

## 2.1. PODSIEĆ LINIOWA

Podsieć liniowa jest najprostszym typem podsieci. Składa się ona jedynie z operatorów. Wyjście pierwszego operatora jest połączone z wejściem drugiego operatora itd. Wyjście przedostatniego operatora jest połączone z wejściem ostatniego operatora (patrz rys: 3).



Rys. 3. Schemat blokowy fragmentu programu o podsieci liniowej

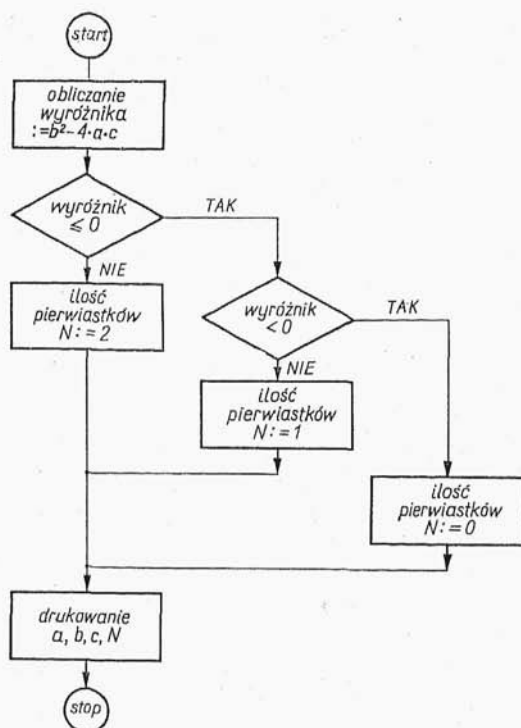
W toku pracy komputera każdy z operatorów wchodzących w skład podsieci liniowej jest wykonywany jeden raz. Teoretycznie, można wyobrazić sobie złożony program o sieci liniowej. Zaletą takiego programu byłaby maksymalna szybkość jego realizacji dla danego komputera. Podstawową jednak wadą byłyby jego rozmiary. Dlatego też dążymy do pisania programów zawierających w sobie części wielokrotnie wykonywane w toku jednej realizacji danego programu.

## 2.2. PODSIECI Z ROZWIDLENIAMI

Dla opisanía tych fragmentów programu, w których w różnych przedziałach (lub układach warunków) rozwiązania uzyskiwane są za pomocą różnych formuł (lub procedur postępowania), wykorzystujemy podsieci z rozwidleniami. Podsieć taka zawiera test (lub testy) spraw-

dzający warunki, sekwencję operatorów — zwane wariantami — realizujących poszczególne formuły i wreszcie operator (lub sekwencję operatorów) realizujący końcowe czynności.

Przez *wariant* rozumiemy tutaj fragment sieci programu, który może być wykonywany lub nie, przy jednorazowej realizacji danego programu lub jego części. Przykład elementarnej podsieci z rozwidleniami jest przedstawiony na rysunku 4.



Rys. 4. Schemat blokowy fragmentu programu o podsieci z rozgałęzieniami

## 2.3. PODSIECI Z CYKLEM

Fragmenty programów, w których zadana czynność (czy też zespół czynności) jest przetwarzana z góry określona ilość razy (np. na nowych danych każdorazowo), wykorzystują podsieci z cyklem. Pod-



Rys. 5. Schemat blokowy fragmentu programu o podsieci z cyklem

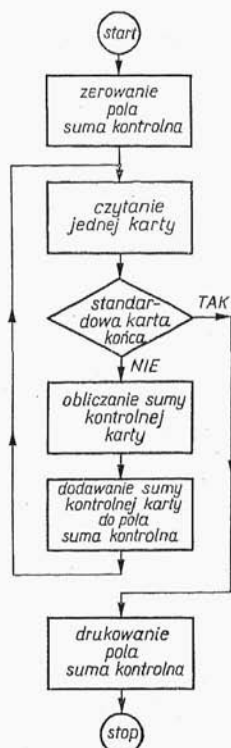
wykorzystują podsieci z cyklem. Podsieć taka składa się z operatora (lub operatorów) przygotowania cyklu (ustawienie wartości początkowych, ustawienie wartości licznika cyklu itp.), z sekwencji operatorów wykonujących powtarzalną czynność (czy też zespół czynności), z operatora zmniejszającego zawartość licznika cyklu o stałą (np. równą jeden), z testu badającego zawartość licznika cyklu  $i$  w zależności od wyniku badania przekazującego sterowanie do pierwszego operatora sekwencji wewnętrznej cyklu albo do dalszej części programu.

