontrará apta para copiar exactamente el funcionamiento de la máquina que se trata y al final obtener el mismo resultado que la máquina. Se podría presentar un algoritmo de imitación semejante aunque sea en forma del siguiente sistema de indicaciones:

Indicación 1. Observa en la cinta la célula (la única) bajo la cual está escrita una letra.

Indicación 2. Encuentra en la tabla *) la columna designada con la misma letra que está escrita debajo de la célula observada.

Indicación 3. En la columna que se ha encontrado fíjate en las tres letras situadas en el cruce con la línea designada con la misma letra que está inscrita en la célula observada.

*) O sea, en el esquema funcional.

Indicación 4. Sustituye la letra de la célula observada por la primera letra de las tres observadas de la tabla.

Indicación 5. Si la segunda letra de las tres observadas es !, entonces para; el proceso está terminado.

Indicación 6. Si la segunda letra de las tres observadas es *M*, entonces sustituye la letra *anotada* debajo de la célula observada por la tercera letra de esas tres.

Indicación 7. Si la segunda letra de las tres observadas es *I*, entonces borra la letra anotada debajo de la célula que observamos y *a su izquierda* apunta la tercera letra de esas tres.

Indicación 8. Si la segunda letra de las tres observadas es *D*, entonces borra la letra anotada debajo de la célula que observamos y *a su derecha* apunta la tercera letra de esas tres.

Indicación 9. Pasa a la indicación 1.

Ahora bien, resulta que en el lugar de una persona que actúa de acuerdo con el algoritmo se puede colocar cierta máquina de Turing. Esta será una máquina de Turing universal, capaz de imitar el funcionamiento de cualquier otra máquina de Turing. Dicho de otra forma, esto significa que el algoritmo de imitación que hemos descrito de palabra anteriormente con un sistema de nueve indicaciones, puede ser de manera adecuada presentado en forma de cierto esquema funcional de Turing (de un esquema universal). La demostración completa y estricta de este hecho que representa una confirmación más de la hipótesis básica de la teoría de los algoritmos, es demasiado grande con sus detalles para que la podamos introducir en este pequeño libro. Nos limitaremos a hacer ciertas aclaraciones generales que seguramente serán suficientes para entender la esencia de la cosa.

Advertamos ante todo que en el algoritmo de imitación que hemos descrito, en calidad de los datos iniciales (de la información inicial) figuran el esquema funcional de la máquina imitada y la configuración inicial correspondiente. Esta información inicial es transformada por el algoritmo a la configuración final que representa el resultado que daría la máquina imitada. La máquina universal debe hacer lo mismo. No obstante, aquí hay que tener en cuenta las dos circunstancias siguientes:

1. No se puede realizar la introducción directa en la cinta de la máquina universal del esquema funcional de la máquina imitada y de la correspondiente configuración, en calidad de información inicial. En efecto, en la máquina universal, como en cualquier otra máquina de Turing, la información se representa con letras dispuestas en la cinta *en forma unidimensional*, o sea, en una línea, formando una o varias palabras en el alfabeto exterior de la máquina. Al mismo tiempo, hasta ahora nosotros hemos presentado los esquemas funcionales por medio de tablas «bidimensionales» en las cuales las letras están colocadas en varias líneas. Análogamente ocurre con las configuraciones en las que las letras que indican los estados se anotan debajo de las letras del alfabeto exterior (debajo de la cinta.)

2. La máquina universal (como cualquier máquina de Turing) sólo puede tener un alfabeto exterior finito fijado. Al mismo tiempo ella tiene que estar adaptada a la posibilidad de ingreso, en calidad de información inicial, de cualesquiera esquemas y configuraciones en las cuales se pueden encontrar letras de diferentes alfabetos con un número lo que se quiera grande de variadas letras.

Por lo dicho, en primer lugar tenemos que preocuparnos de elaborar un procedimiento adecuado de presentación de los esquemas funcionales y de las configuraciones que corresponde a las particularidades indicadas de cualquier máquina de Turing cogida de por sí que son precisamente la *forma unidimensional* de la información y la *calidad finita* del alfabeto. Ahora pasaremos a la descripción de tal procedimiento.

