

214804L/1

INSTYTUT INŻYNIERII CHEMICZNEJ
I URZADZEN CIEPLNYCH
POLITECHNIKI WROCŁAWSKIEJ

Raport Nr 69/79
ser. PRE

PROGRAM ORGANIZACYJNY
MODELOWANIA I PROJEKTOWANIA
SYSTEMOW TECHNOLOGII CHEMICZNEJ
W STANIE USTALONYM

Lechosław J. Królikowski

Publikacja nie zawiera nowego
rozwiązania zagadnienia technicznego
nadającego się do opatentowania.

KIEROWNIK DZIAŁU
Ochrony Patentowej

Mgr inż. Bogusław Rozmus
Rzecznik Patentowy

Praca doktorska

Promotor: doc. dr hab. inż.

Eugeniusz Kuciel

Słowa kluczowe: system informacyjny, analiza systemów, kolej-
ność obliczania obiektów systemu, optymaliza-
cja systemu, modelowanie ciągów technologicz-
nych z zastosowaniem EMC

WROCŁAW 1979

80031503010

Panu doc. dr hab. inż. Eugeniuszowi
Kucielowi, promotorowi pracy, wyrazy
głębokiej wdzięczności za cenne rady,
opiekę i życzliwą atmosferę stworzoną
w czasie jej wykonywania składa

AUTOR

SPIIS TREŚCI

	str.
Streszczenie	
I. Wstęp. Cel pracy	1
II. Teoretyczne podstawy analizy systemów	5
1. System, system informacyjny	5
2. σ -podsystemy informacyjne i kolejność ich obliczania	13
3. Analiza jednopoziomowego zadania optymalizacji systemu	18
4. Analiza dwupoziomowego zadania optymalizacji systemu	33
5. Obliczanie σ -podsystemów informacyjnych	49
III. Ogólna charakterystyka programu organizacyjnego	54
IV. Opis stosowanych w programie organizacyjnym tablic	58
1. Podstawowe definicje	58
2. Tablice występujące w segmencie głównym programu organizacyjnego	62
3. Tablice występujące przy czytaniu danych	36
V. Opis zbiorów na taśmach magnetycznych	103
VI. Omówienie typowych procedur	110
VII. Nakładanie procedur w programie organizacyjnym	123
VIII. Czytanie danych przez program organizacyjny	125
IX. Drukowanie wyników przez program organizacyjny	132
X. Ogólna charakterystyka programu analizy systemów	140

XI. Czytanie danych przez program analizy systemów	142
1. Tablice występujące przy czytaniu danych	142
2. Kolejność czytania danych	147
XII. Drukowanie wyników przez program analizy systemów	150
XIII. Opis programu wprowadzającego dane do zbiorów na taśmach magnetycznych	160
XIV. Podsumowanie	162
XV. Literatura	163
Dodatek 1. Schemat blokowy programu organizacyjnego	
Dodatek 2. Schemat blokowy programu analizy systemów	
Dodatek 3. Przykład analizy systemu technologii che- micznej	168
Dodatek 4. Relacje	173
Dodatek 5. Grafy	175
Załącznik 1. List segmentu głównego programu organizacyj- nego	
Załącznik 2. List procedur READ, KEY, READ DATA 1 PRINT DATA	
Załącznik 3. List programu analizy systemów	
Załącznik 4. List programu wprowadzającego dane do zbiorów na taśmach magnetycznych.	

S T R E S Z C Z E N I E

Opracowano teoretyczne podstawy analizy systemów. Wprowadzono pojęcie systemu informacyjnego, który obrazuje kierunki przepływu informacji podczas obliczeń danego systemu /kierunki przepływu strumieni substancji i informacji nie muszą się pokrywać/. Określono kolejność obliczania grup obiektów połączonych informacyjnymi sprzężeniami zwrotnymi. Podano reguły określania tej kolejności w przypadku gdy mamy do czynienia z obliczeniami optymalizacji systemu. Zastosowanie ich umożliwia zmniejszenie czasu obliczeń i w konsekwencji może przynieść wymierne korzyści ekonomiczne. Sformułowano również warunki dostateczne poprawności zadania optymalizacji. Na podstawie otrzymanych wyników opracowano program analizy systemów ANALYZER. Opisano sposób przygotowania danych do tego programu i formę wydruków.

Scharakteryzowano działanie programu organizacyjnego BOSS, sterującego obliczeniami poszczególnych obiektów systemu. Program ten przygotowuje dane dla procedur obliczających obiekty systemu, wywołuje te procedury oraz magazynuje uzyskane przez nie wyniki. Omówiono tablice występujące w nagłówkach typowych procedur obliczeniowych /np. obiektów systemu/. Opisano zbiory informacji tworzone na taśmach magnetycznych, które są użytkowane przez standardowe procedury programu organizacyjnego. Podano zasady konstrukcji typowych procedur mających współpracować z programem BOSS. Omówiono formę i wyszczególniono kolejność wczytywania danych i drukowania wyników przez program

organizacyjny. Przedstawiono sposób wprowadzenia danych do zbiorów na taśmach magnetycznych /program ARCHIVIST/.

I. WSTĘP. CEL PRACY

W ostatnich kilkunastu latach powstało wiele komputerowych programów modelowania stanów ustalonych i projektowania systemów technologii chemicznej /patrz np. [6-8, 13,15, 19-21,26,27, 29,34,35,37,38]/. Stosuje się je między innymi w celu:

- prognozy wpływu zmiany warunków pracy i struktury systemu
- obliczenia bilansu materialnego i cieplnego
- optymalizacji warunków eksploatacji
- wykrycia i likwidacji wąskich gardeł produkcji
- otrzymania informacji o zachowaniu się systemu
- projektowania nowych zakładów przemysłowych
- szkolenia personelu inżynieryjno-technicznego

Umożliwiają one modelowanie lub projektowanie dowolnego systemu gdyż strukturę systemu podaje się w danych a podprogramy opisujące obiekty /mogą być nimi aparaty, części aparatów lub grupy aparatów/ dołączone są do programu organizacyjnego w miarę potrzeby z biblioteki podprogramów.

Wcześniejsze wersje programów /patrz np. [21,32] /mogły prowadzić obliczenia na jeden z następujących sposobów

1. dane : cechy /parametry/ strumieni^{*/} wejściowych systemu
i cechy /parametry/ obiektów

wyniki: cechy strumieni wyjściowych i pośrednich systemu

Kierunki przepływu informacji podczas obliczania systemu są zgodne z kierunkami przepływu strumieni substancji w procesie technologicznym.

^{*/} strumień - mieszanina substancji płynąca poza obiektami systemu

2. dane: cechy strumieni wyjściowych systemu i cechy obiektów
wyniki: cechy strumieni wejściowych i pośrednich systemu
Kierunki przepływu informacji podczas obliczeń systemu są
przeciwne do kierunków przepływu strumieni substancji w pro-
cesie technologicznym.
3. dane: cechy wszystkich strumieni systemu
wyniki: cechy obiektów
Nie ma przepływu informacji podczas obliczeń.

Przypadek 3 jest trywialny gdyż każdy obiekt można liczyć nie-
zależnie od pozostałych. Ostatnio opracowane programy /patrz
np. [20,29]/ oparto na następującej metodzie:

4. dane: pewne cechy strumieni wejściowych, wyjściowych i
pośrednich systemu oraz pewne cechy obiektów
wyniki: pozostałe cechy strumieni systemu i pozostałe cechy
obiektów
Kierunki przepływu informacji są częściowo zgodne, a częś-
ciowo przeciwne z kierunkami przepływu strumieni substancji
w procesie technologicznym /wyłączywszy oczywiście te cechy
strumieni pośrednich systemu, które są założone/.

Sposoby 1, 2 i 3 są szczególnymi przypadkami metody 4. Co wię-
cej zaciera się granica pomiędzy modelowaniem i projektowaniem
rozumianymi klasycznie. Jeżeli zadanie technologiczne narzuca
wartości pewnym cechom strumieni systemu, to zazwyczaj przy
tradycyjnym podejściu /metody 1 i 2/ obliczenia trzeba prowa-
dzić iteracyjnie. Oznacza to, że w obliczeniach zmieniamy war-
tości cech pełniących rolę zmiennych niezależnych starając się
trafić w zadane wartości określonych cech strumieni systemu.

Zwiększa to nakład pracy na obliczenia i podnosi ich koszty. Metoda 4, przy odpowiednio skonstruowanych procedurach opisujących obiekty systemu, pozwala nam uniknąć obliczeń iteracyjnych tego typu.

Zanim przystąpimy do obliczeń systemu musimy przeprowadzić /bez względu na to czy będziemy liczyć sposobem 1, 2 czy też 4/ analizę systemu tzn. wyodrębnić grupy obiektów powiązane informacyjnymi sprzężeniami zwrotnymi, znaleźć kolejność obliczania tych grup oraz strumienie rozerwane i kolejność obliczania obiektów dla każdej grupy. Opracowano wiele różnych algorytmów/[5, 12, 17, 21, 32, 33, 37, 41, 42, 45]/ realizujących wymienione wyżej zadania. Odczuwa się natomiast brak teoretycznych podstaw analizy systemów. Luki tej nie wypełnia praca [32] napisana jeszcze wtedy gdy nie znano metody 4 obliczania systemu ani praca [43] budząca obawy czy możliwe jest zastosowanie jej w praktyce /przy wykonywaniu analizy systemu/. Nie są również przydatne tutaj opracowania [14, 24, 44] dotyczące analizy układów równań, gdyż musielibyśmy przyjąć, że obiekt systemu technologii chemicznej jest opisany jednym równaniem i tylko jedna zmienna jest zmienną wyjściową /tzn. taką której wartość jest obliczana/. Mogą one natomiast oddać duże usługi przy analizie układów równań opisujących pojedynczy obiekt systemu.

Przy obliczaniu grup obiektów powiązanych informacyjnymi sprzężeniami zwrotnymi stajemy wobec problemu przyspieszenia zbieżności procesu iteracyjnego. Na szczęście w chwili obecnej istnieje wiele sposobów rozwiązania tego zagadnienia /patrz np. [21, 32, 35, 36]/. Bogata literatura poświęcona jest również metodom obliczeniowym optymalizacji /patrz np. [10, 11, 16, 21, 23, 30-32]/.

Cel pracy: opracowanie teoretycznych podstaw analizy systemów
oraz programów organizacyjnego i analizy systemów.

II. TEORETYCZNE PODSTAWY ANALIZY SYSTEMÓW

1. SYSTEM, SYSTEM INFORMACYJNY

Niech X będzie zbiorem obiektów, a C zbiorem cech. Niech będzie dany operator $A : C \rightarrow X$ odwzorowujący zbiór C na zbiór X oraz relacja λ określona dla elementów zbioru C mająca następujące własności.

1. dla każdego $c_p \in C$ istnieje co najwyżej jedno $c_r \in C$ takie, że $c_p \lambda c_r$
2. $\sim c \lambda c$ dla każdego $c \in C$
3. jeżeli $c_p \lambda c_r$ to $\sim c_r \lambda c_p$ dla każdych $c_p, c_r \in C$ i $c_p \neq c_r$
4. dla każdych $x_i, x_j \in X$ istnieje ciąg $c_{p_1}, c_{p_2}, \dots, c_{p_{2t}}$ taki że $A(c_{p_1}) = x_i, A(c_{p_{2t}}) = x_j, c_{p_{2k-1}} \lambda c_{p_{2k}}$ albo $c_{p_{2k}} \lambda c_{p_{2k+1}}$ dla $k = 1, 2, \dots, t$ i $A(c_{p_{2k}}) = A(c_{p_{2k+1}})$ dla $k = 1, 2, \dots, t$

Czwórkę $\langle X, C, A, \lambda \rangle$ będziemy nazywali systemem. Oznaczmy

$$H = \{(c_p, c_r) : c_p \lambda c_r\}$$

Zbiory $S_1, S_2, \dots, S_m$ par cech takie, że

1. jeżeli $(c_p, c_r), (c_s, c_t) \in S_k$ to $A(c_p) = A(c_s)$ i $A(c_r) = A(c_t)$ dla $k = 1, 2, \dots, m$
2. $S_k \cap S_1 = \emptyset$ dla $k \neq 1$
3. $\bigcup_{k=1}^m S_k = H$

będziemy nazywali strumieniami pośrednimi systemu. Naturalnym jest założenie, że dla dowolnego elementu (c_p, c_r) każdego strumienia pośredniego wartość cechy c_p równa się wartości cechy c_r . Z tego względu możemy przyjąć za cechę reprezentującą element

(c_p, c_r) strumienia pośredniego S_k albo cechę c_p albo cechę c_r . W konsekwencji strumień S_k może zostać scharakteryzowany /wyłącz-
wszy jego kierunek/ przez $\bar{S}_k$ cech, które będziemy nazywali cechami strumienia pośredniego. Oznaczmy

$$C_S = \{c_p : \bigvee_{c_r \in C} c_p \wedge c_r \vee c_r \wedge c_p\} \quad (1.1)$$

- zbiór cech wchodzących w skład elementów strumieni pośrednich. Cechy nie wchodzące w skład elementów strumieni pośrednich tzn. cechy ze zbioru $C - C_S$ dzielimy na cechy obiektów, cechy strumieni wejściowych systemu i cechy strumieni wyjściowych systemu. Podzbiory zawierające cechy poszczególnych typów oznaczamy odpowiednio C_C, C_I, C_O . Muszą one spełniać warunki:

1. $C - C_S = C_C \cup C_I \cup C_O$
2. $C_C \cap C_I = \emptyset$
3. $C_C \cap C_O = \emptyset$
4. $C_I \cap C_O = \emptyset$

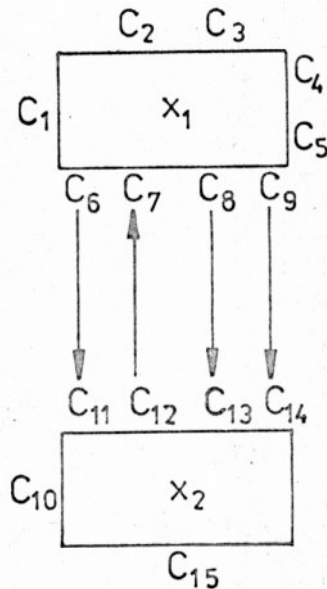
Strumieniami wejściowymi systemu będziemy nazywać zbiory $S_1^I, S_2^I, \dots, S_{m_I}^I$ takie, że

1. jeżeli $c_p, c_r \in S_k^I$ to $A(c_p) = A(c_r)$ dla $k = 1, 2, \dots, m_I$
2. $S_k^I \cap S_l^I = \emptyset$ dla $k \neq l$
3. $\bigcup_{k=1}^{m_I} S_k^I = C_I$

Analogicznie za strumienie wyjściowe systemu przyjmujemy zbiory $S_1^O, S_2^O, \dots, S_{m_O}^O$ takie, że

1. jeżeli $c_p, c_r \in S_k^0$ to $A(c_p) = A(c_r)$ dla $k = 1, 2, \dots, m_0$
2. $S_k^0 \cap S_l^0 = \emptyset$ dla $k \neq l$
3. $\bigcup_{k=1}^{m_0} S_k^0 = C_0$

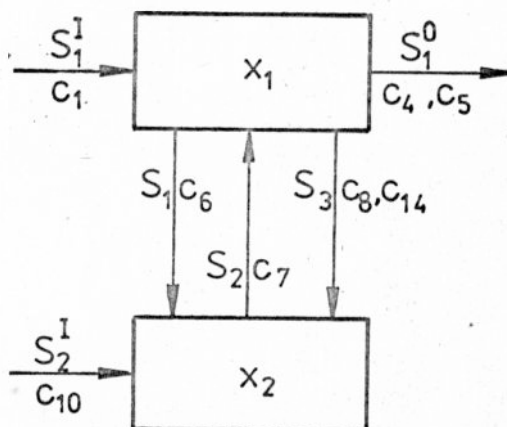
Przykład 1.1: Niech $X = \{x_1, x_2\}$ a $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}, c_{13}, c_{14}, c_{15}\}$. Niech operator A przyporządkowuje cechy $c_1 \div c_9$ obiektowi x_1 , a cechy $c_{10} \div c_{15}$ obiektowi x_2 . Relację λ określamy następująco: cecha c_6 jest w relacji z cechą c_{11} , cecha c_{12} z cechą c_7 , cecha c_8 z cechą c_{13} i cecha c_9 z cechą c_{14} . System (X, C, A, λ) możemy przedstawić jak na rysunku II.1.1.



Rys. II.1.1

Strzałki pomiędzy cechami ilustrują tutaj relację λ . Za strumienie pośrednie możemy przyjąć np. $S_1 = \{(c_6, c_{11})\}$, $S_2 = \{(c_{12}, c_7)\}$ i $S_3 = \{(c_8, c_{13}), (c_9, c_{14})\}$. Definicja λ i przyjęcie takich a nie innych strumieni pośrednich zależą wyłącznie od strumieni substancji w danym procesie technologicznym. Określmy $C_C = \{c_2, c_3, c_{15}\}$, $C_I = \{c_1, c_{10}\}$ i $C_O = \{c_4, c_5\}$. Za strumienie wejściowe systemu możemy przyjąć $S_1^I = \{c_1\}$ i $S_2^I = \{c_{10}\}$ natomiast za strumień wyjściowy systemu $S_1^O = \{c_4, c_5\}$ /zależy to również od procesu technologicznego/. Nasz system jest przedstawiony na rys. II.1.2 ze strumieniami, które zaznaczone są strzałkami /kierunek strzałki wskazuje kierunek przepływu strumienia substancji w procesie technologicznym/ i scharakteryzowane przez cechy /wybór cech opisujących strumienie pośrednie może być oczywiście inny/.

