ZASTOSOWANIA MATEMATYKI
V (1961)

S. PASZKOWSKI (Warszawa) i R. WRONA (Wrocław)

SCHEMATY BLOKOWE PROGRAMÓW

1. Wstęp. W ostatnich latach liczba automatycznych maszyn cyfrowych stale rośnie i powstają coraz nowe ich typy. Ta ilość i różnorodność powoduje pewne kłopoty. Aby wyjaśnić, na czym one polegają, przypomnimy kilka najbardziej istotnych faktów dotyczących obliczeń na maszynach cyfrowych.

Każda maszyna ma *pamięć*, tj. urządzenie, które może być zbudowane na wiele sposobów, wykorzystujących różne zjawiska fizyczne, lecz które zawsze służy do przechowywania przez dostatecznie długi okres czasu liczb i symboli. Chcąc rozwiązać jakieś zagadnienie na maszynie cyfrowej, należy umieścić w jej pamięci wszystkie potrzebne dane liczbowe oraz *program obliczeń*. Ten ostatni opisuje w postaci, która może być bezpośrednio wykorzystana przez maszynę, metodę rozwiązania danego zagadnienia. Program rozpada się na najbardziej elementarne czynności, tzw. *rozkazy*, wykonywane przez poszczególne urządzenia maszyny. Takimi czynnościami są np. operacje arytmetyczne, przenoszenie liczb z jednego miejsca pamięci na inne miejsce, badanie znaku liczb, wybieranie właściwego wariantu obliczeń itd. W zasadzie każdy rozkaz składa się z dwóch części; jedna określa rodzaj czynności, druga jej przedmiot. Druga część zawiera więc adresy, tj. numery tych segmentów (komórek) pamięci, gdzie mieszczą się argumenty czynności lub gdzie powinien się znaleźć wynik wykonania czynności. Liczba adresów jest jedną z cech, które wymienia się zawsze przy opisie maszyny cyfrowej. W maszynie trójadresowej (np. radzieckiej maszynie „Strieła-4", opisanej w Zastosowaniach Matematyki 5 (1960), str. 67-96) każdy rozkaz zawiera adresy trzech liczb lub symboli. Zwykle są to adresy dwóch argumentów operacji — arytmetycznej, logicznej lub innej — i adres wyniku operacji. W maszynie jednoadresowej (taką jest polska XYZ) podobna operacja jest podzielona na kilka rozkazów i każdy z nich zawiera jeden adres: albo argumentu, albo wyniku operacji.

Łatwo zrozumieć, że ograniczony zespół czynności, które maszyna może wykonywać, i ich postać (określona m. in. przez liczbę adresów

właściwą danej maszynie) wpływają w sposób decydujący na postać programu. Często na pierwszy rzut oka trudno doszukać się podobieństwa między programami tych samych obliczeń, przeznaczonymi dla różnych maszyn. Jest to oczywiście bardzo kłopotliwe, bo ośrodkom numerycznym wyposażonym w różne maszyny niezmiernie utrudnia współpracę w dziedzinie programowania. Trzeba wspomnieć i o tym, że postać programu jest zupełnie odmienna od sformułowania zagadnienia, dla którego go ułożono; że program składa się często z setek lub nawet tysięcy rozkazów i trudno go skontrolować jako całość; że nawet osobie, która program układała, trudno wyjaśnić sens jego poszczególnych fragmentów bez uprzedniego przejrzenia całego programu.

Wymienione trudności skłoniły matematyków do opracowania uproszczonej formy programu, uwzględniającej tylko najbardziej zasadnicze cechy obliczeń na maszynach cyfrowych — formy, która dzięki użyciu symboli przyjętych w rozważaniach matematycznych byłaby powszechnie zrozumiała i łatwiej poddawała się kontroli. Najczęściej używaną, uproszczoną formą programu jest jego *schemat blokowy* (inaczej zwany *siecią działań*), który w postaci graficznej przedstawia sposób rozwiązania zagadnienia, do którego go zbudowano. Schemat taki konstruuje się po ustaleniu zasadniczych założeń, dotyczących sposobu rozwiązania danego zagadnienia (a więc np. po wyborze najlepszej metody numerycznej, ustaleniu żądanej dokładności itd.). Schemat stanowi z kolei podstawę dla napisania programu, złożonego z rozkazów tej maszyny cyfrowej, na której program będzie wykonywany.

Rozwiązanie jakiegoś zadania na maszynie cyfrowej jest wykonalne, jeśli to rozwiązanie przebiega według reguł określonych z góry, które można sformułować w programie. Takie rozwiązanie będziemy krótko nazywać *obliczeniami*, pamiętając jednak o tym, że ich treść może w istocie nie mieć nic wspólnego z zagadnieniami numerycznymi i że to tylko sposób działania istniejących maszyn cyfrowych zmusza nas do liczbowej interpretacji każdego rozwiązywanego zadania. W § 2 opiszemy rodzaje czynności, z których składają się obliczenia. Elementy schematów blokowych, odpowiadające tym czynnościom, określimy w § 3. Na tym kończy się część pracy, opisująca budowę schematów blokowych. W § 4 powracamy do tematu § 2, lecz już w innym celu. Chcemy tam możliwie ściśle i ogólnie określić klasę czynności, które może wykonywać maszyna cyfrowa. Mamy tu na myśli czynności stosunkowo złożone — działania arytmetyczne, ich ciągi, obliczanie wartości funkcji itd. — w postaci, narzuconej przez organizację maszyn cyfrowych. Podobną problematyką, ale w innym, znacznie bardziej abstrakcyjnym ujęciu i w innym celu, zajmuje się np. teoria funkcji obliczalnych lub teoria automatów skończonych.

2. Obliczenia i czynności. Czynności rachmistrza w czasie wykonywania obliczeń (w ścisłym sensie tego słowa) można podzielić na *wykonawcze* i *kierujące*. Pierwszy typ, to właściwy rachunek, a więc wykonywanie działań arytmetycznych, wyszukiwanie w tablicach wartości potrzebnych funkcji i interpolacja tych wartości, porządkowanie uzyskanych wyników itp. Drugi typ czynności — niemniej ważny, choć nie zawsze uświadamiamy sobie jego istnienie — to ustalanie dalszego przebiegu obliczeń na podstawie już otrzymanych rezultatów. Prostego przykładu czynności tego typu dostarcza rozwiązywanie równania kwadratowego, zależne od znaku wyróżnika tego równania.

Podobnie można podzielić czynności, wykonywane przez tłumacza jakiegoś tekstu, np. z języka angielskiego na polski. Znalezienie w słowniku angielsko-polskim akapitu, odnoszącego się do słowa „name", jest czynnością pierwszego typu. Analizę sąsiednich wyrazów lub całego zdania możemy uważać za czynność drugiego typu, bo od jej wyniku zależy wybór jednej z czynności wykonawczych, które ustalają najwłaściwszy odpowiednik polski tłumaczonego słowa, zmieniają w razie potrzeby szyk wyrazów w zdaniu itp.

Ten sam podział dotyczy gry w szachy; tam każdy gracz na przemian analizuje pozycję figur na szachownicy (czynność kierująca) po ostatnim posunięciu przeciwnika i posuwa jedną z własnych figur (czynność wykonawcza).

Podział czynności na wykonawcze i kierujące, zilustrowany tymi przykładami, nie jest zupełnie precyzyjny, nie jest też jednoznaczny (np. czynność wykonawczą można zwykle przedstawić w postaci zespołu drobniejszych czynności wykonawczych i kierujących). Pomaga on jednak zrozumieć budowę programów.

Jeśli do wykonania jakiegoś rachunku, przekładu lub do rozegrania gry użyjemy maszyny cyfrowej, to podział czynności na wymienione powyżej typy zachowa się, choć same czynności zmienią swą zewnętrzną postać. Dlatego mamy prawo zrobić następujące założenie: obliczenia są określone przez pewne dane wielkości (np. liczby, tekst oryginalny przy tłumaczeniu, wyjściowe ustawienie figur na szachownicy itp.), układ czynności $s_1, s_2, \ldots, s_m$ oraz reguły ustalające kolejność wykonywania tych czynności i chwilę zakończenia obliczeń.