1. En lugar de representar el esquema en forma de una tabla bidimensional que cuenta con k líneas y m columnas, anotaremos uno detrás de otro mk grupos de cinco letras de esta tabla. En cada uno de estos grupos el primer símbolo indica la columna de la tabla; el segundo, la línea de la tabla; los tres siguientes son los símbolos de las tres letras que están en la tabla en el cruce de la línea y la columna indicadas.

Por ejemplo, en lugar del esquema de la fig. 6 aparece una línea unidimensional de símbolos

$$q_1 \mid \alpha M q_2 q_2 \mid \beta M q_1 q_3 \parallel D q_1 \ldots \qquad (\Omega)$$

Es evidente que basándose en esta línea, si se desea, se puede de manera unívoca reconstruir la tabla inicial. Al examinar las configuraciones por analogía, puede acordarse que la letra que indica el estado se anote no debajo de la letra observada, sino inmediatamente a su izquierda. En este caso la configuración IV de la fig. 18 se representará con la línea

$$|||\, q_4 \,||.$$

También es evidente que con tal representación unidimensional de la configuración se puede, si es necesario, de manera unívoca restablecer su aspecto inicial.

2. Para la característica del esquema funcional y de las configuraciones no tiene importancia decisiva la imagen específica del trazado de las letras del alfabeto exterior y del alfabeto de los estados que en ellos figuran. Por ejemplo, si en toda la tabla de la fig. 6 o en la línea que le corresponde se sustituye la letra β por la letra b, pues eso no traerá ningún cambio en nuestros exámenes. Lo importante sólo es que diferentes objetos se presenten con diferentes símbolos y que se puedan diferenciar las letras de los estados de las letras del alfabeto exterior.

Está claro que se hubiesen podido elegir para la designación de los desplazamientos otras letras que no fuesen I, D, M (a la izquierda, a la derecha, no hay movimiento), pero lo que tiene que estar dicho absolutamente claro es con qué letra precisamente se designa cada desplazamiento. Aquí se pone en manifiesto el hecho de que cada una de las tres letras designa una acción completamente determinada que no se puede cambiar por otra.

Teniendo en cuenta esta circunstancia, sustituiremos en la línea Ω cada letra por separado por cierta sucesión de unidades y ceros (*por un grupo de código*) de tal manera que diferentes letras se sustituyan por diferentes grupos de código pero que una misma letra se sustituya en todos los sitios donde se encuentre siempre con el mismo grupo de código. Como resultado de tal sustitución la línea Ω, por ejemplo, tomará la forma de cierta línea Ω'. Para que por Ω' se pueda reconstruir Ω, el procedimiento de codificación (la asignación de los grupos de código a las letras) debe satisfacer las siguientes condiciones:

1) que se pueda dividir la línea Ω' de manera unívoca en cada uno de los grupos de código;

2) que se pueda distinguir qué grupos de código se han asignado a cada una de las letras I, D, M por separado y que se puedan distinguir los grupos de código asignados a las letras que denotan estado de los asignados a las letras del alfabeto exterior.

Estas dos condiciones se cumplirán sin duda alguna con el siguiente procedimiento de codificación.

1. En calidad de grupos de códigos se toman $3 + k + m$ diferentes palabras de la forma

100...01

(entre las unidades hay sólo ceros).

Entonces la división de la línea Ω' en grupos de código se hará en forma unívoca y fácilmente, seleccionando las sucesiones de ceros que se encuentran entre dos unidades.