Cechami ze zbioru C_C mogą być np. stała zależna od katalizatora, średnica zbiornika, powierzchnia wymiany ciepła itp, natomiast



Rys. II.1.2

cechami ze zbioru $C = C_C$: temperatura, ciśnienie, molowe natężenie przepływu, ułamek molowy składnika.

Zdefiniujmy funkcjonał $F: C \rightarrow \{0,1\}$ w następujący sposób:

$$F(c) = \begin{cases} 0, & \text{jeżeli obiekt } A(c) \text{ sam wyznacza wartość cechy } c \text{ tzn.} \\ & \text{gdy algorytm opisujący obiekt } A(c) \text{ ma możliwość ob-} \\ & \text{liczenia wartości tej cechy.} \\ 1, & \text{jeżeli obiekt } A(c) \text{ wymaga aby wartość cechy } c \text{ została} \\ & \text{wyznaczona poza nim tzn. gdy algorytm opisujący obiekt} \\ & A(c) \text{ wymaga wartości cechy } c \text{ jako danej} \end{cases}$$

Określmy teraz operator $B: H \rightarrow \{0,1\} \times \{0,1\}$ jak następuje

$$B((c_p, c_r)) = (F(c_p), F(c_r))$$

Przeciwobrazy $B^{-1}((0,0))$, $B^{-1}((0,1))$, $B^{-1}((1,0))$ i $B^{-1}((1,1))$ są parami rozłączne. Możemy teraz zdefiniować dla elementów zbioru C relację μ , opisującą rzeczywiste kierunki przepływu informacji pomiędzy cechami i za ich pośrednictwem pomiędzy obiektami z którymi są związane, w następujący sposób

$$c_p \mu c_r \Leftrightarrow (c_p, c_r) \in B^{-1}((0,1)) \vee (c_r, c_p) \in B^{-1}((1,0))$$

Jeżeli $B^{-1}((0,0)) = \emptyset$ to piętkę (X, C, A, F, μ) będziemy nazywać systemem informacyjnym odpowiadającym danemu systemowi. Założenie $B^{-1}((0,0)) = \emptyset$ eliminuje nam przypadek w którym dla pewnych elementów strumieni pośrednich systemu wartości cech elementu wyznaczane były przez różne obiekty. Mogłoby to być sprzeczne z naszym założeniem, że wartości obu cech każdego elementu strumienia pośredniego muszą być takie same.

Oznaczamy

$$W = \{(c_p, c_r) : c_p \mu c_r\}$$

Określamy dla elementów zbioru W relację δ

$$(c_p, c_r) \delta (c_s, c_t) \iff A(c_p) = A(c_s) \wedge A(c_r) = A(c_t)$$

Jest oczywiste, że δ jest relacją równoważności i jako taką ustala podział zbioru W na klasy równoważności, które będziemy nazywać strumieniami informacyjnymi i oznaczać literą Y z różnymi indeksami w razie potrzeby.

Przykład 1.2: Niech funkcjonał F będzie zdefiniowany dla systemu z przykładu 1.1 następująco

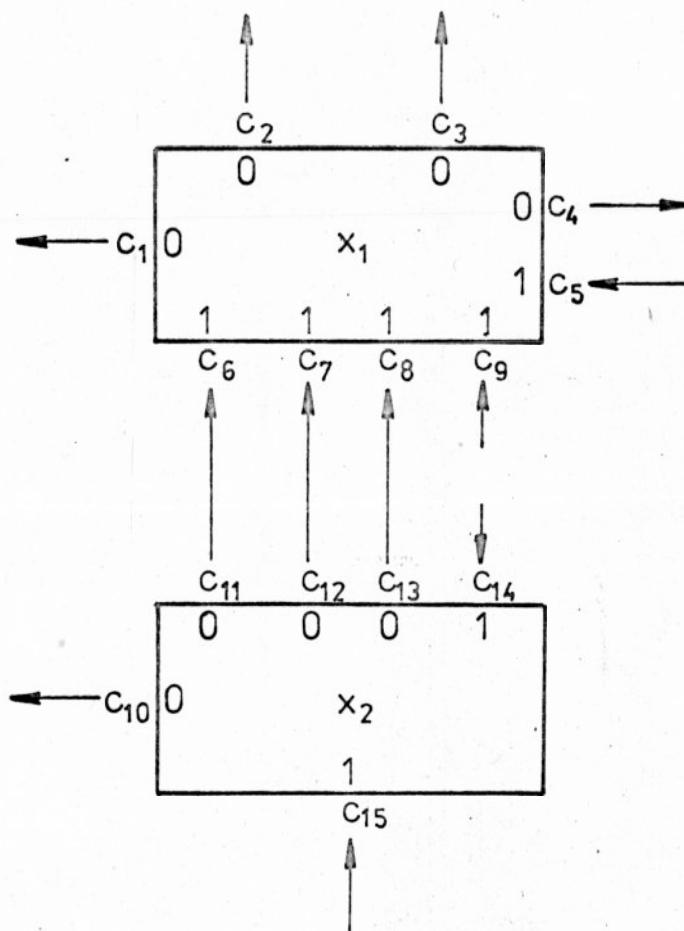
$$F(c) = \begin{cases} 0 & \text{dla } c \in \{c_1, c_2, c_3, c_4, c_{10}, c_{11}, c_{12}, c_{13}\} \\ 1 & \text{dla } c \in \{c_5, c_6, c_7, c_8, c_9, c_{14}, c_{15}\} \end{cases}$$

System informacyjny odpowiadający systemowi z przykładu 1.1 jest przedstawiony na rys. II.1.3. Strzałki pokazują kierunki przepływu informacji i dodatkowo strzałki łączące dwie cechy opisują relację μ . Dla każdej cechy podano wewnątrz obiektu wartość funkcjonału. Zauważmy także, że mamy tutaj tylko jeden strumień informacyjny $Y = \{(c_{11}, c_6), (c_{12}, c_7), (c_{13}, c_8)\}$.

Aby móc przedstawić strukturę powiązań obiektów w systemie informacyjnym w postaci grafu, definiujemy poniżej dla elementów zbioru X relację ϱ

$$x_i \varrho x_j \iff \bigvee_{(c_p, c_r) \in W} A(c_p) = x_i \wedge A(c_r) = x_j$$

Strukturę systemu informacyjnego opisuje nam graf $\langle X, \varrho \rangle$. Łuki tego grafu reprezentują strumienie informacyjne ze zbioru W/δ . Graf $\langle X, \varrho \rangle$ nie musi być spójny. Będziemy zakładać dla wszyst-



Rys. II.1.3

kich dalszych rozważań prowadzonych w rozdziale II, że

$$\bigwedge_{x \in X} \text{ istnieje droga } L(x, x) \text{ o długości } \geq 0$$

Relacja τ określona dla elementów zbioru X wzorem

$$x_i \tau x_j \iff \text{ istnieje łańcuch od } x_i \text{ do } x_j$$

jest wtedy relacją równoważności i ustala nam podział zbioru X na klasy równoważności. W konsekwencji można podzielić graf $\langle X, \rho \rangle$ na podgrafy $\langle N, \rho_N \rangle$ gdzie $N \in X/\tau$ które są albo spójne albo ich rząd jest równy jedności. Rodzinę tych podgrafów

będziemy oznaczać $\mathcal{G}(X/\tau)$.

Zauważmy, że każde dwa podsystemy informacyjne odpowiadające dwu różnym podgrafom z rodziny $\mathcal{G}(X/\tau)$ mogą być obliczane niezależnie od siebie /nie występują między nimi przepływy informacji/ jeżeli dla żadnego elementu (c_p, c_r) ze zbioru $B^{-1}((1,1))$ nie przyjmujemy wartości cechy c_p różnej od wartości cechy c_r .

Przykład 1.3: Graf opisujący strukturę systemu informacyjnego z przykładu 1.2 jest spójny i wygląda następująco:



Rys. II.14

2. σ - PODSYSTEMY INFORMACYJNE I KOLEJNOŚĆ ICH OBLICZANIA

Przypomnijmy, że $\langle X, \rho \rangle$ opisuje strukturę systemu informacyjnego. Dla elementów zbioru X określamy relację σ wzorem

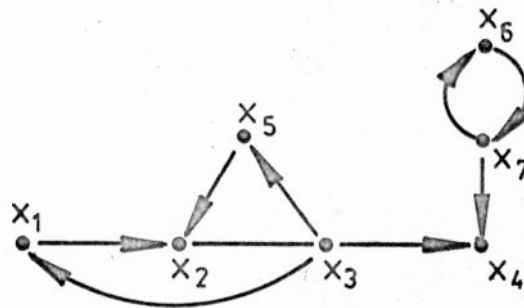
$$x_i \sigma x_j \Leftrightarrow \text{istnieją drogi } L(x_i, x_j) \text{ i } L(x_j, x_i) \text{ o długości } \geq 0$$

Relacja σ jest relacją równoważności i ustala podział zbioru X na klasy równoważności. Oznaczmy

$$\mathcal{G}(X/\sigma) = \{ \langle M, \rho_M \rangle : M \in X/\sigma \}$$

Zauważmy, że jeżeli rząd podgrafu z tej rodziny jest większy od jedności to każde dwa różne wierzchołki tego podgrafu wchodzi w skład pewnego kontum. Jeżeli natomiast rząd podgrafu jest równy jedności to podgrafem jest wierzchołek z pętlą lub bez pętli. Podsystemy informacyjne odpowiadające podgrafom z $\mathcal{G}(X/\sigma)$ będziemy nazywali σ -podsystemami informacyjnymi. Jeżeli $X/\sigma = \{X\}$ to system informacyjny będziemy nazywali σ -systemem informacyjnym.

Przykład 2.1: Niech strukturę systemu informacyjnego przedstawia nam graf z rys. II.2.1



Rys. II. 2.1

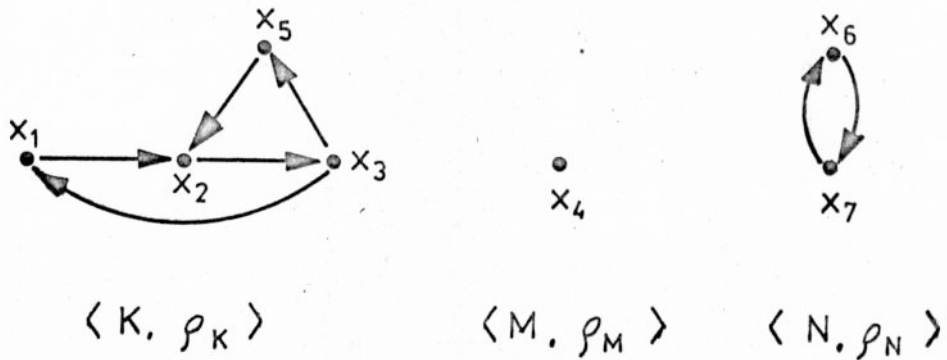
Relacja σ dzieli nam zbiór $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$ na następujące klasy równoważności:

$$K = \{x_1, x_2, x_3, x_5\}$$

$$M = \{x_4\}$$

$$N = \{x_6, x_7\}$$

$X/\sigma = \{K, M, N\}$ a $\mathcal{G}(X/\sigma)$ składa się z grafów pokazanych na rys. 2.2



Rys. II. 2.2

lemat 2.1: Jeżeli $M, N \in X/\sigma$, $M \neq N$, $x_i \in M$, $x_j \in N$ i istnieje droga $L(x_i, x_j)$ to dla wszystkich $x_k \in M$ i $x_l \in N$ istnieje droga $L(x_k, x_l)$ i nie istnieje droga $L(x_l, x_k)$.

Istotnie tak jest gdyż istnieją drogi $L(x_k, x_i)$ i $L(x_j, x_l)$ a więc istnieje droga $L(x_k, x_l)$. To, że nie istnieje droga $L(x_l, x_k)$ wynika natychmiast z przynależności x_i oraz x_j do różnych klas równoważności. $\square$

Określmy teraz dla podgrafów z rodziny $\mathcal{G}(X/\sigma)$ relację $\leq$ w następujący sposób

$$\langle M, \varrho_M \rangle \leq \langle N, \varrho_N \rangle \iff \text{dla pewnych } x_i \in M \text{ i } x_j \in N \text{ istnieje} \quad (2.1) \\ \text{droga } L(x_i, x_j) \text{ o długości } \geq 0$$

Jest oczywiste, że relacja $\leq$ jest zwrotna, gdyż dla dowolnego $x \in M$ gdzie $M \in X/\sigma$, istnieje droga $L(x, x)$ o długości równej zero. Jest także antysymetryczna. Niech podgrafy $\langle M, \varrho_M \rangle$ i $\langle N, \varrho_N \rangle$ będą takie, że

$$\langle M, \varrho_M \rangle \leq \langle N, \varrho_N \rangle \quad \text{ i } \quad \langle N, \varrho_N \rangle \leq \langle M, \varrho_M \rangle$$

Niech $x_i \in M$ i $x_j \in N$. Z definicji relacji $\leq$ oraz z lematu 2.1 wynika, że istnieją drogi $L(x_i, x_j)$ oraz $L(x_j, x_i)$. Oznacza to, że $x_i \sigma x_j$ a więc $x_i, x_j \in M$ i $x_i, x_j \in N$. Ponieważ σ jest więc relacją równoważności, $M = N$ i w konsekwencji $\langle M, \varrho_M \rangle = \langle N, \varrho_N \rangle$. $\square$

Relacja $\leq$ jest również przechodnia. Rzeczywiście. Niech

$$\langle K, \varrho_K \rangle \leq \langle M, \varrho_M \rangle \quad \text{ i } \quad \langle M, \varrho_M \rangle \leq \langle N, \varrho_N \rangle$$

Weźmy $x_i \in K$, $x_j \in M$, $x_k \in N$. Na podstawie definicji relacji $\leq$ i lematu dostajemy, że istnieją drogi $L(x_i, x_j)$ oraz $L(x_j, x_k)$. Zatem istnieje droga $L(x_i, x_k)$ co pociąga za sobą, że

$$\langle K, \varrho_K \rangle \leq \langle N, \varrho_N \rangle \quad \square$$

Ostatecznie ponieważ relacja $\leq$ jest zwrotna, antysymetryczna i przechodnia więc jest relacją porządkującą rodzinę $\mathcal{G}(X/\sigma)$. Jeżeli $X/\tau = \{X\}$ to akceptujemy to uporządkowanie bez zastrzeżeń. Jest ono zgodne z kierunkami przepływu informacji pomiędzy

σ -podsystemami informacyjnymi, tzn. że jeżeli jeden graf poprzedza drugi to informacja jest przesyłana od σ -podsystemu informacyjnego odpowiadającego pierwszemu grafowi do σ -podsystemu informacyjnego odpowiadającego drugiemu. Zatem σ -podsystemy informacyjne możemy obliczać w kolejności wyznaczonej uporządkowaniem odpowiadających im podgrafów z $\mathcal{G}(X/\sigma)$. Jeżeli natomiast $X/\tau \neq \{X\}$ to każdą rodzinę $\mathcal{G}(N/\sigma)$ gdzie $N \in X/\tau$ porządkuje nam relacja $\leq_{\mathcal{G}(N/\sigma)}$ będąca relacją $\leq$ ograniczoną do rodziny $\mathcal{G}(N/\sigma)$. Relacja $\leq_{\mathcal{G}(N/\sigma)}$ jest dobrze określona gdyż $\mathcal{G}(N/\sigma) \subset \mathcal{G}(X/\sigma)$. Rzeczywiście. Niech $x_i \in N$ i $x_j \in X-N$. Mamy $\sim x_i \tau x_j$, a więc $\sim x_i \sigma x_j$. Zatem $N/\sigma \subset X/\sigma$ i w konsekwencji $\mathcal{G}(N/\sigma) \subset \mathcal{G}(X/\sigma)$. σ -podsystemy informacyjne odpowiadające podgrafom z rodziny $\mathcal{G}(N/\sigma)$ obliczamy wg kolejności wyznaczonej uporządkowaniem tej rodziny przez relację $\leq_{\mathcal{G}(N/\sigma)}$. Również w tym przypadku uporządkowanie jest zgodne z kierunkami przepływu informacji pomiędzy σ -podsystemami informacyjnymi odpowiadającymi podgrafom z rodziny $\mathcal{G}(N/\sigma)$. Poza tym nie ma żadnego przepływu informacji pomiędzy dwoma σ -podsystemami informacyjnymi odpowiadającymi podgrafom z dwóch różnych rodzin $\mathcal{G}(M/\sigma)$ i $\mathcal{G}(N/\sigma)$ gdzie $M, N \in X/\tau$ i $M \neq N$ /patrz: definicja relacji τ /. Zatem każdą rodzinę σ -podsystemów informacyjnych odpowiadających podgrafom z $\mathcal{G}(N/\sigma)$ dla ustalonego $N \in X/\tau$ możemy obliczać niezależnie od pozostałych σ -podsystemów informacyjnych.