O czynnościach $s_1, s_2, \ldots, s_m$ zakładamy z kolei, że należą do jednej z dwóch klas O, P. Dowolna czynność klasy O jest zespołem operacji, wykonywanych na przedmiotach (liczbach, słowach jakiegoś tekstu, pozycjach figur na szachownicy itd.) — operacji, przekształcających te przedmioty lub tworzących z nich nowe. Natomiast dowolna czynność klasy P jest zespołem operacji, wybierającym na podstawie własności

znanych przedmiotów jedną z liczb naturalnych $1, 2, \ldots, m+1$. Sens tych czynności wyjaśni się za chwilę.

Kolejność wykonywania (być może, wielokrotnego) czynności $s_1, s_2, \ldots, s_m$ określają następujące reguły:

1. *reguła początkowa*: wykonaj czynność s_1,
2. *reguły indukcyjne*: jeśli ostatnią wykonaną czynnością było s_k,
   2.1. jeśli $s_k \in O$,
      2.1.1. jeśli $k < m$, to wykonaj czynność s_{k+1},
      2.1.2. jeśli $k = m$, to obliczenia zostały skończone,
   2.2. jeśli $s_k \in P$, jeśli wynikiem czynności s_k jest liczba naturalna l,
      2.2.1. jeśli $l \leqslant m$, to wykonaj czynność s_l,
      2.2.2. jeśli $l = m+1$, to obliczenia zostały skończone.

Przy tym operacje, wchodzące w skład każdej czynności wykonuje się na odpowiednich przedmiotach w postaci, jaką one uzyskały po wykonaniu wszystkich poprzednich czynności.

Z wyżej podanych reguł wynika, że podział czynności na klasy O i P pokrywa się z wcześniejszym podziałem na czynności wykonawcze i kierujące. Istotnie, dowolna czynność klasy P kieruje przebiegiem obliczeń, ponieważ stanowiąca jej wynik liczba naturalna — zwykle ustalana po zbadaniu własności pewnych przedmiotów — określa numer czynności, którą z kolei należy wykonać, lub sygnalizuje koniec obliczeń. Natomiast, zgodnie z regułą 2.1.1, czynności klasy O (wykonawcze), mające w ciągu $s_1, s_2, \ldots, s_m$ kolejne wskaźniki, wykonuje się jedną po drugiej.

Przykład 2.1. Równania przestępne rozwiązujemy często metodą iteracji, przedstawiwszy je w postaci $x = g(x)$. Wtedy $(n+1)$-sze przybliżenie x_{n+1} pierwiastka równania $x = g(x)$ jest określone przez poprzednie przybliżenie x_n za pomocą wzoru $x_{n+1} = g(x_n)$. Kolejne przybliżenia obliczamy dotąd, dopóki nie zostanie spełniona nierówność $|x_n - x_{n+1}| < \varepsilon$, gdzie ε jest daną liczbą dodatnią, określającą przy pewnych założeniach dokładność przybliżenia pierwiastka. Te obliczenia można sprowadzić do wielokrotnego i zgodnego z regułami 1-2.2.2 wykonania opisanych niżej czynności s_1, s_2, s_3 i s_4.

I. Czynność s_1 klasy O — obliczanie nowego przybliżenia na podstawie poprzedniego.

II. Czynność s_2 klasy P — porównanie nowego i poprzedniego przybliżenia dla sprawdzenia, czy wartość bezwzględna ich różnicy jest mniejsza od ε. Jeśli tak, to obliczenia są skończone i szukany pierwiastek równania $x = g(x)$ jest równy (z żądaną dokładnością) nowemu przybliżeniu; jeśli nie, to należy wykonać czynność s_3. Czynność s_2 przyporządkowuje więc nowemu i poprzedniemu przybliżeniu albo liczbę 5

(sygnał zakończenia obliczeń, ponieważ $m = 4$) albo liczbę 3 (polecenie przejścia do czynności s_3).

III. Czynność s_3 klasy O — przygotowanie do czynności s_1. Polega ono na tym, że nowe przybliżenie podstawia się na miejsce poprzedniego, tj. nowe przybliżenie nazywa się poprzednim. Przy obliczeniach na maszynie cyfrowej nowe przybliżenie mieści się w pewnej ustalonej komórce, poprzednie przybliżenie — w innej komórce pamięci. Czynność s_3 polega wtedy na przesłaniu liczby z pierwszej komórki do drugiej. Po czynności s_3 należałoby powrócić do czynności s_1. Żeby przy tym pozostać w zgodzie z regułami podanymi na początku tego paragrafu, musimy wprowadzić dodatkową czynność s_4.

IV. Czynność s_4 klasy P — przyporządkowanie dowolnym wartościom przybliżeń pierwiastka liczby 1, co oznacza przejście do wykonania czynności s_1.

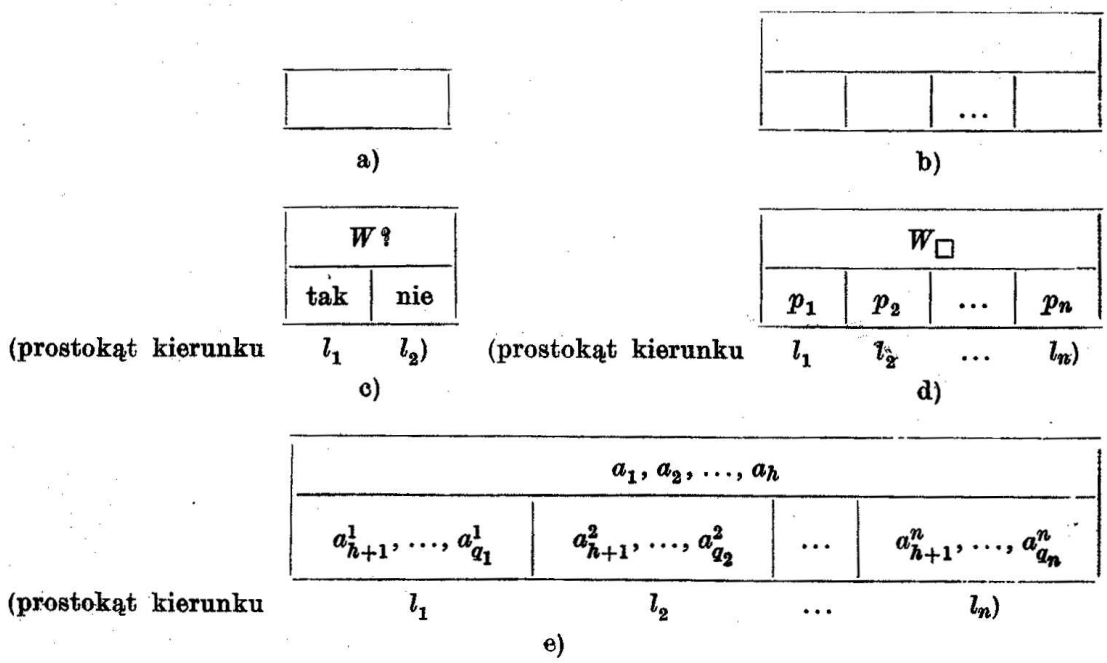
Danymi początkowymi, niezbędnymi dla przybliżonego rozwiązania równania $x = g(x)$, są pierwsze przybliżenie x_1 pierwiastka i liczba ε. Zaczynając od czynności s_1 (reguła początkowa 1) wykonujemy następnie czynność s_2 (reguła indukcyjna 2.1.1), a po niej — zależnie od jej wyniku — wykonujemy czynność s_3 (reguła 2.2.1) lub uznajemy, że pierwiastek równania został obliczony z wystarczającą dokładnością (reguła 2.2.2). Jeśli była wykonana czynność s_3, to po niej następuje czynność s_4 (2.1.1), ponownie s_1 (2.2.1), s_2 itd.