2. La comparación de los grupos de código y las letras iniciales se hace de acuerdo con la siguiente tabla de codificación:

	Letra	*Grupo de código*		
	I	101		
	M	1001		
	D	10001		
Alfabeto exterior	S_1	100001	4 ceros	Número par de ceros que sea mayor que 2
	S_2	10000001	6 ceros	
	.....	.....	.....	
	S_k	10·····01	$2(k+1)$ ceros	
Alfabeto de los estados	q_1	1000001	5 ceros	Número impar de ceros que sea mayor que 5
	q_2	100000001	7 ceros	
	.....	.....	.....	
	q_m	10·····01	$2(m+1)+1$ ceros	

Con tal procedimiento de codificación en nuestro caso la línea Ω' se verá así:

| 00000 || 0000 || 000000 || 00 || 0000000 || 0000000 ||0000
|| 00000000 || 00 || 00000 | . . .

Semejante línea de unidades y ceros compuesta para el esquema funcional o para una configuración, la denominaremos *código del esquema funcional* y *código de la configuración*, respectivamente. Por un código se restauran fácilmente

el propio esquema o la configuración a su aspecto inicial; por eso la presentación de un esquema o de una configuración siempre se puede hacer por medio de sus códigos. Ni que decir tiene que en lugar de unidades y ceros se podían haber cogido cualesquiera otros dos signos, por ejemplo, *a* y *b*.

Ahora ya no es difícil darse cuenta de cómo cambiar el enunciado de las indicaciones 1—9 de la descripción primaria del algoritmo de imitación para obtener un algoritmo que trate y transforme los códigos del esquema de la máquina imitada y de la configuración inicial al código de la configuración resultativa. Nos limitaremos solamente a dar algunas ilustraciones.

Indicación 1. Observa en el código de la configuración el grupo de código (el único) que está inmediatamente a la derecha del grupo de código con un número impar de ceros.

Indicación 2 y 3. Encuentra en el código del esquema un par de grupos de código vecinos, iguales al par de grupos de código en la configuración en el que el segundo grupo es el observado.

Indicación 6. Si en el conjunto de los tres grupos de código observados del código del esquema, el segundo es el grupo 1001, entonces en el código de la configuración sustituye el grupo de código con un número impar de ceros por el tercer grupo de código de los tres observados.

La continuación del estudio de este algoritmo permite reducir cada operación con los grupos de código a una cadena de operaciones estándar realizables en una máquina de Turing (sustitución de un signo por otro, desplazamiento de un paso, etc.). Aquí, además de los signos 1 y 0 participarán otras letras, por ejemplo, la letra que divide un código de otro, las letras que desempeñan el papel de marcas temporales al examinar las unidades y los ceros (compárelo con el algoritmo de Euclides) y otras.

Al detallar de tal manera, el algoritmo de imitación a fin de cuentas resulta la descripción de cierto esquema funcional de Turing. Este precisamente es el esquema de la máquina universal. Si alguna máquina A resuelve cierto problema, pues la máquina universal también es capaz de resolver este problema a condición de que además del código de los datos iniciales del problema se introduzca en su cinta el esquema de la máquina A.

Teniendo en cuenta la existencia de la máquina universal de Turing se pueden interpretar cualesquiera que sean los esquemas funcionales (o sus códigos) de dos maneras:

1) el esquema define la unidad lógica de una máquina *especial* de Turing que realiza el algoritmo correspondiente (éste es el punto de vista que aplicábamos al principio);

2) el esquema define un programa que se introduce en la cinta de la máquina universal para realizar el algoritmo correspondiente.

Al concluir remarquemos que las máquinas computadoras electrónicas se construyen precisamente como máquinas universales en cuyas unidades de memoria se introducen a la par de los datos iniciales del problema planteado también el programa de su resolución.

La división de la memoria en externa e interna es característica también para las máquinas computadoras. La memoria exterior la forman con suma frecuencia tambores, cintas y discos magnéticos en los que se graba la información de modo semejante a la fonografía magnética corriente. Empero, a diferencia de la máquina de Turing en la que la memoria exterior es infinita (la cinta es infinita), en cualquier máquina computadora real la memoria exterior (la cinta, el tambor o los discos magnéticos) es finita.

Es evidente que no se puede eliminar esta diferencia de principio entre una máquina computadora real y la máquina de Turing, la que representa cierta. máquina abstracta e idealizada. Al mismo tiempo es importante hacer notar que en una máquina computadora real se puede aumentar sin límite su memoria exterior sin necesidad de hacer cambios en la construcción de la máquina; para ello es suficiente «pegar» al «trozo» de cinta magnética que está en la máquina otro «trozo» complementario.