Przez pierwszy operator sekwencji wewnętrznej cyklu rozumiemy tutaj pierwszy operator wykonujący powtarzalny zespół czynności realizowanych przez dany cykl. Przykład elementarnej podsieci z cyklem jest przedstawiony na rysunku 5.

## 2.4. PODSIECI Z ITERACJĄ

Podsieci z iteracją są bardzo podobne do podsieci z cyklem. Podstawowa różnica między nimi sprowadza

się do tego, że w podsieciach z iteracją krotność wykonywania powtarzalnej części czynności nie jest z góry określona, ale zależy od wyników uzyskanych w toku realizacji powtarzalnego fragmentu programu. W zasadzie podsieć z iteracją w odróżnieniu od podsieci z cyklem nie ma licznika cyklu. Określenie zostało użyte, ponieważ istnieją pewne szczególne przypadki podsieci z iteracją, w których licznik występuje. Mianowicie są to przypadki, kiedy ustalona jest minimalna, czy też maksymalna krotność wykonania iteracji. Ponadto, w odróżnieniu, od podsieci z cyklem, podsieć z iteracją zawiera test badający warunki decydujące o dalszym powtarzaniu iteracji (czy też o jej przerwaniu) nie na końcu sekwencji operatorów wewnętrznych, ale w środku lub na początku. Przykład elementarnej podsieci z cyklem jest przedstawiony na rysunku 6.

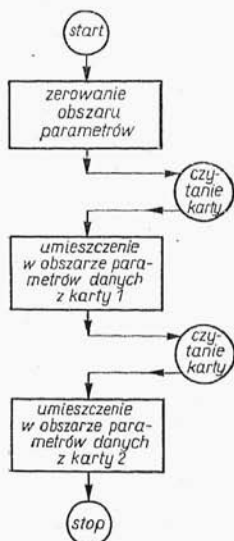


Rys. 6. Schemat blokowy fragmentu programu o podsieci z iteracją

## 2.5. PODSIECI Z PODPROGRAMAMI

Przez *podprogram* będziemy rozumieli część programu o autonomicznej podsieci, mającej jedno lub więcej wejść oraz jedno wyjście, przez które następuje przekaza-

zanie sterowania do miejsca w programie wskazanym przez tzw. ślad. Zawartość śladu jest ustalana w mo-



Rys. 7. Schemat blokowy fragmentu programu o podsieci z podprogramami

mentu przekazania sterowania do któregoś z wejść autonomicznej podsieci. Dla uproszczenia rozważań przyjmujemy dodatkowe założenie: sterowanie z wyjścia podprogramu nie może być bezpośrednio przekazane do żadnego z wejść podprogramu. W ten sposób unikniemy zmuszających rozważań definicyjnych, ale równocześnie zrezygnowaliśmy z tzw. procedur rekursywnych. Wracając do pojęcia podsieci z podprogramem, możemy ją zdefiniować jako podsieć zawierającą w sobie operatory wywoływania podprogramu (lub operatory wywoływania podprogramów). Przykład elementarnej podsieci z podprogramem jest pokazany na rysunku 7.

## 2.6. SIEĆ DZIAŁAŃ PROGRAMU

Dotychczas wyróżniliśmy pięć podstawowych typów podsieci. Każda sieć bardziej rozbudowanego programu komputerowego, w szczególności sieć programu przetwarzania danych, zawiera w sobie elementy, które możemy zaklasyfikować zawsze do jednego z pięciu przedstawionych typów podsieci. W praktyce mamy jednak do czynienia z pewnymi superpozycjami wymienionych

typów podsieci. Na przykład częstym przypadkiem są podsieci z cyklem, zawierające wewnątrz inną podsieć z cyklem lub iteracją, lub podsieć z rozwidleniem, zawierającą z kolei w którymś z wariantów podsieć z podprogramem itd.