Powyższy przykład jest oczywiście prosty i błahy. Jednak i w takich, i w znacznie bardziej skomplikowanych zagadnieniach drobiazgowe ustalenie układu i porządku czynności staje się koniecznością, jeśli rachunek, przekład lub tym podobne czynności ma wykonać maszyna cyfrowa. Nie zrobi ona bowiem nic ponad to, co jej człowiek w formie programu poleci, ani nie usunie błędów z tego programu.

3. Operatory, predykaty i połączenia. Schemat blokowy jest, co zapowiedzieliśmy już w § 1, schematem graficznym, przedstawiającym, z jakich i w jakim porządku wykonywanych czynności składają się obliczenia, dla których schemat zbudowano. Elementy schematu są odpowiednikami czynności $s_1, s_2, \ldots, s_m$ i reguł wiążących te czynności (p. § 2). *Blokiem* nazywamy graficzny obraz dowolnej czynności s_k. Bloki dzielimy na operatory i predykaty. *Operator* jest graficznym obrazem czynności klasy O, a *predykat* — graficznym obrazem czynności klasy P (nazwy „operator" i „predykat" mają tu znaczenie odmienne od powszechnie przyjętego w matematyce).

Operator odpowiadający czynności klasy O przedstawiamy w postaci prostokąta (rys. 1a), wewnątrz którego umieszczamy definicję tej czynności — słowną lub składającą się z symboli teorii, do której odnoszą

się rozpatrywane obliczenia. W zagadnieniach matematycznych czynności klasy O polegają zwykle na obliczeniu wartości funkcji lub na rozwiązaniu jakiegoś równania (algebraicznego, przestępnego, różniczkowego itd.). Wewnątrz prostokąta wpisuje się wtedy właściwą funkcję lub równanie z odpowiednimi objaśnieniami.



Rys. 1

Napisy składające się na predykat odpowiadający czynności $s_k \in P$ umieszczamy w prostokącie, podzielonym linią poziomą na dwie części. Część dolna jest z kolei podzielona liniami pionowymi na tyle prostokątów, ile jest iczb naturalnych, mogących stanowić wynik czynności s_k (rys. 1b). Prostokąt części dolnej, przyporządkowany wynikowi l czynności s_k, nazywamy *prostokątem kierunku* l. Jeśli jednak wynikiem czynności s_k może być tylko jedna liczba naturalna (jak to jest dla s_4 z przykładu 2.1), to tej czynności nie przyporządkowujemy predykatu. Nie spowoduje to zmian w obliczeniach, wobec istnienia wiążących bloki połączeń, które opiszemy nieco później.

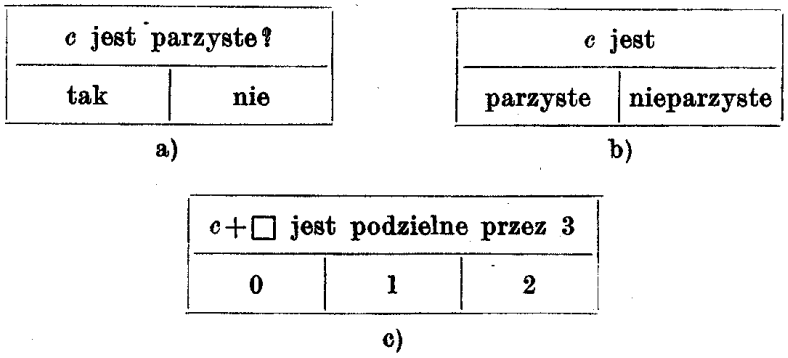
Predykaty odpowiadające pozostałym czynnościom klasy P dzielą się na trzy następujące typy:

I. Jeśli wynik czynności s_k jest jedną z dwóch liczb naturalnych l_1 i l_2, i jeśli l_1 (odpowiednio l_2) stanowi wynik s_k, gdy jest spełniona (odpowiednio — nie jest spełniona) własność W pewnych przedmiotów, to predykat odpowiadający czynności s_k ma postać, jak na rysunku 1c.

II. Jeśli 1° wynik czynności s_k jest jedną z $n \geqslant 2$ różnych liczb naturalnych $l_1, l_2, \ldots, l_n$, a l_i $(i = 1, 2, \ldots, n)$ stanowi wynik s_k wtedy i tylko

wtedy, gdy jest spełniona własność W_i pewnych przedmiotów, 2° W_i formułuje się — w symbolice teorii, do której odnoszą się dane obliczenia — w postaci ciągu symboli $a_1, a_2, \ldots, a_h, a^i_{h+1}, \ldots, a^i_{a_i}$ ($a_1, a_2, \ldots, a_h$ nie zależą od i), to predykat odpowiadający czynności s_k ma postać, jak na rysunku 1e.

III. Jeśli wynik czynności s_k jest jedną z $n \geqslant 2$ różnych liczb naturalnych $l_1, l_2, \ldots, l_n$, a l_i ($i = 1, 2, \ldots, n$) stanowi wynik s_k wtedy i tylko wtedy, gdy jest spełniona własność W_{p_i} zależna od parametru p_i, to predykat odpowiadający czynności s_k ma postać, jak na rysunku 1d.



Rys. 2

Dla wyjaśnienia, kiedy można stosować poszczególne typy predykatów, rozpatrzymy najpierw przykład obliczenia pierwiastka kwadratowego.

PRZYKŁAD 3.1. Na maszynie cyfrowej pierwiastek z liczby dodatniej x oblicza się zwykle w ten sposób, że tę liczbę sprowadza się do postaci $2^c m$, gdzie c (tzw. cecha) jest całkowite, a m (tzw. mantysa) zawiera się w przedziale $\langle \frac{1}{2}, 1)$. Następnie korzysta się ze wzoru

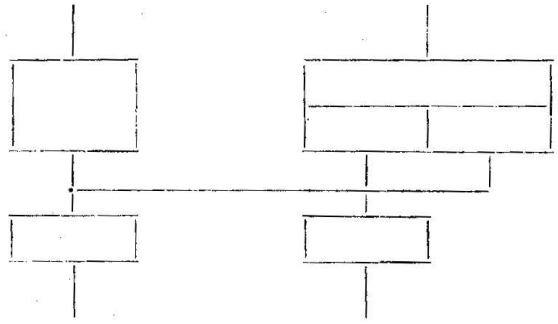
$$\sqrt{x} = \begin{cases} 2^{c/2}\sqrt{m}, & \text{jeśli } c \text{ jest parzyste,} \\ 2^{(c+1)/2}2^{-1/2}\sqrt{m}, & \text{jeśli } c \text{ jest nieparzyste.} \end{cases} \tag{1}$$

Jest to uzasadnione tym, że przybliżone obliczenie pierwiastka z liczby m, należącej do wąskiego przedziału $\langle \frac{1}{2}, 1)$, trwa znacznie krócej, niż bezpośrednie obliczenie $\sqrt{x}$, a pozostałe działania, wskazane przez wzór (1), są również łatwe do wykonania.

Przy tej metodzie obliczenia pierwiastka kwadratowego należy w odpowiednim momencie sprawdzić, czy cecha c jest parzysta, od tego bowiem zależą dalsze działania. Tej czynności — określeniu, czy liczba c jest parzysta i przyporządkowaniu jej numeru czynności, którą potem należy wykonać — odpowiada predykat typu I (rys. 2a) lub typu II (rys. 2b).

W podobny sposób, jak pierwiastek kwadratowy, oblicza się pierwiastek sześcienny, z tym tylko, że bada się podzielność cechy c przez 3. Inaczej mówiąc, należy zbadać, która z liczb $c+0$, $c+1$, $c+2$ jest podzielna przez 3. Tu jest wygodnie zastosować predykat typu III (rys. 2c).