Przykład 2.2: Graf systemu informacyjnego przedstawiony na rys. II.2.1 jest taki, że $X/\tau = \{X\}$. Dla podgrafów z rys. II.2.2 mamy

$$\langle K, \varrho_K \rangle \leq \langle M, \varrho_M \rangle \quad \text{ i } \quad \langle N, \varrho_N \rangle \leq \langle M, \varrho_M \rangle$$

Odpowiadające im σ -podsystemy informacyjne możemy obliczać w kolejności:

1. σ -podsystem informacyjny o strukturze $\langle K, \varrho_K \rangle$
2. σ -podsystem informacyjny o strukturze $\langle N, \varrho_N \rangle$
3. σ -podsystem informacyjny o strukturze $\langle M, \varrho_M \rangle$

lub

1. σ -podsystem informacyjny o strukturze $\langle N, \varrho_N \rangle$
2. σ -podsystem informacyjny o strukturze $\langle K, \varrho_K \rangle$
3. σ -podsystem informacyjny o strukturze $\langle M, \varrho_M \rangle$

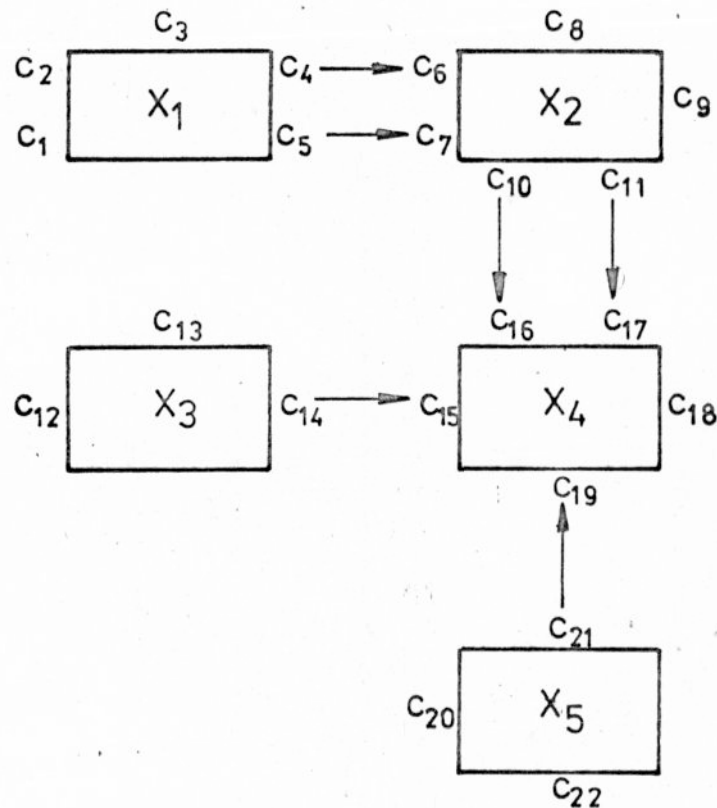
UWAGA:

Zdefiniowana w literaturze [32] relacja $<$ uporządkowania podsystemów /zespołów w tamtejszej terminologii/ jest szczególnym przypadkiem relacji $\leq$ określonej dla podgrafów z $\mathcal{G}(X/\sigma)$.

3. ANALIZA JEDNOPOZIOMOWEGO ZADANIA OPTIMALIZACJI SYSTEMU

Relacja $\leq$ określona wzorem /2.1/ nie zapewnia nam dla zadania optymalizacji systemu takiego uporządkowania rodziny $\mathcal{G}(X/\sigma)$, które wyznaczałoby nam prawidłową kolejność obliczania σ -podsystemów informacyjnych.

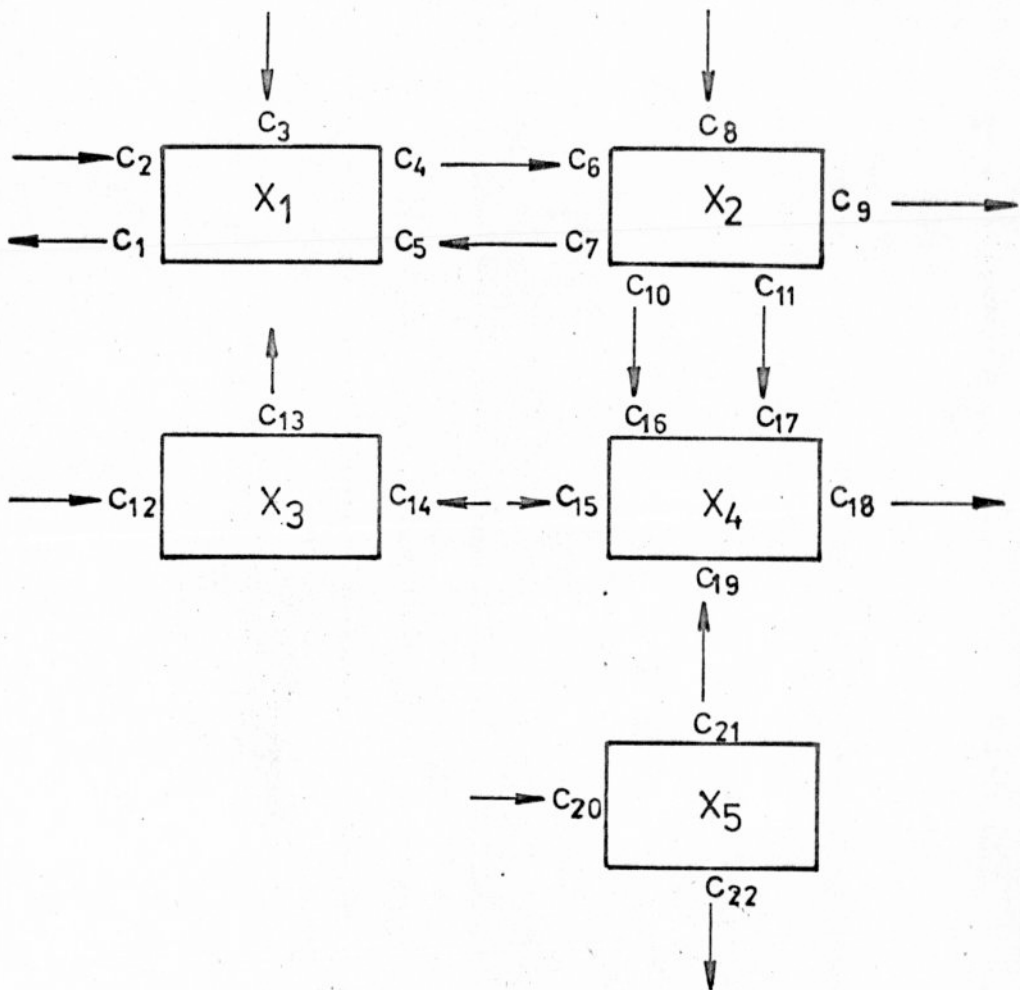
Przykład 3.1: Niech będzie dany system jak na rys. II.3.1.



Rys. II. 3.1

Niech odpowiadający mu system informacyjny wygląda jak na rys. II.3.2.

Zadanie optymalizacji jest sformułowane następująco: Za kryterium optymalności przyjmujemy minimalną wartość cechy c_{18} , a za zmienne decyzyjne cechę c_8 i element (c_{14}, c_{15}) . Wartości cech c_2, c_3, c_{12}, c_{20} są potrzebne jako dane i nie zmieniają się w

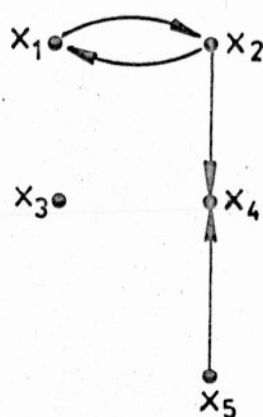


Rys. II.3.2

trakcie obliczeń optymalizacyjnych.

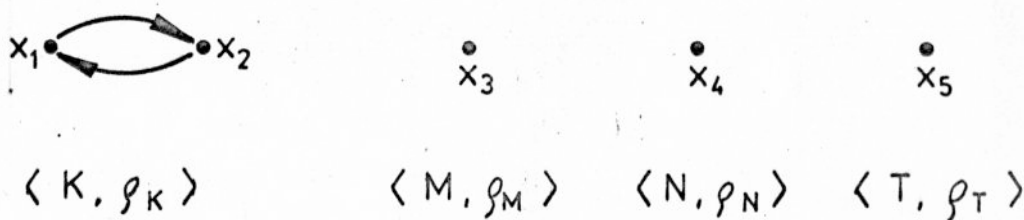
Uwaga: Nie możemy przyjąć za zmienną decyzyjną samej cechy c_{15} gdyż (c_{14}, c_{15}) jest elementem strumienia pośredniego i wartość cechy c_{14} musi być równa wartości cechy c_{15} . Przed przystąpieniem do obliczeń optymalizacyjnych na EMC trzeba oczywiście zadać wartości początkowe cech c_8 i c_{15} /wartość cechy c_{14} jest wtedy również znana/.

Graf $\langle X, \varrho \rangle$ opisujący strukturę systemu jest przedstawiony na rys. II.3.3.



Rys. II. 3.3

Zbiór X/σ składa się z następujących klas równoważności:
 $K = \{x_1, x_2\}$, $M = \{x_3\}$, $N = \{x_4\}$ i $T = \{x_5\}$. Rodzinę $\mathcal{G}(X/\sigma)$ prezentujemy na rys. II.3.4.



Rys. II. 3.4

Relacja $\leq$ ustala wśród tych podgrafów następujące zależności

$$\langle K, \rho_K \rangle \leq \langle N, \rho_N \rangle \quad \text{i} \quad \langle T, \rho_T \rangle \leq \langle N, \rho_N \rangle$$

Kolejność obliczania σ -podsystemów informacyjnych wyznaczone uporządkowaniem rodziny $\mathcal{G}(X/\sigma)$ przez relację $\leq$ może być na przykład następująca:

→ 1. σ -podsystem informacyjny o strukturze $\langle M, \rho_M \rangle$

- 2. - podsystem informacyjny o strukturze $\langle K, e_K \rangle$
- 3. - podsystem informacyjny o strukturze $\langle T, e_T \rangle$
- 4. - podsystem informacyjny o strukturze $\langle N, e_N \rangle$

Taka kolejność obliczeń na EMC jest nieprawidłowa, gdyż

1. σ - podsystem informacyjny o strukturze $\langle T, e_T \rangle$ będzie obliczany niepotrzebnie dla każdej iteracji a wystarczyłoby jednokrotne wyznaczenie wartości cech c_{21} i c_{22} na początku obliczeń
2. σ - podsystem informacyjny o strukturze $\langle M, e_M \rangle$ musi być również obliczany iteracyjnie ponieważ wartość cechy c_{14} po zakończeniu obliczeń optymalizacyjnych będzie z reguły inna niż jej wartość początkowa. Można byłoby tego uniknąć obliczając go na końcu gdy wartość cechy c_{14} jest już ustalona.

Niżej podajemy prawidłową kolejność obliczeń /zgodną również z uporządkowaniem $G(X/\sigma)$ przez relację $\leq /$:

1. σ - podsystem informacyjny o strukturze $\langle T, e_T \rangle$
2. σ - podsystem informacyjny o strukturze $\langle N, e_N \rangle$
3. σ - podsystem informacyjny o strukturze $\langle K, e_K \rangle$
4. σ - podsystem informacyjny o strukturze $\langle M, e_M \rangle$

Zmiennymi decyzyjnymi w zadaniu optymalizacji mogą być wybrane elementy strumieni pośrednich systemu /będziemy je nazywali elementami decyzyjnymi/ i wybrane cechy nie wchodzące w skład strumieni pośrednich /będziemy je nazywali cechami decyzyjnymi/. Wartość funkcji celu będziemy obliczali na podstawie wartości wybranych cech zależących pośrednio od zmiennych decy-

zyjnych. Cechy te mogą wchodzić w skład elementów strumieni pośrednich lub nie. Elementy strumieni pośrednich w skład których wchodzi takie cechy będziemy nazywali elementami funkcji celu, natomiast te z wybranych cech, które niewchodzą w skład elementów strumieni pośrednich będziemy nazywali cechami funkcji celu. Przez zmienne funkcji celu będziemy rozumieli cechy i elementy funkcji celu.

Uwaga: Element decyzyjny/a także element funkcji celu / może być reprezentowany przez jedną ze swoich cech będącą jednocześnie cechą strumienia pośredniego.

Niech będzie dany pewien system i odpowiadający mu system informacyjny. Niech dla pewnego zadania optymalizacji D' i D'' będą zbiorami cech i elementów decyzyjnych odpowiednio, a V' i V'' zbiorami cech i elementów funkcji celu. Zbiory te muszą spełniać następujące warunki

$$D' \subset F^{-1}(1) - C_S \quad (3.1)$$

$$D'' \subset B^{-1}((1,1)) \quad (3.2)$$

$$V' \subset F^{-1}(0) - C_S \quad (3.3)$$

$$V'' \subset B^{-1}((0,1)) \cup B^{-1}((1,0)) \quad (3.4)$$

Analizę zadania optymalizacji będziemy przeprowadzać w odniesieniu do systemu informacyjnego, którego strukturę opisuje graf $\langle X, \rho \rangle$. Oznaczmy

$$V = V' \cup \{c_p : (c_p, c_r) \in V'' \cap B^{-1}((0,1)) \vee (c_r, c_p) \in V'' \cap B^{-1}((1,0))\}$$

i

$$X_V = A(V)$$

V jest więc zbiorem cech z których bezpośrednio się liczy wartość funkcji celu, a X_V zbiorem obiektów wyznaczających /obliczających/ wartości tych cech. Zbiór V będziemy nazywali zbiorem wyjściowym a zbiór X_V zbiorem końcowym. Żądamy aby zbiory D', D'', V', V'' spełniały również następujące warunki

$$D' \neq \emptyset \vee D'' \neq \emptyset \quad (3.5)$$

$$V' \neq \emptyset \vee V'' \neq \emptyset \quad (3.6)$$

$$\bigwedge_{c \in D' \neq \emptyset} \bigvee_{x \in X_V} \text{istnieje } L(A(c), x) \text{ o długości } \geq 0 \quad (3.7)$$

$$\bigwedge_{(c_p, c_r) \in D'' \neq \emptyset} \bigvee_{x \in X_V} \text{istnieje } L(A(c_p), x) \text{ lub } L(A(c_r), x) \text{ o długości } \geq 0 \quad (3.8)$$

Warunek /3.5/ gwarantuje istnienie zmiennych decyzyjnych. Warunek /3.6/ zapewnia możliwość obliczenia wartości funkcji celu. Natomiast warunki /3.7/ i /3.8/ eliminują pozorne zmienne decyzyjne. Oznaczmy

$$D = D' \cup \{c_p : [(c_p, c_r) \in D'' \vee (c_r, c_p) \in D''] \wedge \bigvee_{x \in X_V} \text{istnieje } L(A(c_p), x) \text{ o dż. } \geq 0\}$$

i

$$X_D = A(D)$$

D jest więc zbiorem cech ze zmiennych decyzyjnych, które wpływają na wartość funkcji celu, a X_D zbiorem obiektów, którym przyporządkowane są te cechy. Zbiór D będziemy nazywali zbiorem wejściowym, a zbiór X_D zbiorem początkowym. Muszą także być spełnione dwa poniższe warunki

$$\bigwedge_{c \in V' \neq \emptyset} \bigvee_{x \in X_D} \text{istnieje } L(x, A(c)) \text{ o długości } \geq 0 \quad (3.9)$$

$$\bigwedge_{(c_p, c_r) \in V'' \neq \emptyset} \bigvee_{x \in X_D} \text{istnieje } L(x, A(c_p)) \text{ lub } L(x, A(c_r)) \text{ o d\kern-1pt\lrcorner} \geq 0 \quad (3.10)$$

Warunki te eliminują spośród cech i elementów funkcji celu te, które nie zależą od zmiennych decyzyjnych. Ciąg obiektów

$x_{k_1}, x_{k_2}, \dots, x_{k_t}$ taki, że

$$1. \quad x_{k_i} \in K \subset X \quad \text{dla } i = 1, 2, \dots, t$$

$$2. \quad x_{k_i} \varrho x_{k_{i+1}} \text{ lub } x_{k_{i+1}} \varrho x_{k_i} \text{ lub} \\ = \{x_{k_i}, x_{k_{i+1}}\}$$

$$\bigvee_{(c_p, c_r) \in N \subset B^{-1}((1, 1))} \{A(c_p), A(c_r)\}$$

$$\text{dla } i = 1, 2, \dots, t-1$$

będziemy nazywać pseudołańcuchem i oznaczać $P(x_{k_1}, x_{k_t}; K, N)$. Przyjmujemy, że dla dowolnego obiektu $x \in X$ istnieje pseudołańcuch $P(x, x; K, N)$, podobnie jak przyjęliśmy, że dla dowolnego $x \in X$ istnieje $L(x, x)$ o długości 0. Oznaczmy

$$X_U = \{x \in X: \bigvee_{x_i \in X_D} \bigvee_{x_j \in X_V} \text{istnieją } L(x_i, x) \text{ i } L(x, x_j) \text{ o d\kern-1pt\lrcorner} \geq 0\}$$

X_U jest więc zbiorem obiektów związanych z zadaniem optymalizacji. Zbiór ten będziemy nazywali zbiorem fundamentalnym. Od zadania optymalizacji wymagamy aby spełniony był jeszcze jeden warunek

$$\bigwedge_{x_i, x_j \in X_U} \text{istnieje } P(x_i, x_j; X_U, D'') \quad (3.11)$$

Powyższy warunek gwarantuje nam, że dane zadanie optymalizacji nie da się podzielić na kilka mniejszych wzajemnie na siebie nie wpływających zadań optymalizacji, które można rozwiązać osobno /bez wątpienia bardziej efektywnie/. Zespół warunków /3.1/ ÷ /3.11/ będziemy nazywać warunkami prostej struktury.