Typowym dla programów przetwarzania danych jest bardzo duża liczba warunków, będących przedmiotem badania przez testy programu. Poszczególne kombinacje stanów warunków określają tzw. fazy programu. Często się zdarza, że w kolejnych fazach programu wykorzystuje się inny podział pamięci, operując innymi zbiorami danych (patrz rozdz. 3) lub traktując niektóre zbiory danych w sposób odmienny.

Charakterystyczne jest również wykorzystywanie pewnych grup funkcji programu równocześnie przez różne fazy programu. Każda faza programu realizuje inną sytuację decyzyjną, przy czym warunki wyznaczające fazę programu determinują w sposób jednoznaczny sytuację decyzyjną.

Historycznie pierwsza metoda opisu struktury programu za pomocą schematu blokowego sieci działania programu jest mało przydatna do opisu programów przetwarzania danych. Dla opisu części programu, w których występują grupy funkcji wykorzystywane równocześnie w niektórych fazach, przydatna jest tzw. technika tablic decyzyjnych. Technika tablic decyzyjnych została rozwinięta w drugiej połowie lat pięćdziesiątych i w latach sześćdziesiątych [1], [8]. Technika ta jest obecnie szeroko stosowana zarówno do opisu programów, jak i do projektowania systemów przetwarzania danych. Tablice decyzyjne omówimy na przykładzie opisu struktury wewnętrznej programu przetwarzania danych, przy czym nasz wykład odbiega nieco od techni-

ki opisanej w literaturze. Tablice decyzyjne nawiązują do tradycyjnej już dzisiaj formy przedstawiania różnych funkcji w formie tablic.

## 2.7. TABLICE DECYZYJNE

Sporządzimy początkowo uporządkowaną listę wszystkich operatorów występujących w programie. Uporządkowanie winno zabezpieczyć kolejność występowania operatorów w poszczególnych fazach (problem kolejności postaramy się jeszcze bliżej wyjaśnić). Wyobraźmy sobie, że umiemy wyróżnić kilka lub kilkanaście faz programu, przy czym w każdej fazie działa pewna liczba operatorów, zawsze jednak zgodnie z kolejnością umieszczenia operatorów na liście. Napiszmy z kolei matrycę złożoną z dwu rodzajów elementów, np. „—” i „X”. Przyporządkujmy każdej fazie programu jedną kolumnę naszej matrycy, zaś każdemu operatorowi — jeden wiersz matrycy w kolejności występowania danego operatora na liście. Jeśli  $i$ -ty operator działa przy wykonywaniu  $j$ -tej fazy, to jako element matrycy znajdujący się na przecięciu  $i$ -tego wiersza i  $j$ -tej kolumny położymy „X”. Jeśli  $i$ -ty operator nie działa przy wykonywaniu  $j$ -tej fazy programu, to jako element matrycy znajdującej się na przecięciu  $i$ -tego wiersza i  $j$ -tej kolumny położymy „—”.

Tak zbudowaną tablicę uzupełnimy z kolei listą testów determinujących poszczególne sytuacje decyzyjne i fazy programu (opisane jako kolumny matrycy). Przyjmujemy przy tym, że po zakończeniu dowolnej fazy (zakładając, że nie ma operatorów stop) rozpoczynamy sprawdzanie testów decydujących o wyborze fazy pro-

gramu. Testy muszą zapewniać jednoznaczny wybór fazy programu. W dalszym ciągu, zgodnie z wprowadzonymi wcześniej terminami, każdy zespół wartości warunków decydujący o wyborze (poprzez cały zespół testów) fazy programu będziemy nazywali *sytuacją decyzyjną*.