Trzeci — po operatorach i predykatach — element schematu blokowego odpowiada podanym w § 2 regułom, a zatem wyznacza kolejność wykonywania poszczególnych czynności w każdym z możliwych warian-



Rys. 3

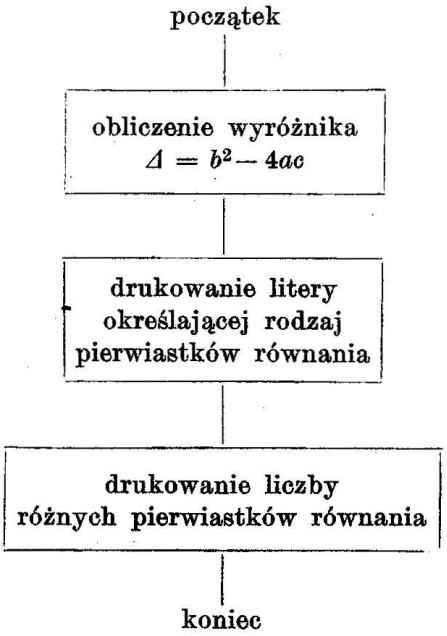
tów obliczeń. Tym elementem są połączenia między blokami oraz między blokami i pomocniczymi napisami „początek" i „koniec". Połączenia, w postaci linii ciągłych, złożonych z odcinków, należy prowadzić

1. od napisu „początek" do środka górnego boku bloku, odpowiadającego czynności s_1,
2. dla każdej czynności s_k klasy
   - 2.1. O — od środka dolnego boku operatora, odpowiadającego s_k,
     - 2.1.1. do środka górnego boku bloku s_{k+1}, jeśli $k < m$,
     - 2.1.2. do napisu „koniec", jeśli $k = m$,
   - 2.2. P — od środka dolnego boku każdego prostokąta kierunku l_{ki} $(i = 1, 2, \ldots, n_k)$, stanowiącego część predykatu, odpowiadającego czynności s_k,
     - 2.2.1. do środka górnego boku bloku $s_{l_{ki}}$, jeśli $l_{ki} \leqslant m$,
     - 2.2.2. do napisu „koniec", jeśli $l_{ki} = m+1$.

Jeśli jednak czynności s_k klasy P nie odpowiada żaden predykat (co dzieje się wtedy, gdy wynikiem tej czynności może być tylko jedna liczba), to połączenie wiąże ze sobą bezpośrednio blok lub napis „początek", poprzedzający czynność s_k z blokiem lub napisem „koniec", następującym po tej czynności. Dzięki temu właśnie nie są potrzebne predykaty, odpowiadające takim czynnościom, jak s_4 w przykładzie 2.1. W tym przykładzie połączenie, poprowadzone od bloku odpowiadającego czynności s_3 do bloku odpowiadającego s_1 (patrz rys. 6a) samo wskazuje kolejność wykonywania obu tych czynności, a więc spełnia rolę czynności s_4.

Połączenia prowadzące do napisu „koniec" lub do tego samego bloku łączymy od pewnego punktu w jedno, oznaczając ten punkt wyraźną kropką (rys. 3). Pozwala to odróżnić takie połączenia od połączeń, które z konieczności krzyżują się na rysunku.

PRZYKŁAD 3.2. Załóżmy, że dla równania kwadratowego ax^2+bx+ $+c=0$ o współczynnikach rzeczywistych należy ustalić, czy jego pierwiastki są rzeczywiste czy zespolone i ustalić liczbę różnych pierwiast-

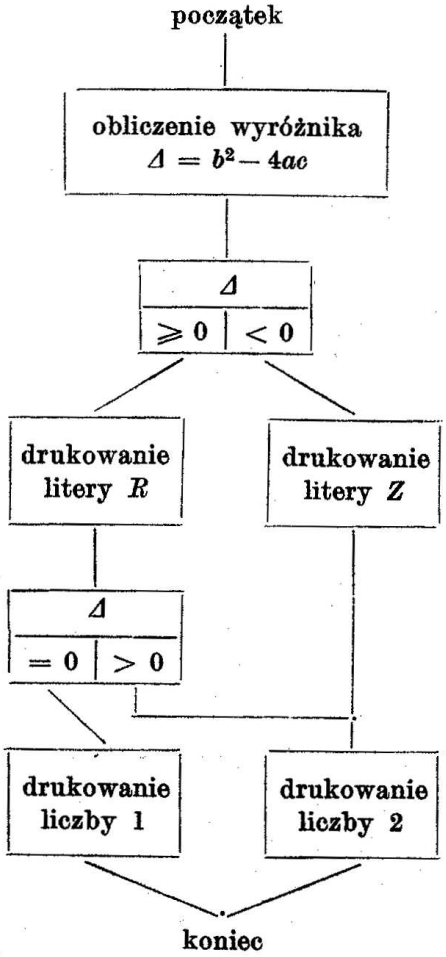


Rys. 4

ków równania. Rozwiązując to zadanie na maszynie cyfrowej możemy zażądać, by urządzenie drukujące wyniki wydrukowało pierwszą literę słowa „rzeczywiste" lub „zespolone" (określającą rodzaj pierwiastków) oraz liczbę 1 lub 2 (tj. liczbę różnych pierwiastków). Wybór jednego z tych słów i jednej z tych liczb zależy od wielkości wyróżnika $\Delta = b^2-4ac$ równania. Dlatego pierwszym operatorem w schemacie blokowym będzie obliczenie wyróżnika (używamy tu pewnego skrótu myślowego, utożsamiając blok z czynnością, do której został on zbudowany). Następny operator ustali typ pierwiastków i wydrukuje odpowiednią literę, a trzeci operator wyznaczy i wydrukuje liczbę różnych pierwiastków (rys. 4). Taki schemat blokowy można jeszcze nieco rozszerzyć przez podział niektórych jego bloków na drobniejsze elementy (rys. 5). Często tak właśnie postępujemy w miarę konkretyzowania się naszych wiadomości o sposobie rozwiązania postawionego zagadnienia.

Rozszerzony schemat z rysunku 5 czytamy w następujący sposób Pierwszą czynnością jest obliczenie wyróżnika. Następnie sprawdzamy

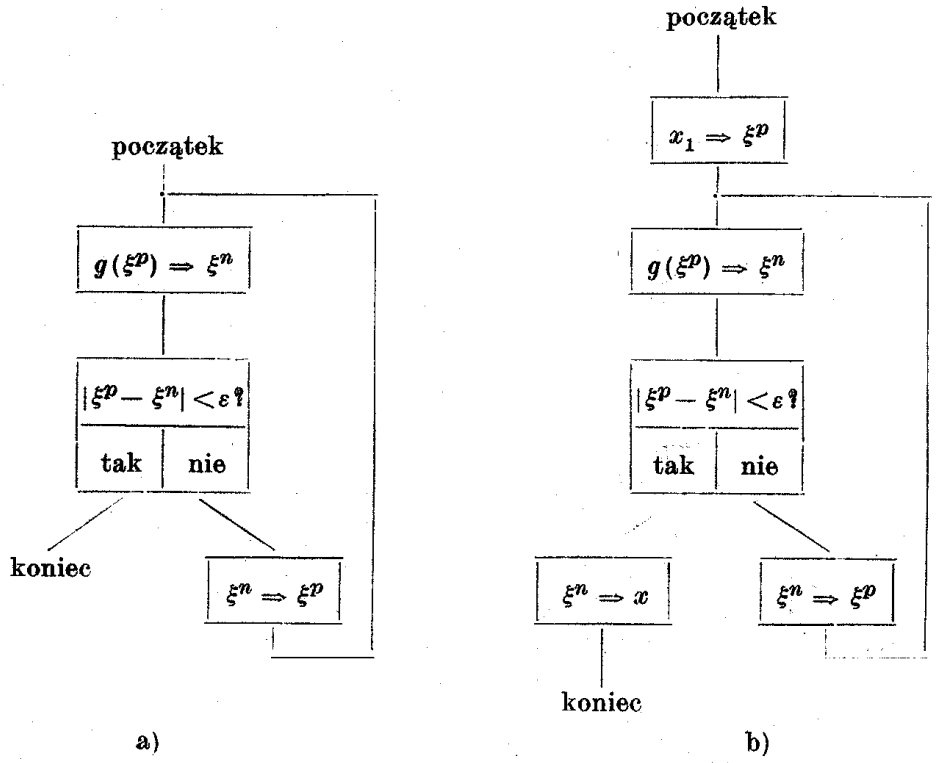
czy jest on ujemny czy nieujemny, co określa rodzaj pierwiastków równania kwadratowego. Jeśli $\Delta < 0$, to po wydrukowaniu litery Z można od razu wydrukować liczbę 2, ponieważ pierwiastki zespolone są na pewno różne. Natomiast w przypadku, gdy $\Delta \geqslant 0$, po wydrukowaniu litery R należy jeszcze sprawdzić, czy $\Delta = 0$. Jeśli $\Delta = 0$, to równanie ma jeden pierwiastek podwójny i drukuje się jedynkę. Jeśli $\Delta > 0$, to istnieją dwa różne pierwiastki rzeczywiste i należy wykonać tę samą czynność, co w przypadku $\Delta < 0$, tj. wydrukować liczbę 2.