§ 11. PROBLEMAS ALGORITMICAMENTE INSOLUBLES

El paso del concepto vago de algoritmo al concepto exacto de máquina de Turing que puede estar representada con su código, permite precisar también la cuestión sobre la solubilidad algorítmica (o de máquina) de una u otra clase de problemas. Precisamente ahora esta cuestión debe comprenderse así: ¿existe una máquina de Turing que resuel-

ve la clase de problemas dada? (lo que quiere decir «una máquina de Turing resuelve cierta clase de problemas», véase el § 7).

La teoría de los algoritmos a esta pregunta da en una serie de casos contestación negativa. Uno de los primeros resultados de este tipo establecido por el matemático norteamericano Church en el año 1936, se refiere también al problema de la distinción de posibilidad de deducción en la lógica matemática (véase el § 6).

Teorema de Church. *El problema de la distinción de la posibilidad de deducción es algorítmicamente insoluble.*

De esta manera no sólo se aclara el motivo del fracaso de todos los intentos anteriores de creación de semejantes algoritmos, sino también se descubre que esos intentos son completamente absurdos.

A las demostraciones de imposibilidad que se hacen en la teoría de los algoritmos les es propia la rigurosidad matemática característica para las demostraciones de imposibilidad a las que se llegan en otros terrenos de las matemáticas (por ejemplo, la imposibilidad de la trisección de un ángulo con la ayuda de un compás y una regla, o la imposibilidad de determinar una medida común para el lado del cuadrado y su diagonal). Ahora expondremos el esbozo de una tal demostración para el *problema de la distinción de la posibilidad de autoutilización.*

Supongamos que en la cinta de una máquina de Turing está representada su propia anotación cifrada (o sea, el código del esquema funcional de la máquina) escrita en el alfabeto de la máquina. Son posibles dos casos: 1) que la máquina es utilizable para su código, es decir, que ella trabaja este código y después de cierto número finito de tiempos se interrumpe y da la señal de parada; 2) que la máquina no sea utilizable para su código, o sea, la señal de parada no aparecerá nunca. En relación a esto las mismas máquinas (los códigos) se dividen en dos clases: la clase de máquinas de Turing (los códigos) *autoutilizables* y la clase de las *no autoutilizables.* Surge el siguiente frecuente problema.

Problema de la distinción de la posibilidad de autoutilización. *A base de cualquier código dado establecer a qué clase pertenece la máquina cifrada por ella: ¿es de la clase de las autoutilizables o de las no autoutilizables?*

Este es un problema típico de construcción de algoritmo, pues para su resolución hay que encontrar un método general (un *algoritmo* o una *máquina*) que permita para cualquier código dado determinar si es autoutilizable o no.

Teorema. *El problema de la distinción de la posibilidad de autoutilización es algorítmicamente insoluble.*

Demostración. Partamos de lo inverso, de que una tal máquina A exista. Entonces en A cualquier código autoutilizable se transforma en cierto símbolo σ (designa una contestación afirmativa a la pregunta dada sobre la autoutilización), y cualquier código no autoutilizable, en otro símbolo τ (designa una contestación negativa a la pregunta dada). En este caso se podría también crear una máquina B que como antes transformase un código no autoutilizable en τ, pero que al mismo tiempo no fuese utilizable para los códigos autoutilizables. Eso se podría conseguir por medio de tales cambios del esquema de la máquina A que después de la aparición del símbolo σ en lugar de la señal de parada la máquina se pusiese a repetir continuamente este símbolo.

Así que B es utilizable para cualquier código no autoutilizable (se elabora en este caso el símbolo τ) y no es utilizable para los códigos autoutilizables. Empero, esto lleva a una contradicción. En efecto, veamos: 1) supongamos que la máquina B es autoutilizable, entonces ella es utilizable para su código B y los transforma en el símbolo τ; pero es que la aparición de ese símbolo debe significar precisamente que B es no autoutilizable; 2) supongamos que B es no autoutilizable, entonces ella no es utilizable para B, lo que debe significar precisamente que B (B') es autoutilizable. La contradicción obtenida demuestra el teorema.