Podobnie jak z operatorami, składającymi się na program, możemy postąpić z testami określającymi sytuację decyzyjną i zapisać ją w formie tablicy. Ograniczmy się na początek do najprostszych testów sprawdzających stan zerojedynkowego indykatora. Podamy przykłady takich indykatorów:

- wykryto etykietę końca zbioru wejściowego (*tak* lub *nie*),

- rekord roboczy znajduje się w obszarze oczekiwania (*tak* lub *nie*),

- saldo dostawcy nr 1 jest dodatnie (*tak* lub *nie*),

- różnica między dwoma kolejnymi przybliżeniami rozwiązania jest mniejsza od 0,0001 (*tak* lub *nie*).

Sporządźmy listę wszystkich indykatorów zerojedynkowych w kolejności ich testowania. Napiszmy z kolei macierz o liczbie kolumn równej liczbie faz i liczbie wierszy równej liczbie indykatorów zerojedynkowych. Jeśli  $k$ -ty indykator nie jest testowany w  $j$ -tej sytuacji decyzyjnej, na przecięciu  $k$ -tego wiersza i  $j$ -tej kolumny położymy „—”. Jeśli  $k$ -ty indykator jest testowany w  $j$ -tej sytuacji decyzyjnej i wpływa na częściowy wybór tej sytuacji, gdy (indikator) jest w stanie *tak*, na przecięciu  $k$ -tego wiersza i  $j$ -tej kolumny położymy „T”. Jeśli wreszcie  $k$ -ty indykator jest testowany w  $j$ -tej sytuacji decyzyjnej, ale wpływa na częściowy wybór sytuacji decyzyjnej, gdy



(indykator) jest w stanie *nie*, na przecięciu  $k$ -tego wiersza i  $j$ -tej kolumny położymy „N”.

Łącząc dwie matryce i dwie listy w jedną całość<sup>1</sup>, otrzymamy najprostszy typ tablicy decyzyjnej. Zanim przejdziemy do przedstawienia przykładu tablicy decyzyjnej, zmniejszymy jeszcze ograniczenia, jakie nałożyliśmy na tablice decyzyjne. Postępowanie takie jest spowodowane koniecznością dostosowania tablic do potrzeb programowania:

1. Elementami „dolnej listy” tablicy decyzyjnej mogą być zarówno operatory, jak operatory poprzedzone testem decydującym o warunkowym wykonaniu operatora.

2. Elementy „dolnej listy” tablicy decyzyjnej mogą być modyfikowane w różny sposób w zależności od istniejącej sytuacji decyzyjnej.

3. Warunki podlegające testowaniu w „górnej części” tablicy decyzyjnej nie muszą mieć jedynie postaci indykatorów zerojedynkowych. W tym przypadku, na przecięciu kolumny i wiersza odpowiadającego złożonemu warunkowi, jeśli jest on testowany dla danej sytuacji decyzyjnej, umieszczamy symbol „L”. Natomiast na liście, na pozycji odpowiadającej danemu wierszowi matrycy, umieszczamy nazwę warunku, symbol „:”, a za nim rozdzielone przecinkami symbole „—” w miejscach odpowiadających symbolom „—” w wierszu matrycy, oraz wartości ujętej w symbole „ ” lub nazwy pola, lub złożonej formuły logicznej, w miejscach odpowiadających symbolom „L” w wierszu matrycy.

<sup>1</sup> Matrycę i listę indykatorów umieszczamy ponad matrycą i listą operatorów.

## 2.8. PRZYKŁAD TABLICY DECYZYJNEJ

Tablica rozpoczyna się etykietą: tablica „nazwa” i ewentualnie komentarzem umieszczonym w nawiasie, a kończy się etykietą: koniec tablicy. Tablica opisuje program (a raczej sieć działań programu), w którym wyróżniono sześć sytuacji decyzyjnych. Pierwszej sytuacji decyzyjnej odpowiada faza przygotowawcza, ostatniej — faza zakończeniowa. Pozostałe cztery fazy, to fazy robocze. Jako ćwiczenie dla czytelnika pozostawimy próbę czytania przykładowej tablicy decyzyjnej, zalecając przy tym ponowne przeczytanie zasad tworzenia tablic decyzyjnych. W dalszej części książki zamieścimy wiele tablic decyzyjnych, używając ich do opisu typowych przebiegów przetwarzania.