Rys. 5

PRZYKŁAD 3.3. Powróćmy teraz do przykładu 2.1 rozwiązywania równania $x = g(x)$ metodą iteracji. Każdy krok iteracji polegał tam na obliczeniu nowego przybliżenia z pomocą poprzedniego i na porównaniu obu tych liczb. W żadnej z tych czynności wskaźniki przybliżeń nie grają istotnej roli i na miejsce symboli $x_1, x_2, \ldots, x_n, x_{n+1}, \ldots$ wprowadzimy inne, ściślej odpowiadające metodzie obliczeń. Porównajmy w tym celu dwie nazwy: „przybliżenie x_3 pierwiastka równania " i „nowe przybli-

żenie". Symbol x_3 oznacza tylko jedną liczbę (oczywiście przy założeniu, że ustalone jest przybliżenie początkowe x_1), która w danym momencie jest już znana lub jeszcze nie jest znana. Natomiast nazwie „nowe przybliżenie" odpowiadają w czasie obliczeń różne liczby — w każdej chwili wartość ostatniego z przybliżeń już obliczonych. Jest to zatem nazwa pewnej zmiennej. Oznaczmy ją symbolem ξ^n i, podobnie, zmienną o wartości równej w każdym momencie poprzedniemu przybliżeniu — symbolem ξ^p.



Rys. 6

W obliczeniach na maszynie cyfrowej pojęcie zmiennej jest bardzo naturalne. Istotnie, mówiliśmy już o tym, że w programach obliczeń nie występują liczby, lecz ich adresy. „Treść" adresu, tj. liczba zawarta w odpowiednim miejscu pamięci, może się zmieniać w czasie obliczeń, jeśli pewne rozkazy przesyłają na to miejsce wyniki wykonanych działań. Takiemu adresowi przyporządkowujemy nazwę zmiennej, której wartości mają być przechowywane pod tym adresem.

Czynność obliczenia nowej (w szczególności pierwszej) wartości zmiennej opisujemy za pomocą symbolu $\Rightarrow$. Wzór $f(\alpha, \beta, \ldots) \Rightarrow \varphi$ oznacza, że w miejsce zmiennej φ podstawia się nową jej wartość, równą wartości funkcji f obliczonej dla aktualnych wartości zmiennych $\alpha, \beta, \ldots$ Funkcja f może być stała, więc dopuszczalne są również wzory typu $0 \Rightarrow \varphi$, $1 \Rightarrow \varphi$ itp. Poza tym, aby nie komplikować oznaczeń, będziemy

wartości zmiennych oznaczać tymi samymi symbolami, co i same zmienne.

Na rysunku 6a jest przedstawiony — zgodnie z powyższymi uwagami — schemat blokowy, odpowiadający podanej w przykładzie 2.1 metodzie rozwiązania równania $x = g(x)$. Rzeczywiście, czynność $g(\xi^p) = \xi^n$ jest czynnością s_1, określoną w § 2. Po niej następuje sprawdzenie, czy jest spełniona nierówność $|\xi^p - \xi^n| < \varepsilon$ (czynność s_2). Jeśli nie, to — jak wynika ze schematu — na miejsce poprzedniego przybliżenia podstawia się nowe przybliżenie (czynność s_3), po czym powraca się do czynności s_1, co sygnalizuje na rysunku 6a odpowiednie połączenie bloków. Jeśli natomiast $|\xi^p - \xi^n| < \varepsilon$, to obliczenia są skończone.

Chcąc podkreślić, że wartością początkową zmiennej ξ^p jest przybliżenie wyjściowe x_1 i że po zakończeniu obliczeń ostatnią wartość zmiennej ξ^n przyjmujemy za pierwiastek x równania $x = g(x)$, powinniśmy przyjąć rozszerzony schemat z rysunku 6b. Jednak tym dwu dodanym operatorom nie będą, być może, odpowiadały żadne czynności wykonywane przez maszynę cyfrową. Jest to raczej tylko nadanie nowej nazwy pewnej wielkości.

4. Zmienne i programy. W § 2 pokazaliśmy, na jakie czynności można rozkładać obliczenia. Pozwoliło to zdefiniować schemat blokowy, tj. uproszczoną formę programu, związaną tylko z najbardziej ogólnymi cechami maszyn cyfrowych. Obecnie będziemy chcieli wyjaśnić, jakie czynności może wykonywać typowa maszyna cyfrowa.

Wiążąc określenie czynności z cechami działań na maszynach cyfrowych, wykorzystujemy przede wszystkim adresową budowę rozkazów i jednolitą postać symboli w pamięci maszyny. Wspominaliśmy już, że w programie liczby są reprezentowane przez ich adresy. W czasie wykonywania programu mogą ulegać zmianie nie tylko liczby, lecz także — jeśli program zawiera odpowiednie rozkazy — adresy. Załóżmy np., że trzeba zsumować grupę liczb, znajdujących się w komórkach pamięci o kolejnych adresach. Realizując tę czynność najbardziej oszczędnie pod względem wykorzystania pamięci, umieścimy w programie tylko jeden rozkaz dodawania, służący do tworzenia kolejnych sum częściowych, a w końcu i sumy wszystkich liczb danych. W czasie obliczeń rozkaz ten będzie na przemian wykonywany i przystosowywany (przez zmianę jego części adresowej) do tworzenia następnej z kolei sumy częściowej.

Adres oznacza zwykle adres jakiejś liczby, lecz w programach często mamy do czynienia również z adresami adresów, tj. adresami komórek pamięci, zawierających adresy pewnych symboli, lub z adresami adresów adresów,m ającymi jeszcze bardziej złożoną interpretację. Ten rodzaj

klasyfikacji adresów, istotny dla zrozumienia budowy programów, także uwzględnimy w poniższych definicjach.

Konsekwencją jednakowej postaci liczb i rozkazów jest możliwość zmiany tych ostatnich. W programie zatem mogą się zmieniać nie tylko liczby lub ich adresy, ale nawet ta część dowolnego rozkazu, która określa rodzaj wykonywanej przezeń czynności. Tej możliwości nie dopuszczały — przynajmniej w jakimś szerszym zakresie — definicje z § 2. Natomiast w nowych definicjach zmienna będzie pojęciem podstawowym i ogarnie wszystko z wyjątkiem reguł postępowania. Opis programów, uwzględniający powyższe uwagi, zaczniemy właśnie od wprowadzenia klasy zmiennych, których wartości są argumentami i wynikami pewnych operacji lub samymi operacjami, a także określają kolejność wykonywania tych operacji. Sama zmienna jest związana ze zbiorem wartości, które ona może przyjmować. Natomiast wartości zmiennych, przyjmowane w określonej chwili w czasie realizacji programu, są wyznaczone przez dane początkowe i czynności wykonane do tej chwili.