Wartość funkcji celu oblicza się na podstawie wartości cech ze zbioru V. Możemy to zapisać w następujący sposób

$$g(c_{p_1}, c_{p_2}, \dots, c_{p_{n_V}})$$

gdzie $n_V = \bar{V}$, $c_{p_k} \in V$ dla $k = 1, 2, \dots, n_V$. Z kolei wartości cech ze zbioru V dla danego systemu informacyjnego zależą od wartości cech ze zbioru D

$$c_{p_k} = h_k(c_{r_1}, c_{r_2}, \dots, c_{r_{n_D}}) \quad \text{dla } k = 1, 2, \dots, n_V$$

gdzie $n_D = \bar{D}$, $c_{r_l} \in D$ dla $l = 1, 2, \dots, n_D$. Postać każdej funkcji h_k zależy oczywiście od obiektów i połączeń między nimi, ale dla danego systemu informacyjnego jest ona stała. Funkcja celu Ψ jest więc dana wzorem

$$\Psi(c_{r_1}, c_{r_2}, \dots, c_{r_{n_D}}) = g(h_1(c_{r_1}, c_{r_2}, \dots, c_{r_{n_D}}), \dots, h_{n_V}(c_{r_1}, c_{r_2}, \dots, c_{r_{n_D}})) \quad (3.12)$$

Przykład 3.2: Dla zadania optymalizacji z przykładu 3.1 zbiory cech i elementów decyzyjnych oraz cech i elementów funkcji celu wyglądają następująco

$$D' = \{c_8\}$$

$$D'' = \{(c_{14}, c_{15})\}$$

$$V' = \{c_{18}\}$$

$$V'' = \emptyset$$

Natomiast zbiory wejściowy, początkowy, wyjściowy i końcowy są postaci

$$D = \{c_8, c_{15}\}$$

$$X_D = \{x_2, x_4\}$$

$$V = \{c_{18}\}$$

$$X_V = \{x_4\}$$

Zbiór fundamentalny przedstawia się następująco

$$X_U = \{x_1, x_2, x_4\}$$

łatwo sprawdzić, że warunki /3.1/ ÷ /3.11/ są spełnione. Wartość funkcji celu dla tego zadania jest równa wartości cechy c_{18}

Temat 3.1: Jeżeli $M \in X/\sigma$ i $X_U \cap M \neq \emptyset$ to $M \subset X_U$

Rzeczywiście. Jeżeli $x_i, x_j \in M$ i $x_i \in X_U$ to istnieją drogi $L(x_i, x_j)$ i $L(x_j, x_i)$ oraz

$$\bigvee_{x_k \in X_D} \bigvee_{x_l \in X_V} \text{ istnieją } L(x_k, x_i) \text{ i } L(x_i, x_l) \text{ o } d_k \geq 0$$

a zatem istnieje droga $[x_k, \dots, x_i, \dots, x_j, \dots, x_i, \dots, x_l]$ i w konsekwencji $x_j \in X_U$. $\square$

Na podstawie powyższego lematu rodzinę X/σ możemy podzielić na dwie części

$$\mathcal{E} = \{M \in X/\sigma : M \cap X_U = \emptyset\}$$

$$\mathcal{J} = \{M \in X/\sigma : M \cap X_U \neq \emptyset\}$$

Rodzinę $\mathcal{E}$ będziemy nazywali rodziną zewnętrzną a rodzinę $\mathcal{J}$ rodziną wewnętrzną. Jest oczywiste, że

$$\bigcup_{M \in \mathcal{J}} M = X_U$$

$$\bigcup_{M \in \mathcal{E}} M \cup X_U = X$$

$$\mathcal{J} \cup \mathcal{E} = X/\sigma$$

Tworzymy rodzinę podgrafów

$$\mathcal{G}(\mathcal{E} \cup \{X_U\}) = \{ \langle K, \varrho_K \rangle : K \in \mathcal{E} \cup \{X_U\} \}$$

Oznaczmy

$$X_E = X_U \cup \{A(c_p) : (c_p, c_r) \in D'' \vee (c_r, c_p) \in D''\}$$

X_E jest więc zbiorem obiektów związanych z zadaniem optymalizacji lub z cechami elementów decyzyjnych. Zbiór ten będziemy nazywali zbiorem rozszerzonym. Zdefiniujmy teraz w rodzinie $\mathcal{G}(\mathcal{E} \cup \{X_U\})$ relację $<$

$$(3.13) \quad \langle K, \varrho_K \rangle < \langle N, \varrho_N \rangle \Leftrightarrow \left\{ \begin{array}{l} \text{dla pewnych } x_i \in K \text{ i } x_j \in N \text{ istnieje } L(x_i, x_j) \\ \quad \text{o } d\lambda \geq 0, \text{ gdy } K \neq X_U \\ \\ \text{dla pewnych } x_i \in X_E \text{ i } x_j \in N \text{ istnieje } L(x_i, x_j) \\ \quad \text{o } d\lambda \geq 0, \text{ gdy } K = X_U \\ \\ \text{dla pewnych } x_i \in K, x_j \in X_U, x_k \in X_E, x_l \in N \text{ istnieje} \\ \text{ } L(x_i, x_j) \text{ i } L(x_k, x_l) \text{ o } d\lambda \geq 0, \text{ gdy } K \neq X_U \text{ i} \\ N \neq X_U \text{ i } \bigwedge_{x_i \in K} \bigwedge_{x_l \in N} \text{nie istn. } L(x_i, x_l) \end{array} \right.$$

łatwo sprawdzić, że jest to relacja zwrotna i przechodnia. Jest także antysymetryczna. Dla każdej pary grafów $\langle K, \varrho_K \rangle$ i $\langle N, \varrho_N \rangle$ takich, że $K \neq X_U$ i $N \neq X_U$ nie budzi to wątpliwości. Co będzie jednak gdy jeden z nich będzie równy $\langle X_U, \varrho_{X_U} \rangle$? Przypuśćmy, że dla pewnego grafu $\langle N, \varrho_N \rangle$ takiego, że $N \neq X_U$ zachodzi

$$\langle N, \varrho_N \rangle < \langle X_U, \varrho_{X_U} \rangle \quad \text{ i } \quad \langle X_U, \varrho_{X_U} \rangle < \langle N, \varrho_N \rangle$$

Oznacza to, że

$$\bigvee_{x_i \in N} \bigvee_{x_j \in X_U} \text{ istnieje } L(x_i, x_j)$$

oraz

$$\bigvee_{x_k \in X_E} \bigvee_{x_l \in N} \text{ istnieje } L(x_k, x_l)$$

Ponieważ $N \in \mathcal{E} \subset X/\sigma$ więc istnieje $L(x_l, x_i)$. Na mocy definicji zbioru fundamentalnego X_U istnieje także takie $x_t \in X_V$, że istnieje $L(x_j, x_t)$. A zatem istnieje droga $[x_k, \dots, x_l, \dots, x_i, \dots, x_j, \dots, x_t]$. Jeżeli $x_k \in X_E - X_U \neq \emptyset$ to na podstawie definicji zbiorów początkowego X_D i fundamentalnego X_U każdy obiekt opisujący tę drogę musi należeć do X_U . Jeżeli zaś $x_k \in X_U$ to na mocy definicji X_U istnieje $x_s \in X_D$ takie, że istnieje droga $L(x_s, x_k)$, a więc istnieje droga $[x_s, \dots, x_k, \dots, x_l, \dots, x_j, \dots, x_t]$ i w konsekwencji każdy obiekt opisujący drogę musi należeć do X_U . Obiekty x_i, x_l muszą zatem w obu przypadkach należeć do X_U . Sprzeczność ponieważ $x_i, x_l \in N$ i $N \cap X_U = \emptyset$. Tak więc relacja $<$ jest antysymetryczna. $\square$

Ostatecznie, ponieważ $<$ jest zwrotna, antysymetryczna i przechodnia więc porządkuje rodzinę $\mathcal{G}(\mathcal{E} \cup \{X_U\})$.

Definiujemy teraz rodzinę podgrafów

$$\mathcal{G}(\mathcal{U}) = \{ \langle K, \varrho_K \rangle : K \in \mathcal{U} \}$$

Mamy $\mathcal{G}(\mathcal{U}) \subset \mathcal{G}(X/\sigma)$. Przypomnijmy, że rodzina $\mathcal{G}(X/\sigma)$ jest uporządkowana przez relację $\leq$. Oznaczmy przez $\leq_{\mathcal{G}(\mathcal{U})}$ relację $\leq$ ograniczoną do rodziny $\mathcal{G}(\mathcal{U})$. Relacja $\leq_{\mathcal{G}(\mathcal{U})}$ porządkuje rodzinę $\mathcal{G}(\mathcal{U})$. Zdefiniujemy teraz dla elementów rodziny $\mathcal{G}(X/\sigma)$ relację $<$ w następujący sposób

$$\langle K, \varrho_K \rangle < \langle N, \varrho_N \rangle \Leftrightarrow \begin{cases} \langle K, \varrho_K \rangle < \langle N, \varrho_N \rangle & \text{gdy } K, N \in \mathcal{E} \\ \langle K, \varrho_K \rangle < \langle X_U, \varrho_{X_U} \rangle & \text{gdy } K \in \mathcal{E} \text{ i } N \in \mathcal{U} \\ \langle X_U, \varrho_{X_U} \rangle < \langle N, \varrho_N \rangle & \text{gdy } K \in \mathcal{U} \text{ i } N \in \mathcal{E} \\ \langle K, \varrho_K \rangle \leq_{\mathcal{G}(\mathcal{U})} \langle N, \varrho_N \rangle & \text{gdy } K, N \in \mathcal{U} \end{cases}$$

Nietrudno wykazać, że relacja $<$ porządkuje rodzinę $\mathcal{G}(X/\sigma)$. Należy pamiętać, że relacja ta została określona dla ustalonego zadania optymalizacji i w związku z tym dla innego zadania przy tym samym niezmiennym systemie i odpowiadającym mu systemie informacyjnym uporządkowanie $\mathcal{G}(X/\sigma)$ będzie z reguły inne.

Relacja ν określona dla elementów zbioru X wzorem

$$x_i \nu x_j \Leftrightarrow \text{istnieje } P(x_i, x_j; X, D'') \quad (3.14)$$

jest relacją równoważności i ustala nam podział zbioru X na klasy równoważności. Oznaczmy

$$\mathcal{G}(X/\nu) = \{ \langle N, \varrho_N \rangle : N \in X/\nu \}$$

Zauważmy, że podsystemy informacyjne odpowiadające podgrafom z rodziny $\mathcal{G}(X/\nu)$ mogą być obliczane niezależnie od siebie

/nie występują między nimi przepływy informacji/, jeżeli tylko dla żadnego elementu (c_p, c_r) ze zbioru $B^{-1}((1,1))$ nie przyjmujemy wartości cechy c_p różnej od wartości cechy c_r .

Jeżeli $X/\nu = \{X\}$ to akceptujemy bez zastrzeżeń uporządkowanie rodziny $\mathcal{G}(X/\sigma)$ przez relację $<$. Jest ono zgodne z kierunkami przepływu informacji pomiędzy σ -podsystemami informacyjnymi, a ponadto uwzględnia specyficzne wymogi zadania optymalizacji. Zatem σ -podsystemy informacyjne możemy obliczać w kolejności wyznaczonej uporządkowaniem odpowiadających im podgrafów z $\mathcal{G}(X/\sigma)$ z tym, że dla grupy σ -podsystemów informacyjnych odpowiadających podgrafom z $\mathcal{G}(\mathcal{U})$ trzeba powtarzać obliczenia cyklicznie tak długo, aż znajdziemy poszukiwane ekstremum funkcji celu.

Jeżeli natomiast $X/\nu \neq \{X\}$ to dla każdego $N \in X/\nu$ mamy $\mathcal{G}(N/\sigma) \subset \mathcal{G}(X/\sigma)$. Rzeczywiście. Niech $x_i \in N$ i $x_j \in X - N$. Mamy $\sim x_i \nu x_j$ a więc $\sim x_i \sigma x_j$. Zatem $N/\sigma \subset X/\sigma$ i w konsekwencji $\mathcal{G}(N/\sigma) \subset \mathcal{G}(X/\sigma)$. Każdą rodzinę $\mathcal{G}(N/\sigma)$ gdzie $N \in X/\nu$ i $N \cap X_U = \emptyset$ porządkuje nam relacja $\leq_{\mathcal{G}(N/\sigma)}$ będąca relacją $\leq$ ograniczoną do rodziny $\mathcal{G}(N/\sigma)$. Natomiast rodzinę $\mathcal{G}(N/\sigma)$ gdzie $N \in X/\sigma$ i $N \cap X_U \neq \emptyset$ porządkuje nam relacja $<_{\mathcal{G}(N/\sigma)}$ która jest relacją $<$ ograniczoną do rodziny

$\mathcal{G}(N/\sigma)$. Uporządkowania te są zgodne z kierunkami przepływu informacji pomiędzy σ -podsystemami informacyjnymi. Zatem σ -podsystemy informacyjne odpowiadające podgrafom z rodziny $\mathcal{G}(N/\sigma)$ obliczamy wg kolejności wyznaczonej uporządkowaniem tej rodziny przez relację $\leq_{\mathcal{G}(N/\sigma)}$ gdy $N \cap X_U = \emptyset$ albo przez relację $<_{\mathcal{G}(N/\sigma)}$ gdy $N \cap X_U \neq \emptyset$. W tym ostatnim przypadku dla grupy σ -podsystemów informacyjnych odpowiadających podgrafom z $\mathcal{G}(\mathcal{U})$ trzeba przeprowadzać obliczenia cyklicznie, aż do uzyskania poszukiwanego ekstremum funkcji celu. Zauważmy, że nie ma żadnego przepływu informacji pomiędzy dwoma σ -podsystemami informacyjnymi odpowiadającymi podgrafom z dwu różnych rodzin $\mathcal{G}(K/\sigma)$ i $\mathcal{G}(N/\sigma)$

Dzięki temu każdą rodzinę σ -podsystemów informacyjnych odpowiadających podgrafom z $\mathcal{G}(N/\sigma)$ dla ustalonego $N \in X/v$ możemy obliczać niezależnie od pozostałych σ -podsystemów informacyjnych.

Uwaga: Ponieważ dla pewnych $K \in \mathcal{E}$ i wszystkich $M \in \mathcal{M}$ może zachodzić

$$\langle K, \varrho_K \rangle \not\leq \langle M, \varrho_M \rangle \quad \text{ i } \quad \langle M, \varrho_M \rangle \not\leq \langle K, \varrho_K \rangle$$

więc zarówno w przypadku $X/v = \{X\}$ jak i $X/v \neq \{X\}$ kolejność obliczeń musi spełniać warunek: pomiędzy σ -podsystemami informacyjnymi odpowiadającymi podgrafom z $\mathcal{G}(\mathcal{M})$ nie występują żadne σ -podsystemy informacyjne odpowiadające podgrafom z $\mathcal{G}(\mathcal{E})$.

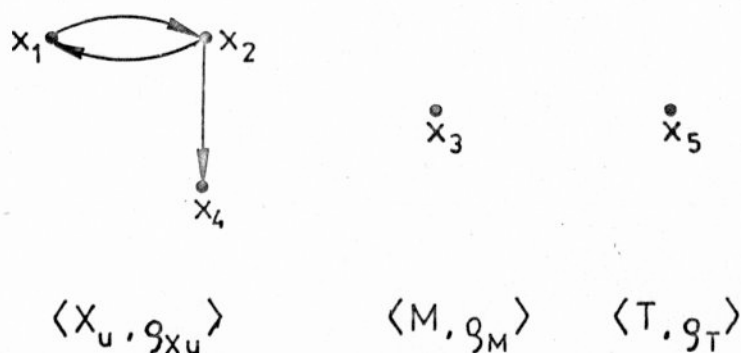
Przykład 3.3: Dla zadania optymalizacji z przykładu 3.1 /patrz także przykład 3.2/ rodziny zewnętrzna i wewnętrzna oraz zbiór rozszerzony wyglądają następująco

$$\mathcal{E} = \{M, T\} = \{\{x_3\}, \{x_5\}\}$$

$$\mathcal{M} = \{K, N\} = \{\{x_1, x_2\}, \{x_4\}\}$$

$$X_E = \{x_1, x_2, x_3, x_4\}$$

Rodzina $\mathcal{G}(\mathcal{E} \cup \{X_U\})$ jest przedstawiona na rys. II.3.5.



Rys. II. 3.5

Relacja $<$ ustala wśród podgrafów tej rodziny poniższe zależności

$$\langle x_U, e_{x_U} \rangle < \langle M, e_M \rangle \quad i \quad \langle T, e_T \rangle < \langle x_U, e_{x_U} \rangle$$

Rodzina $\mathcal{G}(\mathcal{Y}) = \{ \langle K, e_K \rangle, \langle N, e_N \rangle \}$ jest uporządkowana przez relację $\leq_{\mathcal{G}(\mathcal{Y})}$ w następujący sposób

$$\langle K, e_K \rangle \leq_{\mathcal{G}(\mathcal{Y})} \langle N, e_N \rangle$$

Ostatecznie relacja $<$ ustala podane niżej uporządkowanie rodziny $\mathcal{G}(X/\sigma)$.

$$\langle T, e_T \rangle < \langle K, e_K \rangle < \langle N, e_N \rangle < \langle M, e_M \rangle$$

Ponieważ $X/\sigma = \{X\}$ więc wszystkie σ -podsystemy informacyjne odpowiadające podgrafom z $\mathcal{G}(X/\sigma)$ trzeba liczyć razem. Jedyna możliwa /nie zawsze tak musi być/ kolejność ich obliczenia jest następująca

1. σ -podsystem informacyjny o strukturze $\langle T, e_T \rangle$
2. σ -podsystem informacyjny o strukturze $\langle N, e_N \rangle$
3. σ -podsystem informacyjny o strukturze $\langle K, e_K \rangle$
4. σ -podsystem informacyjny o strukturze $\langle M, e_M \rangle$

Uwaga: Jak długo nie osiągniemy poszukiwanego ekstremum funkcji celu tak długo musimy po obliczeniu σ -podsystemu informacyjnego z punktu 3 wracać do punktu 2.