Los primeros resultados sobre la insolubilidad algorítmica fueron establecidos para problemas que surgen en la misma lógica matemática (el problema de la deductividad) y en la teoría de los algoritmos (por ejemplo, el problema de la autoutilización). No obstante, más tarde se aclaró que semejantes fenómenos también tienen lugar en ciertos problemas que parecen menos generales de las más variadas partes especiales de las matemáticas.

En primer lugar debe indicarse aquí a una serie de problemas algebraicos que llevan a diferentes variantes del

problema de las palabras que fueron investigadas por matemáticos soviéticos.

El problema de la equivalencia de las palabras para cálculos asociativos (véase el § 3) fue definido ya en el año 1914 por el matemático noruego Thue; él mismo propuso un algoritmo para la distinción de la equivalencia de las palabras en ciertos cálculos asociativos especiales. Desde entonces se emprendieron muchos intentos de crear un algoritmo general tal que permitiese para *cualquier* cálculo asociativo y para cualquier par de palabras en él establecer si esas palabras son equivalentes o no. En los años 1946 y 1947 el matemático soviético Andréi Andréievich Márkov y el matemático norteamericano Emilio Post compusieron independientemente el uno del otro *ejemplos concretos* de cálculos asociativos, en cada uno de los cuales el problema de la equivalencia de las palabras era algorítmicamente insoluble. Tanto más no existe un algoritmo para la distinción de la equivalencia de las palabras en cualquier cálculo. Posteriormente, basándose en este resultado, A. A. Márkov y sus discípulos establecieron la imposibilidad de existencia de algoritmos de distinción para una amplia clase de propiedades de los cálculos asociativos.

En el mundo de las matemáticas causó una gran impresión el resultado de Piotr Sergéievich Nóvikov sobre la insolubilidad algorítmica del problema de identidad de la teoría de los grupos que fue publicado en el año 1955*). Formalmente este problema representa un caso particular del problema de equivalencia de palabras en el cálculo asociativo**). Precisamente se examinan sólo tales cálculos asociativos en los que para cada letra a del alfabeto en la lista de sustituciones de cálculo admisibles haya una sustitución del tipo

$$a\alpha \to \Lambda,$$

*) Por este trabajo a P. S. Nóvikov le fue concendido en el año 1957 el premio Lenin.

**) Esto quiere decir que si existiese un algoritmo para aclarar la identidad de las palabras en el cálculo asociativo, este mismo algoritmo establecería la identidad de las palabras en un grupo. Empero, de la insolubilidad algorítmica del problema de la identidad de las palabras en el cálculo asociativo de ningún modo se deduce la insolubilidad algorítmica del problema correspondiente en la teoría de los grupos. (*Nota de la editorial.*)

donde α es cualquier letra del mismo alfabeto que puede coincidir con a.

El sentido que contiene esta exigencia se aclara al interpretar, por analogía con el ejemplo 4 del § 3, las palabras en cualquier cálculo asociativo como ciertas transformaciones complejas obtenidas por medio de la *multiplicación* de transformaciones elementales dadas con las letras correspondientes que forman esta palabra. En este caso la palabra vacía Λ da una transformación idéntica que no cambia nada (compárese con el § 3); la existencia de sustituciones admisibles del tipo $a\alpha \to \Lambda$ significa que para cada transformación elemental (dada con la letra a) existe una transformación elemental (dada con la letra α) tal que sus aplicaciones sucesivas dan una transformación idéntica. Sin profundizar en detalles, observaremos solamente que el examen de tales conjuntos de transformaciones, llamados grupos de transformaciones, es de un interés teórico y práctico exclusivamente grande y el propio concepto de grupo es una de las nociones básicas de las matemáticas contemporáneas.