Definicja 1. *Zmienną porządkową* nazywamy zmienną z o wartościach naturalnych. Jak się później okaże, wartości tej zmiennej wyznaczają kolejność wykonywania operacji, zawartych w programie. W definicjach § 2 tę samą rolę odgrywały wyniki czynności klasy P.

W trzech następnych definicjach wprowadzimy rodzinę Z zmiennych $z(l, m; n)$ z parametrami całkowitymi nieujemnymi l, m i z parametrem naturalnym n. Parametry l i m określają typ zmiennej, natomiast n jest wskaźnikiem, pozwalającym na odróżnianie między sobą przy ustalonych l i m zmiennych tego samego typu. Taką funkcję spełnia również adres, określający miejsce zmiennej w pamięci. Dlatego w dalszym ciągu będziemy często utożsamiali trzeci parametr zmiennej $z(l, m; n)$ z jej adresem.

Definicja 2. Dla danych zbiorów A_n $(n = 1, 2, \ldots)$ zmienną $z(0, 0; n)$, przyjmującą wartości ze zbioru A_n, nazywamy *zmienną elementarną*. W zadaniach numerycznych zbiory A_n są zbiorami liczb, a wartości zmiennych $z(0, 0; n)$ — liczbami otrzymywanymi w czasie rozwiązywania tych zadań. W przykładzie 3.3 zmiennymi elementarnymi były ξ^p i ξ^n. W programach tłumaczenia tekstów wartościami zmiennych elementarnych mogą być słowa lub ich zespoły, w programie gry w szachy — pozycje poszczególnych figur na szachownicy itd. Jak widać, we wszystkich tych przykładach zmienne elementarne byłyby potrzebne — i to stanowi ich istotną cechę — także wtedy, gdyby dane zagadnienie rozwiązywano bez pomocy maszyn cyfrowych.

Definicja 3. Zmienną $z(l, m; n)$ dla $m > 0$, przyjmującą wartości naturalne, nazywamy *zmienną adresową m-tego rzędu*. Taką nazwę uza-

sadnia interpretacja wartości tych zmiennych, wynikająca z następnej definicji. Wartość dowolnej zmiennej adresowej może określać adres innej zmiennej, pojmowany zgodnie z uwagą, poprzedzającą definicję 2, tj. równy trzeciemu parametrowi tej innej zmiennej. Wprowadzenie pojęcia rzędu adresu odpowiada temu, że dla dodatniego m wartości zmiennych $z(l, m; n)$ będą tylko adresami zmiennych $z(l, m-1; 1)$, $z(l, m-1; 2), \ldots$ W szczególności wartości zmiennych $z(0, 1; n)$ mogą być tylko adresami zmiennych elementarnych $z(0, 0; 1)$, $z(0, 0; 2), \ldots$ Zmienne $z(0, 1; n)$ są zmiennymi adresowymi 1-go rzędu. Do ich adresowania (tj. do adresowania adresów) służą z kolei zmienne adresowe 2-go rzędu $z(0, 2; n)$. Podobnie wartości zmiennych $z(1, 1; n)$, $z(1, 2; n)$ itd. będą adresami, adresami adresów itd. zmiennych operacyjnych $z(1, 0; 1)$, $z(1, 0; 2), \ldots$ Dokładniej wyjaśnią to przykłady, które podamy po określeniu zmiennych operacyjnych.

Wartościami tych zmiennych są operacje, tj. czynności wykonywane na wartościach innych zmiennych (elementarnych, adresowych i, być może, również operacyjnych). Definicja zmiennych operacyjnych powinna — zgodnie z wcześniejszymi uwagami — uwzględnić potrzebę operacji, przekształcających inne operacje. Musi to być zatem definicja indukcyjna, w której określa się kolejno operacje na wartościach zmiennych elementarnych i ich adresach — czyli operacje 1-go rzędu — następnie operacje na poprzednio określonych operacjach 1-go rzędu i ich adresach — czyli operacje 2-go rzędu itd. To wszystko komplikuje definicję, ale choć dla większości istniejących programów wystarczyłyby operacje 2-go rzędu, to jednak trudno byłoby określić maksymalny rząd operacji, jaki możemy napotkać w bardziej skomplikowanych zagadnieniach.

Definicja 4. Zmienną $z(l, 0; n)$ dla $l > 0$ nazywamy *zmienną operacyjną l-tego rzędu*. Dowolna wartość tej zmiennej jest operacją, której argumentami są

I. wartość zmiennej z,

II. 1. liczba naturalna s,
2. liczby całkowite $k_1, k_2, \ldots, k_s$ z przedziału $\langle 0, l-1\rangle$,
3. liczby naturalne $m_1, m_2, \ldots, m_s$,
4. liczby naturalne $z_{1m_1}, z_{2m_2}, \ldots, z_{sm_s}$,
5. wartości z_{i,m_i-j} zmiennych $z(k_i, m_i-j;\ z_{i,m_i-j+1})$ dla $j = 1, 2, \ldots, m_i$ i dla $i = 1, 2, \ldots, s$,

III. 1. liczba naturalna σ,
2. liczby całkowite $\varkappa_1, \varkappa_2, \ldots, \varkappa_\sigma$ z przedziału $\langle 0, l-1\rangle$,
3. liczby naturalne $\underline{\mu}_i, \bar{\mu}_i, \mu_i$ takie, że $\underline{\mu}_i \leqslant \bar{\mu}_i \leqslant \mu_i$ $(i = 1, 2, \ldots, \sigma)$,
4. liczby naturalne $\zeta_{1\mu_1}, \zeta_{2\mu_2}, \ldots, \zeta_{\sigma\mu_\sigma}$,

5. wartości ζ_{i,μ_i-j} zmiennych $z(\varkappa_i, \mu_i-j; \zeta_{i,\mu_i-j+1})$ dla $j = 1, 2, \ldots, \mu_i-\underline{\mu}_i$ i dla $i = 1, 2, \ldots, \sigma$.

Operacja ta przyporządkowuje wymienionym argumentom nową wartość zmiennej z i nowe wartości ζ_{i,μ_i-j} zmiennych $z(\varkappa_i, \mu_i-j; \zeta_{i,\mu_i-j+1})$ dla $j = \mu_i-\underline{\mu}_i+1, \ldots, \mu_i-\underline{\mu}_i+1$ i dla $i = 1, 2, \ldots, \sigma$. Liczby $s, k_i, m_i, z_{i,m_i-j}, \sigma, \varkappa_i, \underline{\mu}_i, \underline{\mu}_i, \mu_i, \zeta_{i,\mu_i-j}$ zależą również od parametrów l i n; zależności tej nie uwzględniliśmy w oznaczeniach, by ich nie komplikować.

Wytłumaczymy teraz poszczególne elementy definicji zmiennej operacyjnej, pomijając na razie udział w tej definicji zmiennej porządkowej z, której rola wyjaśni się później.

Przykład 4.1. Rozważmy operację $g(\xi^p) \Rightarrow \xi^n$, a więc jedną z czynności, składających się na rozwiązanie równania $x = g(x)$ (patrz rys. 6a). Zmienne ξ^p i ξ^n są zmiennymi elementarnymi, ponieważ ich wartości oznaczały odpowiednio poprzednie i nowe przybliżenie pierwiastka równania. Przyjmijmy dla tych zmiennych symbole $z(0, 0; 1)$ i $z(0, 0; 2)$. Zbiory A_1 i A_2 wartości tych zmiennych można określić jednakowo, jako przedział, do którego należą wszystkie kolejne przybliżenia pierwiastka.