4. ANALIZA DWUPOZIOMOWEGO ZADANIA OPTIMALIZACJI SYSTEMU

Niech będą dane system i odpowiadający mu system informacyjny. Będziemy rozważać dwupoziomowe zadanie optymalizacji systemu zawierające n zadań optymalizacji /niehierarchicznych/ dolnego poziomu. Zakładamy, że każde z zadań lokalnych / dolnego poziomu/ spełnia warunki prostej struktury, przy czym dla k -tego zadania optymalizacji / $k = 1, 2, \dots, n$ / zbiory D'_k i D''_k są odpowiednio zbiorami cech i elementów decyzyjnych, zbiory V'_k i V''_k - zbiorami cech i elementów funkcji celu /zadania lokalnego/, X_U^k - zbiorem fundamentalnym, a $X_E^{k\wedge}$ - zbiorem rozszerzonym. Żądamy aby dla dwóch dowolnych ale różnych lokalnych zadań optymalizacji zachodziło

$$D'_k \cap D'_1 = \emptyset \quad \text{dla } k \neq 1 \quad (4.1)$$

$$D''_k \cap D''_1 = \emptyset \quad \text{dla } k \neq 1 \quad (4.2)$$

$$V'_k \cap V'_1 = \emptyset \quad \text{dla } k \neq 1 \quad (4.3)$$

$$V''_k \cap V''_1 = \emptyset \quad \text{dla } k \neq 1 \quad (4.4)$$

a także

$$\bigwedge_{x_i \in X_E^k} \bigwedge_{\substack{x_j \in X_U^1 \\ k \neq 1}} \text{nie istnieje } L(x_i, x_j) \text{ o } d_L \geq 0 \quad (4.5)$$

Niech D' i D'' będą zbiorami cech i elementów koordynacyjnych /tj. zbiorami cech i elementów decyzyjnych zadania górnego poziomu/ spełniającymi odpowiednio warunki /3.1/ i /3.2/ a także warunek /3.5/. Wymagamy również aby

$$D' \cap D'_k = \emptyset \quad \text{dla } k = 1, 2, \dots, n \quad (4.6)$$

$$D'' \cap D''_k = \emptyset \quad \text{dla } k = 1, 2, \dots, n \quad (4.7)$$

Za zbiory cech i elementów funkcji celu zadania nadrzędnego /górnego poziomu/ przyjmujemy

$$V' = \bigcup_{k=1}^n V'_k \quad (4.8)$$

$$V'' = \bigcup_{k=1}^n V''_k \quad (4.9)$$

Jest oczywiste, że zbiory te spełniają warunek /3.6/. Udowodnimy teraz, że zbiór wyjściowy V zadania nadrzędnego jest sumą zbiorów wyjściowych V_k wszystkich zadań lokalnych

$$V = \bigcup_{k=1}^n V_k \quad (4.10)$$

Mamy

$$\begin{aligned} V &= V' \cup \{c_p : (c_p, c_r) \in V'' \cap B^{-1}((0, 1)) \vee (c_r, c_p) \in V'' \cap B^{-1}((1, 0))\} = \\ &= V' \cup \{c_p : (c_p, c_r) \in V'' \cap B^{-1}((0, 1))\} \cup \{c_p : (c_r, c_p) \in V'' \cap B^{-1}((1, 0))\} = \\ &= \bigcup_{k=1}^n V'_k \cup \{c_p : (c_p, c_r) \in \bigcup_{k=1}^n V''_k \cap B^{-1}((0, 1))\} \cup \{c_p : (c_r, c_p) \in \bigcup_{k=1}^n V''_k \cap B^{-1}((1, 0))\} = \\ &= \bigcup_{k=1}^n V'_k \cup \bigcup_{k=1}^n \{c_p : (c_p, c_r) \in V''_k \cap B^{-1}((0, 1))\} \cup \bigcup_{k=1}^n \{c_p : (c_r, c_p) \in V''_k \cap B^{-1}((1, 0))\} = \\ &= \bigcup_{k=1}^n (V'_k \cup \{c_p : (c_p, c_r) \in V''_k \cap B^{-1}((0, 1))\} \cup \{c_p : (c_r, c_p) \in V''_k \cap B^{-1}((1, 0))\}) = \\ &= \bigcup_{k=1}^n (V'_k \cup \{c_p : (c_p, c_r) \in V''_k \cap B^{-1}((0, 1)) \vee (c_r, c_p) \in V''_k \cap B^{-1}((1, 0))\}) = \bigcup_{k=1}^n V_k \end{aligned}$$

Zatem wzór /4.10/ jest prawdziwy. $\square$

W konsekwencji zbiór końcowy X_V zadania nadrzędnego jest równy sumie zbiorów końcowych X_V^k wszystkich zadań lokalnych:

$$X_V = \bigcup_{k=1}^n X_V^k \quad (4.11)$$

Rzeczywiście, gdyż

$$X_V = A(V) = A\left(\bigcup_{k=1}^n V_k\right) = \bigcup_{k=1}^n A(V_k) = \bigcup_{k=1}^n X_V^k$$

Wymagamy aby dla zadania nadrzędnego spełnione były również warunki (3.7), (3.8) i (3.11). Natomiast warunki (3.9) i (3.10) nie muszą być spełnione. Oznaczmy przez D zbiór wejściowy zadania nadrzędnego. Ostatni warunek jaki musi spełniać zadanie nadrzędne wygląda następująco

$$\bigwedge_{1 \leq k \leq n} \bigvee_{x_i \in X_D} \bigvee_{x_j \in X_V^k} \text{ istnieje } L(x_i, x_j) \text{ o } d_L \geq 0 \quad (4.12)$$

gdzie X_D jest zbiorem początkowym zadania nadrzędnego.

Niech D_k , $k = 1, 2, \dots, n$ będzie zbiorem wejściowym k -tego zadania lokalnego. Oznaczmy przez $\vec{D}_k$, $k = 1, 2, \dots, n$ wektor składający się z $\vec{D}_k$ elementów zbioru wejściowego D_k ułożonych kolejno wg wzrastających indeksów /zakładamy oczywiście, że elementy zbioru C są indeksowane/. Podobnie niech $\vec{D}$ będzie wektorem odpowiadającym zbiorowi wejściowemu D . Niech Ψ_k dla $k = 1, 2, \dots, n$ będzie funkcją celu k -tego zadania lokalnego. Jej wartość zależy od wartości zmiennych decyzyjnych tego zadania i od wartości zmiennych decyzyjnych zadania nadrzędnego:

$$\Psi_k(c_{p_1}, c_{p_2}, \dots, c_{p_{n_D}}, c_{r_1}, c_{r_2}, \dots, c_{r_{n_k}})$$

gdzie $n_D = \bar{n}$, $c_{p_i} \in D$ dla $i = 1, 2, \dots, n_D$, a $n_k = \bar{n}_k$, $c_{r_j} \in D_k$ dla $j = 1, 2, \dots, n_k$. W zapisie wektorowym wygląda to następująco

$$\Psi_k(\vec{D}, \vec{D}_k)$$

Funkcje Ψ_k można oczywiście wyrazić podobnie jak funkcję Ψ we wzorze /3.12/. Nie będziemy jednak tego robić.

Zauważmy, że zbiorami cech i elementów decyzyjnych naszego zadania dwupoziomowego są odpowiednio

$$\tilde{D}' = D' \cup \bigcup_{k=1}^n D'_k \quad (4.13)$$

$$\tilde{D}'' = D'' \cup \bigcup_{k=1}^n D''_k \quad (4.14)$$

a zbiorami cech i elementów funkcji celu V' i V'' określone wzorami /4.8/ i /4.9/. Analogicznie jak wzory /4.10/ i /4.11/ można udowodnić, że zbiory wejściowy $\tilde{D}$ i początkowy $\tilde{X}_D$ zadania dwupoziomowego wyrażają się następująco

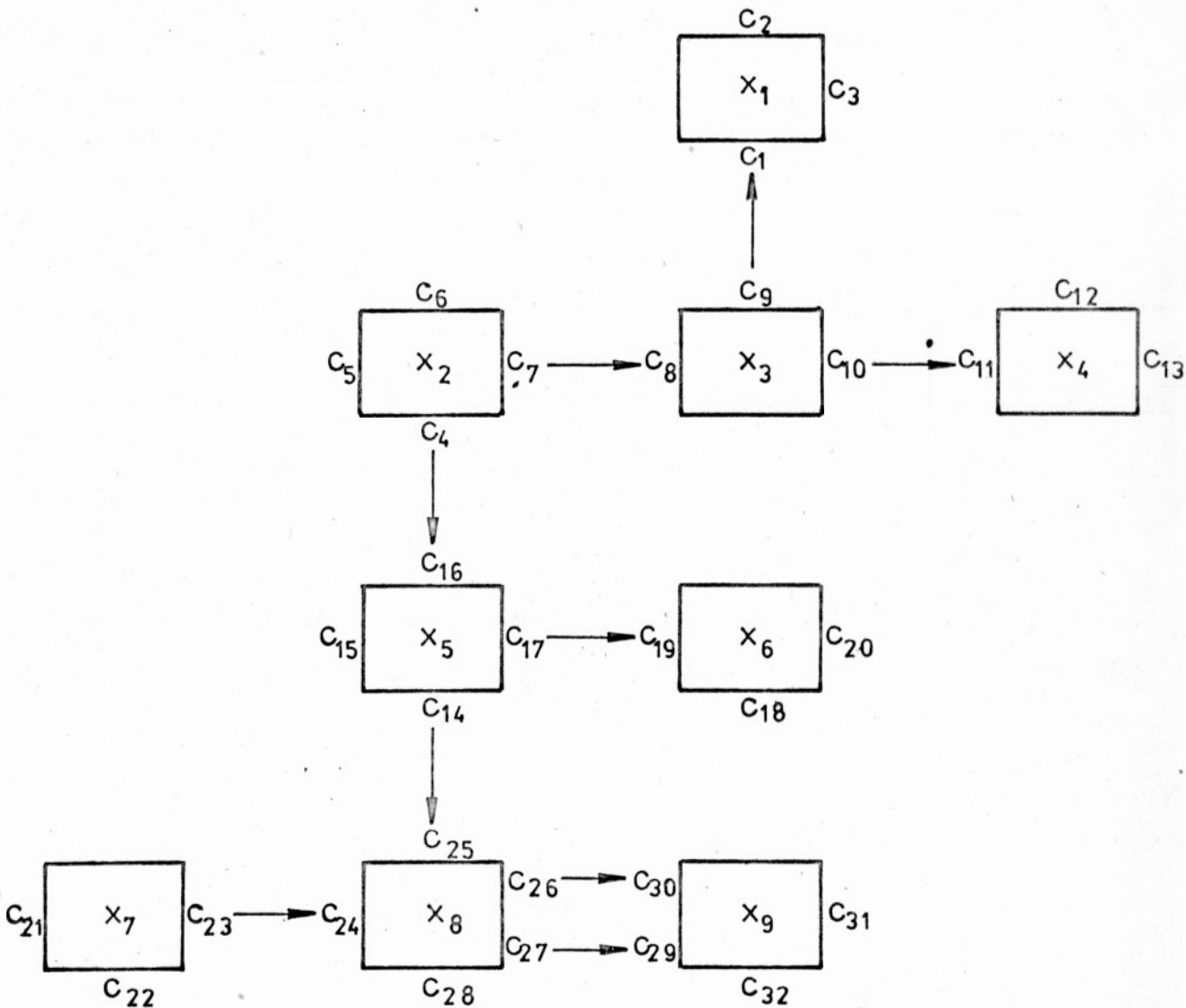
$$\tilde{D} = D \cup \bigcup_{k=1}^n D_k \quad (4.15)$$

$$\tilde{X}_D = X_D \cup \bigcup_{k=1}^n X_D^k \quad (4.16)$$

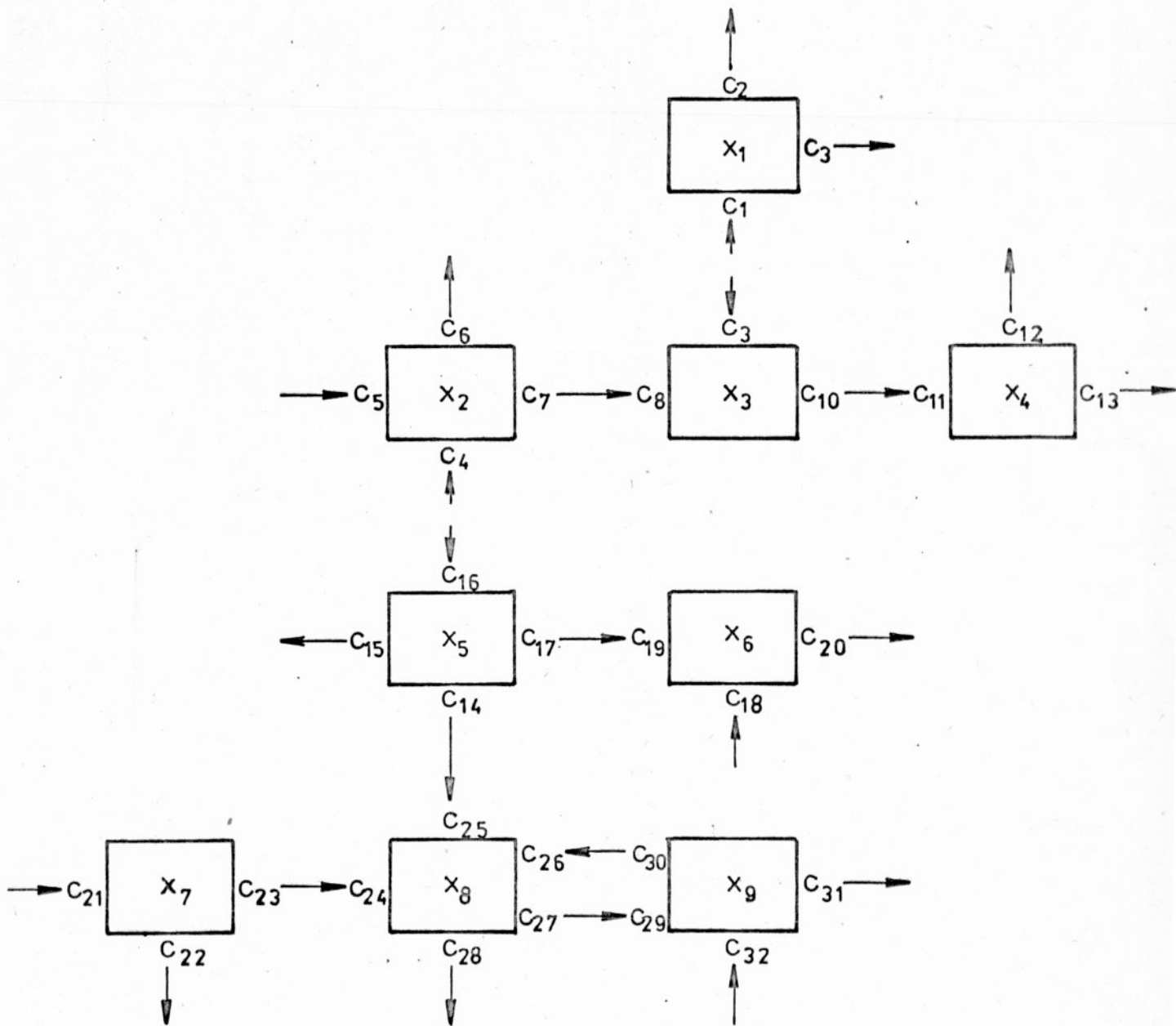
gdzie X_D^k dla $k = 1, 2, \dots, n$ jest zbiorem początkowym k -tego zadania lokalnego. Zbiory wyjściowy V i końcowy X_V zadania dwupoziomowego są oczywiście takie same jak dla zadania nadrzędnego i opisane wzorami /4.10/ i /4.11/. Funkcja celu zadania dwupoziomowego przedstawia się wzorem

$$\Psi(\vec{d}, \vec{d}_1, \vec{d}_2, \dots, \vec{d}_n) = f(\Psi_1(\vec{d}, \vec{d}_1), \Psi_2(\vec{d}, \vec{d}_2), \dots, \Psi_n(\vec{d}, \vec{d}_n))$$

Przykład 4.1: Niech system i odpowiadający mu system informacyjny będą przedstawione na rys. II.4.1 i II.4.2 odpowiednio.