Dla ułatwienia zadania na rysunku 8 podany jest schemat blokowy sieci programu opisanego w tablicy 1.

Tablica 1

tablica „nazwa” (przykład tablicy decyzyjnej).

N T T N T T	indykator 1.
N N N T T T	indykator 2.
— T N — T N	indykator 3.
— L L — L L	wartość: —, „100”, „200”, —, „300”, „400”.
X — — X — —	operator 1.
X — — X — —	test z warunkowym przekazaniem sterowania do operatora 3.
X — — X — —	operator 2.
X — — X — —	operator przekazania sterowania do operatora 1.
X X X X — —	operator 3.
X X — — — —	test z warunkowym przekazaniem sterowania do operatora 7.
X X — — — —	operator 4.
X X — — — —	operator przekazania sterowania do operatora 3.
X X — X X —	operator 5.
— — — X — —	test z warunkowym przekazaniem sterowania do operatora 9.
X X — — — —	operator 6.

---

X X — — —	operator przekazania sterowania do operatora 3.
— — — X X —	operator przekazania sterowania do operatora 5.
X X — — —	test z warunkowym przekazaniem sterowania do operatora 5.
X X — — —	operator wywołania testowania sytuacji decyzyjnej.
X X — — —	operator 7.
X — — — —	operator 8.
— X X X — —	operator 9.
— — X — — —	operator 10.
— — X — — —	test z warunkowym przekazaniem sterowania do operatora 3.
— — — — — X	operator 11 (ze stopem końcowym).

koniec tablicy.

W celu zmniejszenia rozmiarów schematu blokowego, operator 1, test, operator 2 i operator przekazania sterowania do operatora 1 nazwiemy podprogramem A i pominiemy sieć działań tego podprogramu na rysunku 8. Podobnie postąpimy z operatorem 6, operatorem przekazania sterowania do operatora 3, testem, operatorem wywołania testowania sytuacji decyzyjnej i operatorem 7, nazywając je łącznie podprogramem B. Ponadto wprowadzimy oznaczenia:

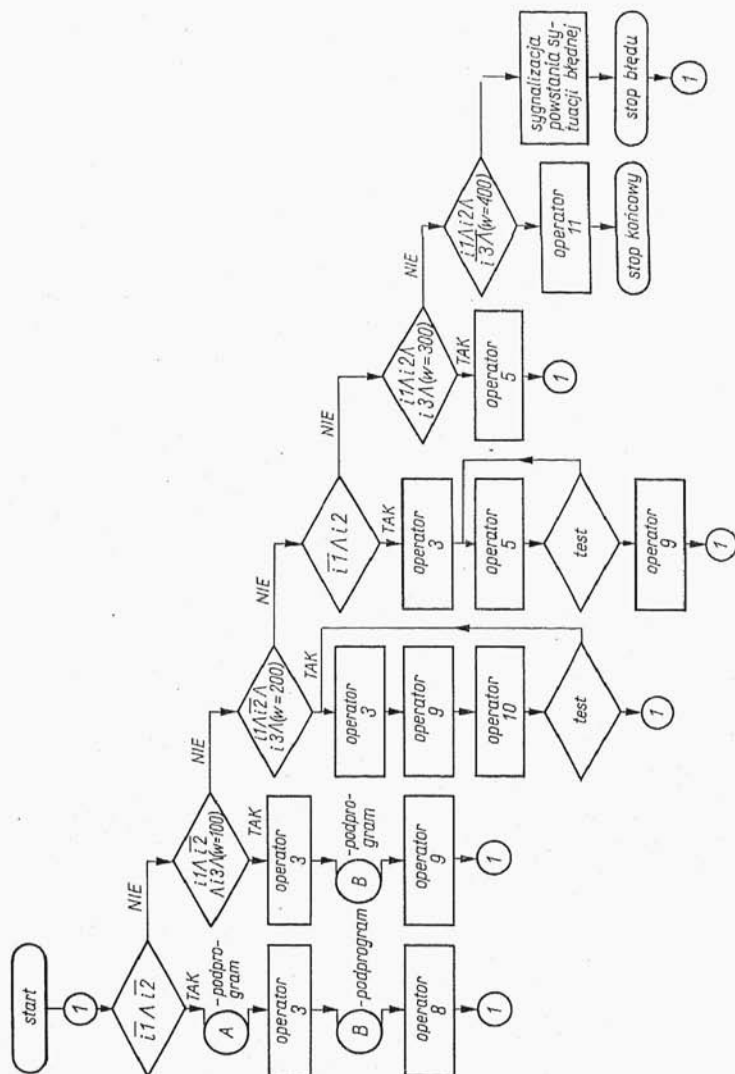
- i1 — zamiast indykator 1,
- i2 — zamiast indykator 2,
- i3 — zamiast indykator 3 oraz
- w — zamiast wartość.