Operacja $g(\xi^p) \Rightarrow \xi^n$ oznacza podstawienie na miejsce zmiennej $\xi^n \equiv z(0, 0; 2)$ pewnej wartości, określonej przez wartość zmiennej $\xi^p \equiv z(0, 0; 1)$. Postać operacji wyznacza funkcja g oraz numery (adresy) 2 i 1 zmiennych elementarnych, których wartości stanowią odpowiednio argument i wynik operacji. Aby opisać tę operację w postaci, jaką definicja 4 przewiduje dla dowolnej wartości zmiennej operacyjnej, trzeba dobrać odpowiednio wymienione tam liczby. Przede wszystkim przyjmiemy $l = 1$, ponieważ rozpatrywana operacja, działając tylko na zmienne elementarne, jest operacją 1-go rzędu. Stąd od razu wynika, że liczby k_i i $\varkappa_i$ muszą być równe zeru. Liczba s określa liczbę zmiennych elementarnych, których wartości wraz z adresami są argumentami operacji; stąd $s = 1$. Liczby $m_1, m_2, \ldots, m_s$ określają maksymalne rzędy adresów poszczególnych zmiennych elementarnych lub operacyjnych, które to adresy stanowią argumenty operacji wraz z wartościami tych zmiennych. W naszym przykładzie $m_1 = 1$. Liczba $z_{1m_1} \equiv z_{11}$ wyznacza natomiast wartość adresu najwyższego rzędu. W rozważanym przykładzie mamy $z_{11} = 1$, bo argumentem operacji jest zmienna $z(0, 0; 1)$. Punkt II.5 definicji da już teraz automatycznie właściwy argument operacji. Istotnie, jest nim wartość z_{1,m_1-j} zmiennej $z(k_1, m_1-j; z_{1,m_1-j+1})$ dla $j = 1$, tj. wartość z_{10} zmiennej $z(0, 0; z_{11}) \equiv z(0, 0; 1) \equiv \xi^p$.

Następna grupa argumentów jest związana z wynikami operacji w tym sensie, że ustala ich adresy. Przy tym rola liczb $\sigma, \varkappa_i, \mu_i, \zeta_{i\mu_i}$ jest analogiczna do roli liczb s, k_i, m_i, z_{im_i}. Przyjmijmy wobec tego

$\sigma = 1$ (liczba zmiennych elementarnych lub operacyjnych, których wartości lub adresy są wynikami operacji), $\mu_1 = 1$ (maksymalny rząd znanego adresu), $\zeta_{1\mu_1} \equiv \zeta_{11} = 2$ (adres wyniku). Przyjmijmy też $\underline{\mu}_1 = \bar{\mu}_1 = 1$. Wtedy punkt III.5 definicji można pominąć, ponieważ wskaźnik j nie przebiega ani jednej wartości (ściślej mówiąc, właśnie tak chcemy rozumieć równość $\mu_i - \bar{\mu}_i = 0$). Natomiast wynikiem operacji jest — prócz nowej wartości zmiennej porządkowej — nowa wartość ζ_{1,μ_1-j} zmiennej $z(\varkappa_1, \mu_1 - j; \zeta_{1,\mu_1-j+1})$ dla $j = 1$, czyli wartość ζ_{10} zmiennej $z(0, 0; \zeta_{11}) \equiv z(0, 0; 2) \equiv \xi^n$, co jest zgodne z naszymi życzeniami.

Tym, co charakteryzuje opis operacji zgodny z definicją 4, jest podkreślenie roli adresów, zupełnie równorzędnej z rolą zmiennych o tych adresach. Potoczne stwierdzenie, że „danej wartości zmiennej ξ^p operacja przyporządkowuje wartość zmiennej ξ^n" definicja 4 tłumaczy na zdanie: „danej zmiennej ξ^p (tj. pewnemu adresowi, odróżniającemu tę zmienną od innych) i jej wartości oraz danej zmiennej ξ^n (tj. jej adresowi) operacja przyporządkowuje wartość tej ostatniej zmiennej". Takie ujęcie jest zgodne z budową rozkazu, zawierającego adresy argumentów lub wyników operacji.

Przykład 4.2. Rozważmy teraz operację nieco bardziej skomplikowaną. Załóżmy, że grupę liczb napisanych na kartce papieru mamy uporządkować według wielkości, od najmniejszej liczby do największej. Możemy to zrobić, wypisując kolejno najmniejszą z porządkowanych liczb, liczbę najmniejszą z pozostałych itd., i skreślając liczby już wykorzystane. Podobnie jest zbudowany jeden z programów porządkujących grupę liczb zawartych w pamięci maszyny cyfrowej. Skreślanie liczby wybranej, które nie ma odpowiednika wśród rozkazów maszyny, zastąpiono w programie przez wpisywanie na miejsce tej liczby największej liczby (oznaczmy ją $L_{\max}$), jaka może się znajdować w pamięci. Nie zmienia to postaci ciągu liczb po uporządkowaniu.

Podstawową operacją, powtarzającą się w programie wielokrotnie, jest wybranie najmniejszej liczby spośród liczb porządkowanych (niektóre z nich są już zastąpione przez $L_{\max}$ i te na pewno wybrane nie będą) i dołączenie tej liczby do grupy liczb, wybranych w poprzednich krokach postępowania. Liczby po uporządkowaniu mają się mieścić w określonych komórkach pamięci i możemy te liczby interpretować jako wartości pewnych zmiennych elementarnych. Wtedy wspomniana operacja podstawowa sprowadzi się do określenia miejsca w pamięci, na którym trzeba umieścić wybraną liczbę, tj. adresu jednej ze zmiennych elementarnych, i określenia wartości tej zmiennej. Ten adres jest z kolei wartością pewnej zmiennej adresowej pierwszego rzędu, np. zmiennej $z(0, 1; 3)$. Wprowadzimy jeszcze oznaczenia $z(0, 0; 1)$, $z(0, 0; 2)$, ..., $z(0, 0; p)$ dla zmiennych elementarnych, których początkowymi war-

tościami są liczby przed ich uporządkowaniem. W miarę wybierania spośród tych liczb najmniejszej, najmniejszej z pozostałych itd., wartości odpowiednich zmiennych są, jak powiedzieliśmy, zastępowane przez $L_{\max}$. Argumentami rozważanej operacji podstawowej są 1° liczby porządkowane, a ściślej wartości zmiennych $z(0, 0; i)$ dla $i = 1, 2, \ldots, p$, 2° wartość zmiennej $z(0, 1; 3)$, pozostała po poprzednim wykonaniu takiej samej operacji, a więc określająca miejsce, na którym umieszczono ostatnią liczbę, wybraną z pierwotnego układu. Wynikami operacji są 1° nowa wartość zmiennej adresowej $z(0, 1; 3)$, większa od poprzedniej o 1 i określająca adres, pod którym należy umieścić najmniejszą z aktualnych wartości zmiennych $z(0, 0; i)$, 2° nowa wartość zmiennej elementarnej o tym adresie.

Definicja 4 obejmuje i taką operację. Istotnie, przyjmijmy $l = 1$ (jak w poprzednim przykładzie, operacja działa tylko na zmiennych elementarnych i ich adresach), $s = p+1$, $k_1 = k_2 = \ldots = k_{p+1} = 0$, $m_1 = m_2 = \ldots = m_p = 1$, $m_{p+1} = 2$, $z_{11} = 1$, $z_{21} = 2, \ldots, z_{p1} = p$, $z_{p+1,2} = 3$. Z punktu II.5 definicji 4 wynika, że argumentami operacji są wartości z_{i0} zmiennych $z(0, 0; i)$ dla $i = 1, 2, \ldots, p$, wartość $z_{p+1,1}$ zmiennej $z(0, 1; 3)$ i wartość $z_{p+1,0}$ zmiennej $z(0, 0; z_{p+1,1})$. Są to rzeczywiście te argumenty, które wymieniliśmy poprzednio, tylko ostatni — wymuszony niejako przez budowę definicji 4 — jest niepotrzebny, wobec czego założymy, że wynik operacji od $z_{p+1,0}$ nie zależy.