Rys. II.4.1



Rys. II. 4.2

Niech dwupoziomowe zadanie optymalizacji zawiera trzy zadania lokalne dla których zbiory cech i elementów decyzyjnych oraz cech i elementów funkcji celu będą określone następująco

$$\begin{array}{llll}
 D_1' = \emptyset & D_1'' = \{(c_9, c_1)\} & V_1' = \{c_{13}\} & V_1'' = \emptyset \\
 D_2' = \{c_{18}\} & D_2'' = \emptyset & V_2' = \{c_{20}\} & V_2'' = \emptyset \\
 D_3' = \{c_{32}\} & D_3'' = \emptyset & V_3' = \emptyset & V_3'' = \{(c_{26}, c_{30})\}
 \end{array}$$

Zbiory wejściowe, początkowe, wyjściowe i końcowe zadań lokalnych podajemy niżej:

$$\begin{array}{llll}
 D_1 = \{c_9\} & x_D^1 = \{x_3\} & V_1 = \{c_{13}\} & x_V^1 = \{x_4\} \\
 D_2 = \{c_{18}\} & x_D^2 = \{x_6\} & V_2 = \{c_{20}\} & x_V^2 = \{x_6\} \\
 D_3 = \{c_{32}\} & x_D^3 = \{x_9\} & V_3 = \{c_{30}\} & x_V^3 = \{x_9\}
 \end{array}$$

Natomiast zbiory fundamentalne i rozszerzone przedstawiają się następująco

$$\begin{array}{ll}
 x_U^1 = \{x_3, x_4\} & x_E^1 = \{x_1, x_3, x_4\} \\
 x_U^2 = \{x_6\} & x_E^2 = \{x_6\} \\
 x_U^3 = \{x_8, x_9\} & x_E^3 = \{x_8, x_9\}
 \end{array}$$

Nietrudno sprawdzić, że zadania lokalne spełniają warunki prostej struktury oraz warunki /4.1/ ÷ /4.7/. Niech zbiorami cech i elementów koordynacyjnych /tzn. zbiorami cech i elementów decyzyjnych zadania nadrzędnego/ będą

$$D' = \emptyset \quad D'' = \{(c_4, c_{16})\}$$

Zbiory cech i elementów funkcji celu zadania nadrzędnego określone wzorami /4.8/, /4.9/ wyglądają następująco

$$V' = \{ c_{13}, c_{20} \} \quad V'' = \{ c_{26}, c_{30} \}$$

a zbiorami wyjściowym i końcowym /patrz wzory /4.10/ i /4.11// są

$$V = \{ c_{13}, c_{20}, c_{30} \} \quad X_V = \{ x_4, x_6, x_9 \}$$

Zbiory wejściowy, początkowy, fundamentalny i rozszerzony mają następującą postać

$$D = \{ c_4, c_{16} \} \quad X_D = \{ x_2, x_5 \}$$

$$X_U = \{ x_2, x_3, x_4, x_5, x_6, x_8, x_9 \} \quad X_E = \{ x_2, x_3, x_4, x_5, x_6, x_8, x_9 \}$$

Wartości funkcji celu zadań lokalnych oblicza się na podstawie wartości cech z odpowiednich zbiorów wyjściowych za pomocą zadanych przez nas funkcji

$$g_1(c_{13}) \quad g_2(c_{20}) \quad g_3(c_{30})$$

Ponieważ

$$c_{13} = h_1(c_4, c_9)$$

$$c_{20} = h_2(c_{16}, c_{18})$$

$$c_{30} = h_3(c_{16}, c_{32})$$

więc funkcje celu zadań lokalnych wyglądają następująco

$$\Psi_1(c_4, c_9) = g_1(h_1(c_4, c_9))$$

$$\Psi_2(c_{16}, c_{18}) = g_2(h_2(c_{16}, c_{18}))$$

$$\Psi_3(c_{16}, c_{32}) = g_3(h_3(c_{16}, c_{32}))$$

przy czym wartość cechy c_4 jest zawsze równa wartości cechy c_{16} .

Na ogół nie znamy tych funkcji celu, gdyż nie znamy postaci

funkcji h_k zależnych od obiektów i połączeń między nimi. Ze wzorów /4.13/ i /4.14/ otrzymujemy zbiory cech i elementów decyzyjnych zadania dwupoziomowego

$$\tilde{D}' = \{c_{18}, c_{32}\} \quad \tilde{D}'' = \{\wedge c_4, c_{16}\wedge, \wedge c_9, c_1\wedge\}$$

Zbiory cech i elementów funkcji celu zadania dwupoziomowego są takie same jak dla zadania nadrzędnego. Zbiory wejściowy i początkowy zadania dwupoziomowego otrzymujemy wykorzystując wzory /4.15/ i /4.16/. Przedstawiają się one następująco

$$\tilde{D} = \{c_4, c_9, c_{16}, c_{18}, c_{32}\} \quad \tilde{X}_D = \{x_2, x_3, x_5, x_6, x_9\}$$

Zbiory wyjściowy i końcowy zadania dwupoziomowego są identyczne jak dla zadania nadrzędnego. Funkcji celu zadania dwupoziomowego o postaci ogólnej

$$\Psi(c_4, c_{16}, c_9, c_{18}, c_{32}) = f(\Psi_1(c_4, c_9), \Psi_2(c_{16}, c_{18}), \Psi_3(c_{16}, c_{32}))$$

nie możemy z reguły wyrazić w postaci jawnej wzorami mimo, że znamy funkcję f /jest ona zadana przez nas/. Możemy natomiast łatwo obliczyć jej wartość posługując się niżej podanym wyrażeniem

$$f(g_1(c_{13}), g_2(c_{20}), g_3(c_{30}))$$

Na tym chwilowo zakończymy rozważanie naszego przykładu.

Zauważmy, że zadanie dwupoziomowe spełnia warunki /3.1/ ÷ /3.10/. Wynika to z tego, że poszczególne zadania lokalne spełniają te warunki, a zadanie nadrzędne czyni zadość warunkom /3.1/ ÷ /3.8/ oraz ze wzorów /4.10/, /4.11/, /4.15/ i /4.16/. Powstaje pytanie czy zadanie dwupoziomowe spełnia również warunek /3.11/

i w konsekwencji spełnienia warunki prostej struktury. Wykażemy najpierw, że zbiór fundamentalny X_U zadania dwupoziomowego jest równy sumie zbiorów fundamentalnych: X_U^k wszystkich zadań lokalnych i X_U zadania nadrzędnego tj.

$$\tilde{X}_U = X_U \cup \bigcup_{k=1}^n X_U^k \quad (4.17)$$

Istotnie

$$\begin{aligned} \tilde{X}_U &= \{x \in X: \bigvee_{x_i \in \tilde{X}_D} \bigvee_{x_j \in X_V} \text{istnieją } L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0\} = \quad (4.16) \\ &= \{x \in X: (\bigvee_{x_i \in X_D} \bigvee_{x_j \in X_V} \text{istn. } L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0) \vee \\ &(\bigvee_{x_i \in \bigcup_{k=1}^n X_D^k} \bigvee_{x_j \in X_V} \text{istn. } L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0)\} = \{x \in X: \bigvee_{x_i \in X_D} \bigvee_{x_j \in X_V} \\ &\text{istn. } L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0\} \cup \{x \in X: \bigvee_{x_i \in \bigcup_{k=1}^n X_D^k} \bigvee_{x_j \in X_V} \text{istn.} \\ &L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0\} = X_U \cup \bigcup_{k=1}^n \{x \in X: \bigvee_{x_i \in X_D^k} \bigvee_{x_j \in \bigcup_{l=1}^n X_V^l} \text{istnieją} \\ &L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0\} = X_U \cup \bigcup_{k=1}^n \{x \in X: \bigvee_{x_i \in X_D^k} \bigvee_{x_j \in X_V^k} \text{istnieją} \\ &L(x_i, x) \text{ i } L(x, x_j) \text{ o } d\lambda \geq 0\} = X_U \cup \bigcup_{k=1}^n X_U^k \end{aligned}$$

Tak więc udowodniliśmy wzór /4.17/. Zauważmy, że

Jeżeli $x_i, x_j \in X_U^k$ to istnieje $P(x_i, x_j; X_U^k, D_k'')$ a zatem na podstawie wzorów /4.14/ i /4.17/ istnieje $P(x_i, x_j; \tilde{X}_U, \tilde{D}'')$.

Jeżeli $x_i, x_j \in X_U$ to istnieje $P(x_i, x_j; X_U, D'')$, a zatem istnieje $P(x_i, x_j; \tilde{X}_U, \tilde{D}'')$. Jeżeli $x_i \in X_U$ a $x_j \in X_U - X_U^k$ to z /4.12/ mamy, że

$$\bigvee_{x_s \in X_D} \bigvee_{x_t \in X_V^k} \text{ istnieje } L(x_s, x_t) \text{ o } d_L \geq 0$$

Istnieją także $P(x_i, x_s; X_U, D'')$ i $P(x_t, x_j; X_U^k, D_k'')$. W konsekwencji istnieje więc $P(x_i, x_j; \tilde{X}_U, \tilde{D}'')$.

Jeżeli $x_i \in X_U^k$ a $x_j \in X_U^1$ gdzie $k \neq 1$ to na podstawie powyższych wywodów jest oczywiste, że $P(x_i, x_j; \tilde{X}_U, \tilde{D}'')$.

Biorąc pod uwagę wszystkie omówione wyżej przypadki szczególne dochodzimy do wniosku, że

$$\bigwedge_{x_i, x_j \in \tilde{X}_U} \text{ istnieje } P(x_i, x_j; \tilde{X}_U, \tilde{D}'')$$

czyli, że dla zadania dwupoziomowego spełniony jest warunek /3.11/.
Zatem zadanie dwupoziomowe spełnia warunki prostej struktury. $\square$

Uwaga: Nie każde jednak zadanie jednopoziomowe spełniające warunki prostej struktury da się rozpatrywać jako zadanie dwupoziomowe.

Oznaczmy zbiór rozszerzony zadania nadrzędnego przez X_E a zbiór rozszerzony zadania dwupoziomowego przez $X_{\tilde{E}}$. Łatwo można wykazać wychodząc z definicji zbioru rozszerzonego, że

$$\tilde{X}_E = X_E \cup \bigcup_{k=1}^n X_E^k \quad /4.18/$$

Niech $\tilde{\mathcal{E}}$ będzie rodziną zewnętrzną a $\tilde{\mathcal{I}}$ rodziną wewnętrzną zadania dwupoziomowego. Oznaczmy

$$\mathcal{G}(\tilde{\mathcal{E}} \cup \{X_U\}) = \{ \langle N, \mathcal{E}_N \rangle : N \in \tilde{\mathcal{E}} \cup \{X_U\} \}$$

Ponieważ zadanie dwupoziomowe spełnia warunki prostej struktury więc w $\mathcal{G}(\tilde{\mathcal{E}} \cup \{X_U\})$ można zdefiniować relację $<$ wzorem /3.13/ przyjmując tylko $\tilde{X}_U, \tilde{X}_E, \tilde{\mathcal{E}}, \tilde{\mathcal{I}}$ zamiast $X_U, X_E, \mathcal{E}, \mathcal{I}$ odpowiednio. Niech

$\mathfrak{U}_k$ będzie rodziną wewnętrzną dla k -tego zadania lokalnego,
 $k = 1, 2, \dots, n$. Oznaczmy

$$\mathcal{G}(\left(\tilde{\mathfrak{U}} - \bigcup_{k=1}^n \mathfrak{U}_k\right) \cup \bigcup_{k=1}^n \{x_U^k\}) = \{\langle N, \varrho_N \rangle : N \in (\tilde{\mathfrak{U}} - \bigcup_{k=1}^n \mathfrak{U}_k) \cup \bigcup_{k=1}^n \{x_U^k\}\}$$

Dla podgrafów z rodziny $\mathcal{G}(\left(\tilde{\mathfrak{U}} - \bigcup_{k=1}^n \mathfrak{U}_k\right) \cup \bigcup_{k=1}^n \{x_U^k\})$ definiujemy relację $<$ w następujący sposób

$$\langle K, \varrho_K \rangle < \langle N, \varrho_N \rangle \Leftrightarrow \text{dla pewnych } x_i \in K \text{ i } x_j \in N \text{ istnieje } L(x_i, x_j) \text{ o } d_L \geq 0$$

Zauważmy, że ze względu na warunek /4.5/ i definicję zbioru fundamentalnego mamy

$$\langle x_U^k, \varrho_{x_U^k} \rangle \not< \langle x_U^1, \varrho_{x_U^1} \rangle \text{ dla } k \neq 1$$

oraz

$$\langle x_U^k, \varrho_{x_U^k} \rangle \not< \langle N, \varrho_N \rangle \text{ dla } N \in \tilde{\mathfrak{U}} - \bigcup_{k=1}^n \mathfrak{U}_k \text{ i } 1 \leq k \leq n$$

łatwo więc wykazać, że relacja $<$ porządkuje $\mathcal{G}(\left(\tilde{\mathfrak{U}} - \bigcup_{k=1}^n \mathfrak{U}_k\right) \cup \bigcup_{k=1}^n \{x_U^k\})$.

Oznaczmy przez $\leq_{\mathcal{G}(\mathfrak{U}_k)}$ relację $\leq$ ograniczoną do rodziny $\mathcal{G}(\mathfrak{U}_k)$,

$k = 1, 2, \dots, n$ /przypomnijmy, że $\leq$ porządkuje $\mathcal{G}(X/\sigma)$ /. Zdefiniujemy

teraz dla podgrafów z rodziny $\mathcal{G}(X/\sigma)$ relację $<^2$ w następujący sposób

$$\langle K, \varrho_K \rangle <^2 \langle N, \varrho_N \rangle \Leftrightarrow \begin{cases} \langle K, \varrho_K \rangle < \langle N, \varrho_N \rangle & \text{gdy } K, N \in \tilde{\mathfrak{E}} \\ \langle K, \varrho_K \rangle < \langle x_U, \varrho_{x_U} \rangle & \text{gdy } K \in \tilde{\mathfrak{E}}, N \in \tilde{\mathfrak{U}} \\ \langle x_U, \varrho_{x_U} \rangle < \langle N, \varrho_N \rangle & \text{gdy } K \in \tilde{\mathfrak{U}}, N \in \tilde{\mathfrak{E}} \\ \langle K, \varrho_K \rangle < \langle N, \varrho_N \rangle & \text{gdy } K, N \in \tilde{\mathfrak{U}} - \bigcup_{k=1}^n \mathfrak{U}_k \\ \langle K, \varrho_K \rangle < \langle x_U^k, \varrho_{x_U^k} \rangle & \text{gdy } K \in \tilde{\mathfrak{U}} - \bigcup_{l=1}^n \mathfrak{U}_l, N \in \mathfrak{U}_k, \\ & 1 \leq k \leq n \\ \langle K, \varrho_K \rangle \leq_{\mathcal{G}(\mathfrak{U}_k)} \langle N, \varrho_N \rangle & \text{gdy } K, N \in \mathfrak{U}_k, 1 \leq k \leq n \end{cases}$$

Nietrudno jest wykazać, że relacja $<^2$ porządkuje rodzinę $\mathcal{G}(X/\sigma)$. Dla elementów zbioru X określamy wzorem /3.14/ relację ν przyjmując tylko $\tilde{D}''$ zamiast D'' . Jest ona relacją równoważności, a ponadto dla każdego $N \in X/\nu$ mamy, że $\mathcal{G}(N/\sigma) \subset \mathcal{G}(X/\sigma)$.

Jeżeli $X/\nu = \{X\}$ to akceptujemy bez zastrzeżeń uporządkowanie rodziny $\mathcal{G}(X/\sigma)$ przez relację $<^2$. Jest ono zgodne z kierunkami przepływu informacji pomiędzy σ -podsystemami informacyjnymi. Zatem σ -podsystemy informacyjne możemy obliczać w kolejności wyznaczonej uporządkowaniem odpowiadających im podgrafów z $\mathcal{G}(X/\sigma)$ z tym, że dla grup σ -podsystemów informacyjnych odpowiadających podgrafom z $\mathcal{G}(\mathcal{U})$ oraz $\mathcal{G}(\mathcal{U}_k)$ dla $k = 1, 2, \dots, n$ trzeba powtarzać obliczenia cyklicznie tak długo, aż znajdziemy poszukiwane ekstrema odpowiednich frunkcji celu.

Jeżeli natomiast $X/\nu \neq \{X\}$ to każdą rodzinę $\mathcal{G}(N/\sigma)$ gdzie $N \in X/\nu$ i $N \cap \tilde{X}_U = \emptyset$ porządkuje nam relacja $\leq_{\mathcal{G}(N/\sigma)}$ będąca relacją $\leq$ ograniczoną do rodziny $\mathcal{G}(N/\sigma)$. Natomiast rodzinę $\mathcal{G}(N/\sigma)$ taką, że $N \in X/\nu$ i $N \cap \tilde{X}_U \neq \emptyset$ porządkuje nam relację $<^2_{\mathcal{G}(N/\sigma)}$, która jest relacją $<^2$ ograniczoną do rodziny $\mathcal{G}(N/\sigma)$.