Względem ograniczone rozmiary symbolu testu koniunkcję warunków elementarnych oznaczamy  $\wedge$ , zaś negację — przez kreskę poziomą ponad warunkiem (np.  $\overline{i1}$ ).<sup>1</sup>

Analizując przykładową tablicę decyzyjną, możemy stwierdzić, że zawiera ona „wewnątrz” przykłady podsieci liniowej (operatory 7 i 8 w pierwszej fazie), podsieci z iteracją (np. operator 1, test z warunkowym

---

<sup>1</sup> W dalszym ciągu, opisując warunki złożone w tablicach decyzyjnych, będziemy się posługiwali zarówno symbolem koniunkcji  $\wedge$ , jak i symbolem alternatywy  $\vee$ .



Rys. 8. Schemat blokowy odpowiadający tablicy 1

przekazaniem sterowania do operatora 3, operator 2, operator przekazania sterowania do operatora 1 i operator 3 — wszystkie w pierwszej fazie) itp.

## 2.9. UWAGI KOŃCOWE

Przedstawiona tablica decyzyjna jest stosunkowo prostym przykładem, ilustrującym jednak przydatność tej techniki dla opisywania sieci czy fragmentów sieci złożonych z wielofazowych programów przetwarzania danych. Dla uproszczenia tablicy decyzyjnej wygodnie jest niekiedy podzielić ją na kilka mniejszych tablic decyzyjnych. Podział taki może iść w dwu kierunkach:

1) w kierunku opisu części faz programu w ramach jednej tablicy; w ten sposób opis programu składa się z pewnej sekwencji tablic decyzyjnych;

2) w kierunku wprowadzenia podfaz (ewentualnie podpodfaz) towarzyszących jakiejś wybranej fazie (czy kilku fazom); w tym przypadku tablica decyzyjna opisująca podfazy jest jak gdyby podtablicą-podprogramem tablicy nadrzędnej i jest wywoływana przez przekazanie sterowania z wnętrza tablicy nadrzędnej, np. z przechowaniem śladu, czyli miejsca, do którego należy przekazać sterowanie do tablicy nadrzędnej po zakończeniu czynności opisanych podtablicą-podprogramem.

Dotychczasowe rozważania należy uzupełnić stwierdzeniem, że istnieje wiele fragmentów sieci programu, które ze względu na swą prostotę nie wymagają opisu przy użyciu tablic decyzyjnych. Dla zapewnienia jednak jednolitości opisu poszczególnych fragmentów sieci danego programu, wygodne jest wprowadzenie pseudo-

tablic, zawierających jedynie dolną część, z jedną jedynie kolumną po lewej stronie, wypełnioną przy wszystkich funkcjach prawej strony symbolami „X”. Funkcje występujące po prawej stronie zarówno w tablicach, jak i pseudotablicach decyzyjnych mogą być dowolnie złożoną podsiecią, w której np. pewien fragment może być opisywany za pomocą oddzielnej podtablicy, traktowanej jako podprogram.

Pominięliśmy dotąd opis danych (lub informacji), na których działa określony program lub które powstają w wyniku działania danego programu. Sprawie tej poświęcimy następny rozdział.