Przyjmijmy następnie, że $\sigma = 1$, $\varkappa_1 = 0$, $\underline{\mu}_1 = 1$, $\bar{\mu}_1 = \mu_1 = 2$, $\zeta_{12} = 3$. Wtedy, jak w poprzednim przykładzie, punkt III.5 definicji można pominąć. Wynikiem operacji jest wartość ζ_{11} zmiennej adresowej $z(0, 1; 3)$ i wartość ζ_{10} zmiennej elementarnej $z(0, 0; \zeta_{11})$. Liczba naturalna ζ_{11}, równa $z_{p+1,1}+1$ (ponieważ wybierane liczby umieszczamy w komórkach pamięci o sąsiednich adresach), wyznacza tę zmienną elementarną, której wartością ma być liczba $\zeta_{10} = \min\{z_{10}, z_{20}, \ldots, z_{p0}\}$.

Przykład 4.3. Rozważaliśmy dotąd jedynie operacje pierwszego rzędu ($l = 1$), tj. operacje na zmiennych elementarnych i ich adresach. Rzadziej zdarzają się operacje na operacjach, ale i taki przykład łatwo podać.

Większość maszyn cyfrowych operuje liczbami, wyrażonymi w układzie dwójkowym, ale do pamięci maszyny wprowadza się liczby w układzie dziesiętnym, odpowiednio zaszyfrowane. Wobec tego, po wprowadzeniu liczby x należy ją przetłumaczyć na układ dwójkowy. Wystarczy oczywiście przetłumaczyć liczbę $|x|$ i wynikowi dać taki znak, jaki miała liczba x jeszcze w układzie dziesiętnym. Nadanie znaku liczbie $(|x|)_2$ (wskaźnik 2 oznacza wyrażenie liczby w układzie dwójkowym) polega na dodaniu jej do zera, jeśli $x \geqslant 0$ i odjęciu jej od zera, jeśli $x < 0$. W programie jest przewidziane za rozkazami tłumaczenia miejsce na jedną

z tych dwóch czynności i temu miejscu odpowiada pewna zmienna operacyjna 1-go rzędu. Jej wartością jest operacja „dodaj $(|x|)_2$ do zera", jeśli $x \geqslant 0$ i operacja „odejmij $(|x|)_2$ od zera", jeśli $x < 0$. Wybór jednej z tych dwóch operacji, zależny od znaku liczby x i dokonywany bezpośrednio po wprowadzeniu tego znaku do pamięci, też jest pewną operacją, ale już bardziej skomplikowaną — operacją drugiego rzędu. Opiszemy ją zgodnie z definicją 4.

Załóżmy, że liczba $\operatorname{sign} x$ jest wartością zmiennej elementarnej $z(0, 0; 2)$ i że wspomniana zmienna operacyjna 1-go rzędu jest zmienną $z(1, 0; 4)$. Wtedy przyjmując $l = 2$ (rząd definiowanej operacji), $s = 1$, $k_1 = 0$, $m_1 = 1$, $z_{11} = 2$, $\sigma = 1$, $\varkappa_1 = 1$, $\underline{\mu}_1 = \bar{\mu}_1 = \mu_1 = 1$, $\zeta_{11} = 4$ otrzymamy ścisłe sformułowanie operacji, wybierającej jedną z dwóch możliwych czynności, które powinny nadać znak liczbie $(|x|)_2$. Argumentem tej operacji (prócz wymienionych już parametrów całkowitych) jest wartość z_{10} zmiennej $z(0, 0; 2)$, tj. liczba $\operatorname{sign} x$, a wynikiem operacji jest wartość ζ_{10} zmiennej $z(0, 1; 4)$, tj. operacja nadania znaku liczbie $(|x|)_2$.

Po określeniu zmiennej porządkowej oraz zmiennych elementarnych, adresowych i operacyjnych możemy opisać strukturę programu. Wykonanie programu składa się z *taktów*, numerowanych kolejnymi liczbami naturalnymi. W każdym takcie wykonuje się jedną operację, tj. pewną wartość jakiejś zmiennej operacyjnej.

Program jest w pełni zdefiniowany przez 1° wartości początkowe zmiennej z i potrzebnych zmiennych klasy Z, 2° ciąg zmiennych operacyjnych $z_1, z_2, \ldots, z_q$ i 3° następującą regułę, wyznaczającą operację wykonywaną w k-tym takcie ($k = 1, 2, \ldots$): jeśli w jest wartością zmiennej z, ustaloną w poprzednim takcie (a dla $k = 1$, jeśli w jest wartością początkową zmiennej z), to (i) w k-tym takcie należy wykonać operację, będącą wartością zmiennej z_w po $(k-1)$-szym takcie (wartością początkową zmiennej z_w dla $k = 1$) na wartościach zmiennych, danych po tymże takcie (wartościach początkowych dla $k = 1$), jeśli $w \leqslant q$; (ii) obliczenia zostały skończone, jeśli $w > q$.

Jest to w zasadzie ta sama definicja, którą podaliśmy już w § 2. Różnica między nimi polega na tym, że w nowych określeniach połączono operacje wykonawcze i kierujące. Zgodnie z definicją 4 dowolna operacja jest w pewnym sensie kierującą, bo wyznacza numer następnej zmiennej operacyjnej z ciągu $z_1, z_2, \ldots, z_q$. Tym numerem jest wartość zmiennej porządkowej z. Takie ujęcie operacji jest najbliższe czteroadresowych maszyn cyfrowych, w których rozkaz zawiera oprócz adresów argumentów i wyniku operacji także adres następnego rozkazu. Drugą, ważniejszą różnicę między definicjami z § 2 i z § 4 omówiliśmy

już wcześniej; jest nią dopuszczenie zmienności również operacji, a nie tylko ich liczb i adresów.

INSTYTUT MATEMATYCZNY POLSKIEJ AKADEMII NAUK
INSTYTUT BADAŃ JĄDROWYCH POLSKIEJ AKADEMII NAUK
KATEDRA MATEMATYKI POLITECHNIKI WROCŁAWSKIEJ

Praca wpłynęła 6. 2. 1960

С. ПАШКОВСКИЙ (Варшава) и Р. ВРОНА (Вроцлав)

БЛОК-СХЕМЫ ПРОГРАММ

РЕЗЮМЕ

В § 1 вводятся основные понятия, связанные с вычислениями на автоматических цифровых машинах. В § 2 определены (с точностью, достаточной для вычислительной практики) типы действий, из которых состоят вычисления, и способ соединения этих действий в программу. На этом основывается описание блок-схем программ (§ 3).

В § 4 мы определяем — возможно общо и четко — класс действий, выполняемых типичной цифровой машиной. Приведенные там рассуждения опираются на введение четырех типов переменных. Значения порядковой переменной определяют порядок следования отдельных действий (операций), из которых составлена программа. Значения элементарных переменных являются основными объектами, которыми оперирует решаемая задача (числа, слова при переводе текста и т. д.). Значения адресных переменных являются адресами элементарных, операционных, а также других адресных переменных. Значения операционных переменных являются операциями. Любая операция определяет новые значения некоторых элементарных, адресных и операционных переменных, а также адрес следующей операции.

S. PASZKOWSKI (Warszawa) and R. WRONA (Wrocław)

ROUTINE FLOW-DIAGRAMS

SUMMARY

In § 1 we find a description of the basic concepts connected with calculations on automatic digital computers. In § 2 the authors define (with an accuracy sufficient for practical calculations) the types of functions making up the calculations and the method of combining those operations into a routine. This gives a basis for the description, given in § 3, of the construction of routine flow-diagrams.

§ 4 defines, as accurately and as generally as possible, the class of functions which the computer can perform. The considerations contained in it are based on the introduction of four kinds of variables. The values of the order variable determine

the successive order of performing the individual functions making up the routine. The values of the elementary variables are the basic objects appearing in the problem which is being solved (numbers, words of the text which is being translated, etc.). The values of the address variables are the addresses of the elementary variables and the operation variables. The values of the latter variables are operations, i. e. functions which can make up routines. Every operation determines new values of some of the elementary, address and operation variables and the address of the next operation.