Uporządkowania te są zgodne z kierunkami przepływu informacji pomiędzy σ -podsystemami informacyjnymi. Zatem σ -podsystemy informacyjne odpowiadające podgrafom z rodziny $\mathcal{G}(N/\sigma)$ obliczamy wg kolejności wyznaczonej uporządkowaniem tej rodziny przez relację $\leq_{\mathcal{G}(N/\sigma)}$ gdy $N \cap \tilde{X}_U = \emptyset$ albo przez relację $<^2_{\mathcal{G}(N/\sigma)}$ gdy $N \cap \tilde{X}_U \neq \emptyset$. W tym ostatnim przypadku dla grup σ -podsystemów informacyjnych odpowiadającym podgrafom z $\mathcal{G}(\mathcal{U})$ oraz $\mathcal{G}(\mathcal{U}_k)$ dla $k = 1, 2, \dots, n$ trzeba przeprowadzać obliczenia cyklicznie aż do uzyskania poszukiwanych ekstremów odpowiednich funkcji celu. Zauważmy, że nie ma żadnego przepływu informacji pomiędzy dwoma

σ - podsystemami informacyjnymi odpowiadającymi podgrafom z dwu różnych rodzin $\mathcal{G}(K/\sigma)$ i $\mathcal{G}(N/\sigma)$ gdzie $K, N \in X/\nu$ i $K \neq N$. Dzięki temu każdą rodzinę σ - podsystemów informacyjnych odpowiadających podgrafom z $\mathcal{G}(N/\sigma)$ dla ustalonego $N \in X/\nu$ możemy obliczać niezależnie od pozostałych σ - podsystemów informacyjnych.

Uwaga: Zarówno w przypadku $X/\nu = \{X\}$ jak i $X/\nu \neq \{X\}$ kolejność obliczeń musi spełniać warunki:

1. pomiędzy σ - podsystemami informacyjnymi odpowiadającymi podgrafom z $\mathcal{G}(\tilde{U})$ nie występują żadne σ - podsystemy informacyjne odpowiadające podgrafom z $\mathcal{G}(\tilde{E})$
2. pomiędzy σ - podsystemami informacyjnymi odpowiadającymi podgrafom z $\mathcal{G}(\mathcal{U}_k)$ dla $k = 1, 2, \dots, n$ nie występują żadne σ - podsystemy informacyjne odpowiadające podgrafom z $\mathcal{G}(\mathcal{E}_k)$, gdzie $\mathcal{E}_k$ jest rodziną zewnętrzną k -tego lokalnego zadania optymalizacji

Przykład 4.2: Będziemy rozważać dalej zadanie optymalizacji z przykładu 4.1. Zbiory fundamentalny i rozszerzony tego zadania mają następującą postać

$$\tilde{X}_U = \{x_2, x_3, x_4, x_5, x_6, x_8, x_9\} \quad \tilde{X}_E = \{x_1, x_2, x_3, x_4, x_5, x_6, x_8, x_9\}$$

W naszym przykładzie zbiory fundamentalne zadań dwupoziomowego i nadrzędnego są takie same, ale nie zawsze tak jest. Rodzinami zewnętrzną i wewnętrzną zadania dwupoziomowego są

$$\tilde{\mathcal{E}} = \{K_1, K_7\} \quad \tilde{\mathcal{U}} = \{K_2, K_3, K_4, K_5, K_6, K_8\}$$

gdzie $K_1 = \{x_1\}$, $K_2 = \{x_2\}$, $K_3 = \{x_3\}$, $K_4 = \{x_4\}$, $K_5 = \{x_5\}$,

$K_6 = \{x_6\}$, $K_7 = \{x_7\}$, $K_8 = \{x_8, x_9\}$. Rodzina $\mathcal{G}(\tilde{X}_U)$ jest uporządkowana przez relację $<$:

$$\langle K_7, e_{K_7} \rangle < \langle x_U, e_{x_U} \rangle < \langle K_1, e_{K_1} \rangle$$

Rodziny wewnętrzne zadań lokalnych mają postać

$$\mathcal{U}_1 = \{K_3, K_4\} \quad \mathcal{U}_2 = \{K_6\} \quad \mathcal{U}_3 = \{K_8\}$$

Relacja $<$ ustanawia następujący porządek dla elementów rodziny

$$\mathcal{G}((\tilde{\mathcal{U}} = \bigcup_{k=1}^3 \mathcal{U}_k) \cup \bigcup_{k=1}^3 \{x_U^k\}):$$

$$\langle K_2, e_{K_2} \rangle < \langle x_U^1, e_{x_U^1} \rangle$$

$$\langle K_5, e_{K_5} \rangle < \langle x_U^2, e_{x_U^2} \rangle$$

$$\langle K_5, e_{K_5} \rangle < \langle x_U^3, e_{x_U^3} \rangle$$

Relacja $\leq_{\mathcal{G}(\mathcal{U}_1)}$ ustala podane niżej zależności dla podgrafów z rodziny $\mathcal{G}(\mathcal{U}_1)$

$$\langle K_3, e_{K_3} \rangle \leq_{\mathcal{G}(\mathcal{U}_1)} \langle K_4, e_{K_4} \rangle$$

Ostatecznie rodzinę $\mathcal{G}(X/\sigma)$ porządkuje relacja $<^2$ w następujący sposób

$$\langle K_7, e_{K_7} \rangle <^2 \left\{ \begin{array}{l} \langle K_2, e_{K_2} \rangle <^2 \langle K_3, e_{K_3} \rangle <^2 \langle K_4, e_{K_4} \rangle \\ \langle K_5, e_{K_5} \rangle <^2 \left\{ \begin{array}{l} \langle K_6, e_{K_6} \rangle \\ \langle K_8, e_{K_8} \rangle \end{array} \right\} \end{array} \right\} <^2 \langle K_1, e_{K_1} \rangle$$

W naszym przypadku $X/v = \{X\}$ więc σ - podsystemy informacyjne obliczamy w kolejności wyznaczonej uporządkowaniem rodziny $\mathcal{G}(X/\sigma)$ przez relację $<^2$. Jedną z możliwych kolejności obliczania σ - podsystemów informacyjnych podajemy niżej:

1. σ - podsystem informacyjny o strukturze $\langle K_7, \mathcal{Q}_{K_7} \rangle$
- 2. σ - podsystem informacyjny o strukturze $\langle K_8, \mathcal{Q}_{K_8} \rangle$
- 3. σ - podsystem informacyjny o strukturze $\langle K_2, \mathcal{Q}_{K_2} \rangle$
- 4. σ - podsystem informacyjny o strukturze $\langle K_4, \mathcal{Q}_{K_4} \rangle$
5. σ - podsystem informacyjny o strukturze $\langle K_3, \mathcal{Q}_{K_3} \rangle$
- ↻ 6. σ - podsystem informacyjny o strukturze $\langle K_5, \mathcal{Q}_{K_5} \rangle$
- ↻ 7. σ - podsystem informacyjny o strukturze $\langle K_6, \mathcal{Q}_{K_6} \rangle$
8. σ - podsystem informacyjny o strukturze $\langle K_1, \mathcal{Q}_{K_1} \rangle$

Po obliczeniu σ - podsystemu informacyjnego z punktu 4 musimy wracać do punktu 3 tak długo aż znajdziemy ekstremum funkcji celu pierwszego zadania lokalnego. Podobnie po obliczeniu σ - podsystemu informacyjnego z punktu 6, jeżeli nie osiągnęliśmy ekstremum funkcji celu drugiego zadania lokalnego to trzeba powtórzyć obliczenia tego σ -podsystemu informacyjnego. Tak samo przedstawia się sprawa z punktem 7. Po znalezieniu ekstremum funkcji celu trzeciego zadania lokalnego w przypadku gdy nie osiągnęliśmy ekstremum funkcji celu zadania dwupoziomowego musimy powrócić do punktu 2.

5. OBLICZANIE σ - PODSYSTEMÓW INFORMACYJNYCH

Niech będzie dany pewien σ - podsystem informacyjny o strukturze opisanej podgrafem $\langle N, \varrho_N \rangle$ gdzie $N \in X/\sigma$. Podgraf $\langle N, \varrho_N \rangle$ możemy w równoważny sposób określić jako parę uporządkowaną $\langle N, U \rangle$ gdzie

$$U = \{[x_i, x_j] : x_i, x_j \in N \wedge x_i \varrho_N x_j\}$$

jest zbiorem łuków podgrafu. Każdemu łukowi $u \in U$ przyporządkowujemy liczbę $n(u) = \frac{\bar{Y}}{\bar{Y}}$ gdzie $\bar{Y}$ jest strumieniem informacyjnym odpowiadającym danemu łukowi / $\bar{Y}$ oznacza liczbę elementów strumienia

czyli jego wymiar/. Określmy zbiór $I \subset U$ tak, że podgraf częściowy $\langle N, U - I \rangle$ jest spójny, bez pętli i nie zawierający konturów. Ponieważ każdy strumień informacyjny wyłączyszy jego kierunek może zostać scharakteryzowany przez $\bar{Y}$ cech / patrz definicja cechy strumienia pośredniego, str 6 / więc oznaczając przez $\vec{c}_{v_1}, \vec{c}_{v_2}, \dots, \vec{c}_{v_t}$ wektory cech strumieni informacyjnych odpowiadających łukom ze zbioru $I = \{u_{v_1}, u_{v_2}, \dots, u_{v_t}\}$ możemy przedstawić dany σ - podsystem informacyjny w postaci następującego układu równań

$$\vec{c}_{v_k} = \vec{\Phi}_{v_k} (\vec{c}_{v_1}, \vec{c}_{v_2}, \dots, \vec{c}_{v_t}) \quad k = 1, 2, \dots, t \quad (5.1)$$

Liczba równań tego układu jest równa $\sum_{k=1}^t n(u_{v_k})$. Ponieważ przyjmuje się, że układ /5.1/ jest tym łatwiej rozwiązać im mniej zawiera on równań /nie zawsze jest to oczywiście prawdą / więc będziemy starali się znaleźć taki podzbiór $I \subset U$ dla którego $\sum_{k=1}^t n(u_{v_k})$ jest minimalna. W literaturze / patrz [12,33,37,41,42,45]/ przedstawiono szereg metod rozwiązania tego problemu. Przedstawimy algorytm opisany w [45].

Niech podgraf $\langle N, U \rangle$ zawiera w konturów prostych. Oznaczmy przez L_i zbiór łuków i -tego konturu prostego $i = 1, 2, \dots, w$, a przez $\mathcal{L}$ rodzinę wszystkich takich zbiorów. Zbiór $I_{\mathcal{B}} \subset U$ będziemy nazywali zbiorem przecinającym dla $\mathcal{B} \subset \mathcal{L}$ gdy

$$\bigwedge_{L \in \mathcal{B}} I_{\mathcal{B}} \cap L \neq \emptyset$$

Niech $I_{\mathcal{B}} = \{u_{r_1}, u_{r_2}, \dots, u_{r_d}\}$. Przez wymiar zbioru $I_{\mathcal{B}}$ rozumiemy liczbę $\dim(I_{\mathcal{B}}) = \sum_{i=1}^d n(u_{r_i})$. Wśród wszystkich zbiorów przecinających dla $\mathcal{B}$ wyróżniamy te, które mają najmniejszy wymiar i nazywamy je optymalnymi zbiorami przecinającymi. Utwórzmy ciąg zbiorów

$$\mathcal{A}_1 = \{L_1\}, \mathcal{A}_2 = \{L_1, L_2\}, \dots, \mathcal{A}_w = \{L_1, L_2, \dots, L_w\} = \mathcal{L}$$

Oznaczmy przez I_k zbiór przecinający dla $\mathcal{A}_k$, $k = 1, 2, \dots, w$. Algorytm buduje kolejno rodziny $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_w$ gdzie $\mathcal{M}_0 = \{\emptyset\}$, $\mathcal{M}_k = \{I_k^1, I_k^2, \dots, I_k^{p_k}\}$ / p_k jest ilością elementów rodziny $\mathcal{M}_k$ /. $\mathcal{M}_k$ zawiera zbiory przecinające dla $\mathcal{A}_k$, a $\mathcal{M}_w$ optymalne zbiory przecinające dla $\mathcal{A}_w$ /dowolny z nich możemy przyjąć za poszukiwany przez zbiór I /. Przejście od $\mathcal{M}_{k-1}$ do $\mathcal{M}_k$ odbywa się według następujących reguł:

- a/ jeżeli $I_{k-1}^i \cap L_k \neq \emptyset$ to I_{k-1}^i zaliczamy do $\mathcal{M}_k$
- b/ jeżeli $I_{k-1}^i \cap L_k = \emptyset$ to wszystkie zbiory postaci $I_{k-1}^i \cup \{u\}$ gdzie $u \in L_k$ zaliczamy do $\mathcal{M}_k$

Niech

$$E(I_k) = \{L \in \mathcal{L} : I_k \cap L \neq \emptyset\}$$

Zauważmy, że $\mathcal{A}_k \subset E(I_k)$. Niech $T(I_k^j, I_k^i)$ oznacza optymalny zbiór

przecinający dla $E(I_k^j) - E(I_k^i)$. Jeżeli

$$\dim(I_k^i) + \dim(T(I_k^j, I_k^i)) < \dim(I_k^j)$$

to algorytm usuwa I_k^j z rodziny M_k . Ma to na celu zapobieżenie zbyt silnemu wzrostowi elementów w rodzinach M_k . Można wykazać, że rodzina M_w zbudowana przez algorytm, którego idea została przedstawiona powyżej zawiera wszystkie optymalne zbiory przecinające dla A_w i nie zawiera poza nimi żadnych innych zbiorów. Dalsze zmniejszenie ilości obliczeń i zapotrzebowania na pamięć komputera uzyskuje się stosując przed uruchomieniem algorytmu następujące reguły eliminacji:

1. Jeżeli $E(\{u_i\}) \subset E(\{u_j\})$ i $n(u_i) > n(u_j)$ to łuk u_i usuwa się ze wszystkich zbiorów $L \in \mathcal{L}$ w których on występuje.
2. Jeżeli zbiór L zawiera tylko jeden łuk u to łuk ten od razu zalicza się do optymalnego zbioru przecinającego dla A_w usuwając jednocześnie z $\mathcal{L}$ wszystkie zbiory z rodziny $E(\{u\})$.
3. Jeżeli $L_i \subset L_j$ to zbiór L_j usuwa się z rodziny $\mathcal{L}$.

Podamy teraz pełny opis algorytmu. Niech l_k oznacza ilość elementów zbioru L_k , a u_e^k e -ty element zbioru L_k , $1 \leq e \leq l_k$.

1. $M_0 = \{\emptyset\}$, $k = 1$, $i = 1$, $e = 1$
2. Obliczamy $I_{k-1}^i \cap L_k$. Jeżeli $I_{k-1}^i \cap L_k \neq \emptyset$ to przechodzimy do punktu 3. Jeżeli $I_{k-1}^i \cap L_k = \emptyset$ to przechodzimy do punktu 4.
3. Przyjmujemy $I = I_{k-1}^i$ i przechodzimy do punktu 5.
4. Przyjmujemy $I = I_{k-1}^i \cup \{u_e^k\}$ i przechodzimy do punktu 5.
5. Jeżeli rodzina M_k jest jeszcze pusta to I zaliczamy do M_k .

Jeżeli nie jest pusta, to obliczamy $M_k \cap \{I\}$. Jeżeli $M_k \cap \{I\} \neq \emptyset$ to przechodzimy do punktu 6. Jeżeli $M_k \cap \{I\} = \emptyset$ to znajdujemy

- $E(I)$ oraz $E(I_k^j)$, gdzie j przebiega wszystkie zbiory, które dotąd zostały zaliczone do M_k . Obliczamy $E(I) - E(I_k^j)$, $T(I, I_k^j)$, $\dim(T(I, I_k^j))$, $\dim(I)$, $\dim(I_k^j)$. Jeżeli dla jakiegoś j spełniona jest nierówność $\dim(T(I, I_k^j)) + \dim(I_k^j) < \dim(I)$, to I odrzucamy. W przeciwnym wypadku I umieszczamy w rodzinie M_k i obliczamy $E(I_k^j) - E(I)$, $T(I_k^j, I)$, $\dim(T(I_k^j, I))$. Wszystkie te zbiory I_k^j dla których spełniona jest nierówność $\dim(T(I_k^j, I)) + \dim(I) < \dim(I_k^j)$ usuwamy z rodziny M_k . Jeżeli punkt 5 wykonywano po punkcie 3, to przechodzimy do punktu 7. Jeżeli punkt 5 wykonywano po punkcie 4 to przechodzimy do punktu 6.
6. Jeżeli $e < l_k$ to przyjmujemy $e = e + 1$ i przechodzimy do punktu 4. W przeciwnym wypadku przechodzimy do punktu 7.
 7. Jeżeli $i < p_{k-1}$ to przyjmujemy $i = i + 1$, $e = 1$ i przechodzimy do punktu 2. W przeciwnym wypadku przechodzimy do punktu 8.
 8. Jeżeli $k < w$ to przyjmujemy $k = k + 1$, $i = 1$, $e = 1$ i przechodzimy do punktu 2. Jeżeli $k = w$ to wykonywanie algorytmu jest zakończone.

Uwaga: W każdym kroku iteracyjnym po zakończeniu budowy rodziny M_k można usunąć z pamięci rodzinę M_{k-1} .