Ahora tenemos que aclararnos que el importantísimo resultado de Márkov — Post, citado anteriormente, de por sí no permite hacer ninguna conclusión sobre lo esencial del problema de identidad de la teoría de los grupos. Es que los cálculos asociativos individuales para los cuales A. A. Márkov y E. Post establecieron la insolubilidad algorítmica del problema de la equivalencia, precisamente no satisfacen la exigencia citada antes que es esencial en el planteamiento del problema de identidad de la teoría de los grupos; por eso, la posibilidad del algoritmo para este último problema no se excluye con los resultados de Márkov — Post. La esperanza de elaborar este algoritmo todavía no estaba perdida por completo y su búsqueda todavía continuaba cuando se conoció el resultado de P. S. Nóvikov del que se deduce que tal algoritmo no existe. P. S. Nóvikov compuso un ejemplo individual de cálculo asociativo que satisface la exigencia indicada, para el cual es imposible crear un algoritmo de distinción de la equivalencia; con más razón es imposible crear un algoritmo único para todos los grupos examinados.

Los ejemplos, compuestos por A. A. Márkov y P. S. Nóvikov para refutar la solubilidad algorítmica de los problemas investigados, resultaron demasiado grandes y contaban

con cientos de sustituciones admisibles. Se planteó el problema de componer semejantes ejemplos que fuesen lo más sencillo posible. Esto fue hace poco resuelto brillantemente por el joven matemático de Leningrado G. S. Tseitin; para el cálculo expuesto en el § 3 que compuso Tseitin y que cuenta sólo siete sustituciones admisibles, el problema de la equivalencia de las palabras es también algorítmicamente insoluble.

El descubrimiento de problemas algorítmicamente insolubles ha creado en la ciencia una situación tal en la cual el matemático que aspira a crear el algoritmo deseado tiene que tener en cuenta que este algoritmo puede no existir. Por eso, simultáneamente a los esfuerzos dirigidos a la búsqueda del algoritmo deseado, también se tiene que dedicar esfuerzos a la demostración de la imposibilidad de que exista un algoritmo tal. La conclusión definitiva se aclarará en dependencia de dónde, en cuál de estas dos direcciones, se consiga el éxito; bien se encontrará un algoritmo resolutivo, bien se establecerá la insolubilidad algorítmica del problema.

En el § 1 enunciamos el problema de Hilbert sobre las ecuaciones de Diofante. Durante medio siglo en vano se hicieron investigaciones unilaterales con el fin de componer el algoritmo deseado. Como ya se dijo en el § 1 este problema también es algorítmicamente insoluble (el teorema de Yu. V. Matiashévich.)

OBSERVACIONES FINALES

En conclusión haremos algunas observaciones generales.

1. Lo primero, los teoremas sobre la insolubilidad algorítmica de una u otra clase de problemas no dan pie para caer en el agnosticismo. Efectivamente, cada tal teoría se refiere a una clase entera de problemas y establece la insolubilidad de todos los problemas de esta clase con un método eficaz único que es el algoritmo.

Eso de ninguna manera significa que entre cada uno de los problemas reunidos en esta clase hay tales que son insolubles. Por ejemplo, no se debe entender que el teorema demostrado anteriormente dice que existe tal código para el cual en principio es imposible establecer si es autoutilizable o no.

Esto sólo significa que el tipo de problemas examinado es tan amplio y general que no existe un algoritmo único para la resolución de todos los problemas de este tipo. En este caso el objetivo de las investigaciones matemáticas en la elaboración consecutiva de algoritmos cada vez más generales y que permitan reducir a un cálculo automático cada vez más amplias subclases de problemas del tipo dado.

2. Lo segundo, los teoremas sobre la insolubilidad algorítmica muestran que las matemáticas no se reducen a la creación de algoritmos, que el proceso de cognición en las matemáticas no puede ser automatizado por completo. Ya en algunos terrenos de las matemáticas relativamente limitados (como la teoría de los grupos con un número finito de componentes, etc.) ampliamente surgen problemas para los que no hay autómata capaz de resolverlos (o sea, ninguna máquina de Turing con un número finito de estados y con una memoria finita). Tanto más son absurdas las observaciones de que las máquinas podrían sustituir por completo el trabajo creador de los científicos.

3. Al mismo tiempo hay que confirmar que el campo del empleo de los procesos algorítmicos es muy extenso y no solamente incluye los procesos de cálculo puro que se utilizan en las matemáticas. Es más, en teoría se pueden crear algoritmos para muchos procesos que corrientemente se consideran muy difíciles y complejos. Estos algoritmos en su idea son sificientemente sencillos. Las dificultades prácticas que se encuentran al realizar estos procesos están relacionadas con que los algoritmos indicados son muy grandes

y exigen cumplir un grandísimo número de operaciones (aunque estas operaciones de por sí son sencillas). Esta observación se refiere, en particular, a los procesos de juego (y, en particular, al juego de ajedrez) donde el éxito depende en gran parte de la capacidad de contemplar un *gran* número de variantes para elegir la óptima.

Al crear las veloces máquinas computadoras hemos aumentado considerablemente el número de algoritmos que prácticamente se han hecho realizables.

4. Y por fin, fijemos otra vez la atención en que cada máquina computadora prácticamente realizable puede ser solamente considerada como cierto modelo aproximado de la máquina de Turing. Es decir, en las máquinas reales el volumen de la memoria exterior está limitado mientras que en el esquema de la máquina de Turing figura una cinta infinita. Está claro que es imposible la realización técnica de una memoria de capacidad infinita pero no solamente es deseable sino que absolutamente posible un aumento considerable del volumen de la memoria de las máquinas en comparación con el nivel que ya se ha conseguido. Precisamente por el camino del incremento del volumen de la memoria exterior y de la velocidad de cálculo se pueden esperar en lo sucesivo grandes éxitos en el desarrollo de las ordenadoras electrónicas.

A NUESTROS LECTORES:

«Mir» edita libros soviéticos traducidos al español, inglés, francés, árabe y otros idiomas extranjeros. Entre ellos figuran las mejores obras de las distintas ramas de la ciencia y la técnica: manuales para los centros de enseñanza superior y escuelas tecnológicas; literatura sobre ciencias naturales y médicas. También se incluyen monografías, libros de divulgación científica y ciencia-ficción.

Dirijan sus opiniones a la Editorial «Mir», 1 Rizhski per., 2, 129820, Moscú, I-110, GSP, URSS.

Este año Mir publica el libro

DE G. BERMAN

"PROBLEMAS DE ANÁLISIS MATEMÁTICO.,,

Esta colección de problemas dedicada a los estudiantes que cursan análisis matemático en centros de enseñanza técnica superior.

No contiene la teoría ni las fórmulas necesarias, el lector las encontrará en los capítulos respectivos del manual de análisis matemático. La mayoría de los problemas está subdividida en grupos, lo que facilita el trabajo con este libro. A los grupos de problemas de contenido homogéneo precede una indicación general. Todos los problemas de contenido físico van acompañados de las nociones necesarias sobre la física.

El libro contiene un apéndice: Tablas de las magnitudes de algunas funciones elementales.

El autor analiza un total de 4465 problemas. Estos «Problemas» están destinados a los estudiantes de centros de enseñanza técnica superior.

Lecciones populares de matemáticas

Este año se publicarán las siguientes obras de nuestro sello "Lecciones populares de matemáticas":

1. Bársov A.
"Qué es programación lineal"
2. Beskin N.
"Representación de figuras espaciales"
3. Boltianski V.
"La envolvente"
4. Markushévich A.
"Curvas maravillosas".
"Numeros complejos y representaciones conformes".
"Funciones maravillosas"
5. Natansón I.
"Problemas elementales de máximo y mínimo",
"Suma de cantidades infinitamente pequeñas"
6. Rozenfeld B., Sergëieva N.
"Proyección estereográfica"
7. Véntsel E.
"Elementos de la teoria de los juegos"
8. Yaglom I.
"Algebra extraordinaria"

Editorial MIR **Moscú**