Układ równań 5.1 możemy obliczyć tylko metodą iteracyjną, gdyż nie znamy postaci funkcji wektorowych $\vec{\Phi}_{V_k}$ zależnych od obiektów, połączeń między nimi i przyjętego zbioru przecinającego I . Ponieważ podgraf $\langle N, U-I \rangle$ nie posiada pętli i konturów więc możemy w kolejnych iteracjach /wychodząc od przyjętych pewnych wartości początkowych/ obliczać wartości cech strumieni informacyjnych odpowiadających łukom ze zbioru I tak długo, aż osiągniemy żadaną dokładność obliczeń. Mamy tutaj do wyboru iteracje proste

opisane wzorami

$$\vec{c}_{v_k}^{(l+1)} = \vec{\Phi}_{v_k}(\vec{c}_{v_1}^{(l)}, \vec{c}_{v_2}^{(l)}, \dots, \vec{c}_{v_t}^{(l)}) \quad k = 1, 2, \dots, t$$

gdzie l jest numerem iteracji, lub bardziej efektywne metody przedstawione w literaturze [21,32,35,36]. Proces iteracyjny nie zawsze musi być zbieżny. Zależy to od przyjętych wartości początkowych, wyboru zbioru przecinającego i zastosowanej metody iteracyjnej. Pozostaje nam jeszcze do ustalenia kolejność obliczania obiektów rozpatrywanego σ -podsystemu informacyjnego w ramach jednej iteracji. Zdefiniujmy dla elementów zbioru N relację $\sqsubset_{N,I}$ w następujący sposób

$$x_i \sqsubset_{N,I} x_j \Leftrightarrow \text{w podgrafie } \langle N, U-I \rangle \text{ istnieje droga } L(x_i, x_j) \\ \text{o d\kern-0.08em l} \geq 0$$

Łatwo można wykazać, że jest ona zwrotna, antysymetryczna i przechodnia, a więc jest relacją porządkującą zbiór N dla danego zbioru przecinającego I . Uporządkowanie to jest zgodne z kierunkami przepływu informacji pomiędzy obiektami σ -podsystemu informacyjnego dla każdej iteracji. Zatem wyznacza nam ono szukaną kolejność obliczania obiektów.

III. OGÓLNA CHARAKTERYSTYKA PROGRAMU ORGANIZACYJNEGO

Blok aparaturowy definiujemy jako matematyczny model obiektu systemu. Natomiast blok fikcyjny jest to pewien algorytm niezbędny dla przeprowadzenia obliczeń, który nie ma żadnego odpowiednika w ciągu technologicznym, np. blok rozwiązywania układu równań, blok optymalizacyjny, blok funkcji celu, blok ekonomiczny. Pod pojęciem bloku będziemy rozumieli blok aparaturowy lub blok fikcyjny.

Program organizacyjny BOSS ogólnie rzecz biorąc składa się z segmentów procedur czytania danych READ KEY i READ DATA, segmentu procedury drukowania wyników PRINT DATA, segmentu głównego oraz segmentu procedury JUMPER w której deklarujemy procedury opisujące bloki / bliższe informacje o segmentach znajdują się w literaturze [1]/. Z wyjątkiem segmentu procedury JUMPER, wszystkie wymienione wyżej segmenty oraz segmenty procedur opisujących bloki będą przechowywane w postaci półskompilowanej w zbiorach na taśmach magnetycznych / dla kompilatora #XALM / albo na dyskach / dla kompilatora #XALT / i dołączone do programu BOSS w czasie konsolidacji / ze wszystkich segmentów procedur opisujących bloki dołączone są tylko te, które są nam potrzebne tzn. te które zadeklarowaliśmy w procedurze JUMPER /. Procedury opisujące bloki muszą być napisane w języku ALGOL 1900 / są one kompilowane przez odpowiedni kompilator i zapisywane w postaci półskompilowanej do zbiorów /. Nośnikiem segmentów procedur opisujących bloki mogą być także karty perforowane ale nie jest to dogodne z tego względu, że wtedy za każdym razem muszą być one kompilowane. Segment procedury JUMPER oraz wiersze sterujące programem muszą znajdować się na kartach perforowanych. Program BOSS może być programem

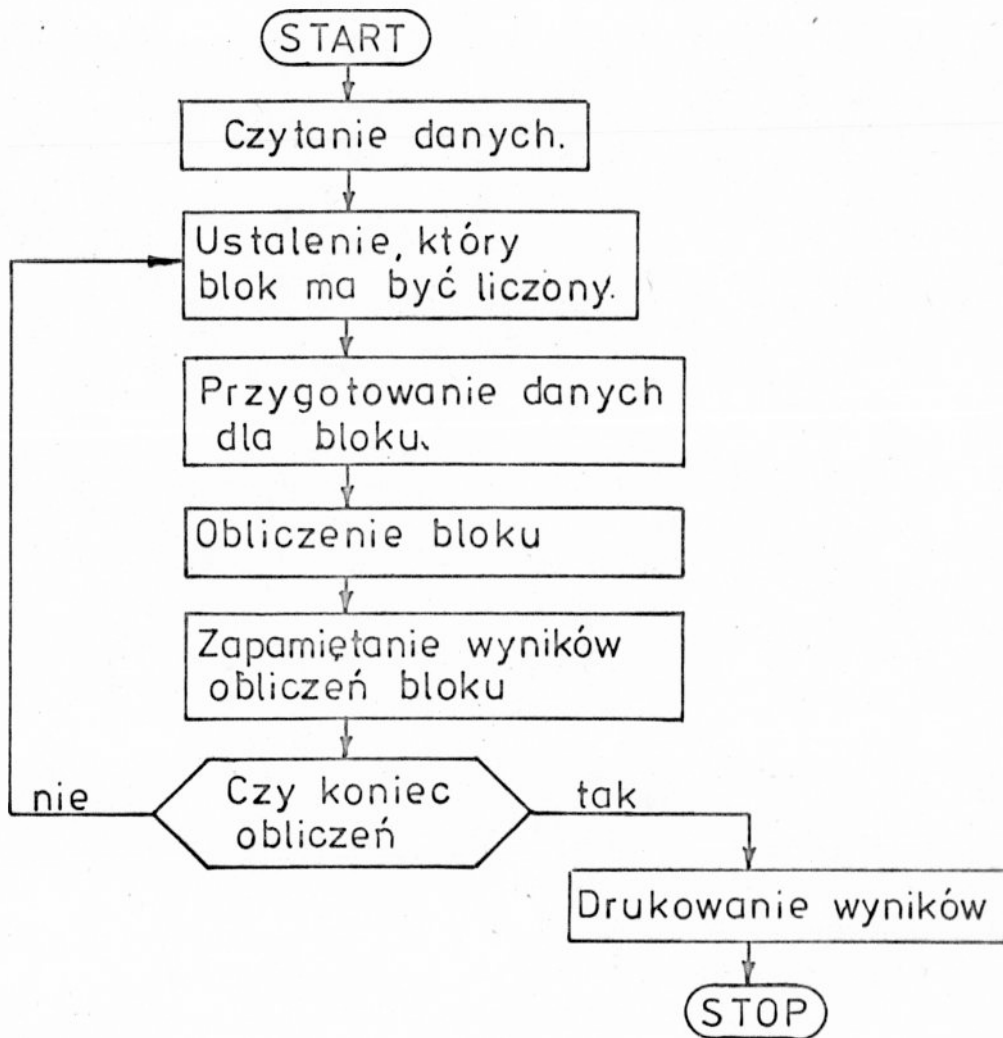
nakładanym, gdyż procedury opisujące bloki oraz procedury READ KEY, READ DATA i PRINT DATA są tak pomyślane, że mogą być nakładkami. Segment główny programu spełnia następujące funkcje

1. ustala jaki blok ma być aktualnie obliczany
2. przygotowuje dane dla tego bloku
3. wywołuje / poprzez procedurę JUMPER / procedurę opisującą dany blok
4. zapamiętuje wyniki działania procedury

Kolejność obliczania bloków jest już znana wcześniej dzięki programowi analizy systemów ANALYZER. Podczas działania programu dla procedur opisujących bloki dostępne są następujące zbiory danych zapisane na taśmach magnetycznych:

1. ADDATA - zawierający informacje zależące od konkretnego zadania obliczeniowego
2. LIBDATA - 1 } w zbiorach tych przechowywane są informacje
3. LIBDATA - 2 } nie zależące od konkretnego zadania obliczeniowego np. normy
4. PHCHP - zawierający dane do obliczeń własności fizykochemicznych substancji

Informacje do tych zbiorów są wprowadzane za pomocą programu ARCHIVIST. Poza tym program BOSS może wprowadzać informacje do zbioru ADDATA. Procedury opisujące bloki rozwiązywania układu równań i bloki optymalizacyjne mogą zapisywać pewne uzyskane w czasie obliczeń wyniki do zbioru STORE / jego nośnikiem jest taśma magnetyczna / które są potem drukowane przez procedurę PRINT DATA. Ogólny schemat działania programu BOSS jest przedstawiony na rys. III.1.



Rys. III.1

Program BOSS przeprowadza obliczenia dla systemu informacyjnego odpowiadającego danemu systemowi przy czym w danych wprowadzamy strukturę połączeń obiektów systemu, a "wejścia" i "wyjścia" procedur opisujących bloki aparaturowe odpowiadają wejściom i wyjściom obiektów systemu. Zaletą programu BOSS jest jego duża elastyczność umożliwiającą nam dobór nie tylko bloków aparaturowych ale również bloków fikcyjnych najlepiej dostosowanych do danego zadania obliczeniowego.

Poza tym prawie wszystkie tablice zadeklarowane w segmencie głównym programu są tablicami o granicach dynamicznych co pozwala w każdym konkretnym przypadku zajmować tylko taki obszar pamięci operacyjnej jaki jest niezbędny do obliczeń.

Etapy liczenia zadania podajemy niżej:

1. Budowa systemu informacyjnego odpowiadającego danemu systemowi
2. Analiza systemu informacyjnego
3. Analiza zadania optymalizacji systemu / wtedy gdy jest potrzebna /
4. Obliczenia systemu informacyjnego

Etapy 1 ÷ 3 mogą być wykonywane przez program ANALYZER lub w prostszych przypadkach przez człowieka. Program organizacyjny BOSS koordynuje tylko ostatni etap obliczeń.

IV. OPIS STOSOWANYCH W PROGRAMIE ORGANIZACYJNYM TABLIC

1. PODSTAWOWE DEFINICJE

Każdemu strumieniowi systemu przyporządkowujemy w sposób wzajemnie jednoznaczny liczbę naturalną mniejszą lub równą 999. Liczby te będziemy nazywali numerami zewnętrznymi strumieni. Każdemu blokowi występującemu w danym zadaniu obliczeniowym przyporządkowujemy w sposób wzajemnie jednoznaczny liczbę naturalną mniejszą lub równą 999 przy czym liczba przypisana dowolnemu blokowi fikcyjnemu musi być większa od liczby odpowiadającej każdemu blokowi aparaturowemu. Liczby te będziemy nazywali numerami zewnętrznymi bloków. Program organizacyjny BOSS porządkuje bloki / strumienie / wg wzrastających numerów zewnętrznych, a następnie przyporządkowuje blokom / strumieniom / kolejne liczby naturalne poczynając od 1, które będziemy nazywali numerami wewnętrznymi bloków / strumieni /

Rodzaj fazy substancji lub mieszaniny będziemy kodować następująco:

- 1 - gaz
- 2 - ciecz
- 3 - ciało stałe

Każdej substancji występującej w ciągu technologicznym przypisujemy w sposób wzajemnie jednoznaczny liczbę naturalną mniejszą lub równą 8388607, którą nazywamy numerem substancji.

Niech danemu strumieniowi systemu odpowiada pewien strumień ciągu technologicznego o k fazach i l składnikami. Identyfikatorem tego strumienia systemu nazywamy ciąg następujących $l+k+3$ liczb

1. liczba cech strumienia /liczba elementów w charakterystyce strumienia /
2. liczba faz strumienia
3. liczba składników strumienia
4. rodzaj 1-szej fazy
5. rodzaj 2-ej fazy
- ⋮
- k+3. rodzaj k-tej fazy
- k+4. nr 1-go składnika /nr substancji/
- k+5. nr 2-go składnika /nr substancji/
- ⋮
- 1+k+3. nr 1-tego składnika /nr substancji/

Zaś przez charakterystykę tego strumienia rozumiemy ciąg następujących $k(1+1)+2$ liczb

1. temperatura strumienia [K]
2. ciśnienie [N/m^2]
3. molowe natężenie przepływu 1-szej fazy [kmol/s]
4. ułamek molowy 1-go składnika w 1-szej fazie [kmol/kmol]
5. ułamek molowy 2-go składnika w 1-szej fazie [kmol/kmol]
- ⋮
- 1+3. ułamek molowy 1-tego składnika w 1-szej fazie [kmol/kmol]
- 1+4. molowe natężenie przepływu 2-ej fazy [kmol/s]
- 1+5. ułamek molowy 1-go składnika w 2-ej fazie [kmol/kmol]
- 1+6. ułamek molowy 2-go składnika w 2-ej fazie [kmol/kmol]
- ⋮
- 2l+4. ułamek molowy 1-tego składnika w 2-ej fazie [kmol/kmol]
- ⋮
- $(k-1)(1+1)+3$. molowe natężenie przepływu k-tej fazy [kmol/s]
- $(k-1)(1+1)+4$. ułamek molowy 2-go składnika w k-tej fazie [kmol/kmol]

$(k-1)(l+1)+5$. ułamek molowy 2-go składnika w k-tej fazie

[kmol/kmol]

⋮

$k(l+1)+2$ ułamek molowy l-tego składnika w k-tej fazie

[kmol/kmol]

Formatem drukowania będziemy nazywać liczbę mn / jest to zapis pozycyjny / gdzie m, n oznaczają odpowiednio ilość cyfr przed przecinkiem i po przecinku / z pewnymi wyjątkami / oraz są takie, że $0 \leq m \leq 9$, $0 \leq n \leq 9$. Jeżeli format drukowania jest równy zeru to wartość zmiennej arytmetycznej nie będzie drukowana. Jeżeli natomiast format drukowania jest liczbą większą od zera to wartość pewnej zmiennej arytmetycznej E będzie drukowana tak jak przez instrukcję `print ( E, m, n )`; patrz opis procedury `print` w literaturze [2,3].

Każdej tablicy występującej w segmencie głównym programu organizacyjnego BOSS przyporządkowana jest wzajemnie jednoznacznie liczba naturalna zwana kodem tablicy.

Cechami rozerwanymi nazywamy te cechy strumieni, których wartości trzeba zadać aby móc obliczyć σ - podsystem informacyjny metodą iteracyjną.

Parametrami bloków nazywamy cechy obiektów lub pewne wielkości związane z blokami.

Kod cechy strumienia / parametru bloku / jest to liczba $k \cdot 10^5 + l \cdot 10^2 + m$ gdzie k jest kodem tablicy ST / PB albo APB /, l numerem zewnętrznym strumienia bloku a m numerem kolumny tablicy o kodzie k w której znajduje się dana cecha strumienia / parametr bloku /. Kodem zmiennej decyzyjnej nazywamy kod cechy strumienia lub obiektu będącej daną zmienną decyzyjną. Przez kod zmiennej

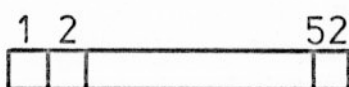
funkcji celu rozumiemy kod cechy strumienia lub parametru bloku, którego wartość zależna pośrednio od zmiennych decyzyjnych służy do obliczania wartości funkcji celu. Kodem funkcji celu nazywamy kod cechy strumienia lub parametru bloku, którego wartość równa się wartości funkcji celu.

2. TABLICE WYSTĘPUJĄCE W SEGMENTCIE GŁÓWNYM PROGRAMU ORGANIZACYJNEGO

KEY - klucz

typ: integer

kod tablicy: 1



Znaczenie poszczególnych elementów:

1. nr wewnętrzny bloku
2. liczba strumieni wchodzących do danego bloku
3. liczba strumieni wychodzących z danego bloku
4. rodzaj obliczeń realizowanych przez blok
 - =0 - technologiczne dla bloków aparaturowych albo niekone-
 - miczne dla bloków fikcyjnych
 - >0 - ekonomiczne
5. typ bloku /patrz opis procedury JUMPER/
6. sygnalizacja błędu
 - =0 - nie ma błędu
 - >0 - nie wiadomo do jakiego bloku wykonać skok
7. sygnalizacja błędu przez bloki
 - =0 - nie ma błędu
 - >0 - nr zaistniałego błędu
8. czy procedury opisujące bloki potrzebują do obliczeń dane ze zbioru ADDATA
 - 0 - nie

- 1 - tak przy czym nie trzeba wczytywać danych do zbioru
- 2 - tak przy czym trzeba utworzyć zbiór i wczytać dane do niego
- 3 - tak przy czym trzeba wczytać dane do zbioru
- 9. czy procedury opisujące bloki potrzebują do obliczeń dane ze zbioru LIBDATA - 1
 - 0 - nie
 - 1 - tak
- 10. czy procedury opisujące bloki potrzebują do obliczeń dane ze zbioru LIBDATA - 2
 - 0 - nie
 - 1 - tak
- 11. liczba elementów tablicy NO
- 12. liczba strumieni w systemie /liczba wierszy IST i ST/
- 13. maksymalny nr kolumny w IST /maksymalna liczba elementów typu integer w wierszu tablicy *SNISTST zwiększona o 1/
- 14. maksymalna liczba cech strumienia w systemie
- 15. maksymalna liczba strumieni wchodzących do pojedynczego bloku/liczba wierszy IS i IIS/
- 16. maksymalna liczba strumieni wychodzących z pojedynczego bloku / liczba wierszy OS i IOS/
- 17. liczba strumieni zawierających cechy rozerwane /liczba wierszy IIV/
- 18. liczba kolumn IIV
- 19. liczba strumieni dla których zadajemy ograniczenia /liczba wierszy LS/
- 20. liczba kolumn LS
- 21. liczba wierszy CE
- 22. liczba wierszy ECE

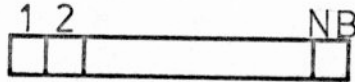
- 23. liczba kolumn ECE
- 24. liczba wierszy SI
- 25. liczba kolumn SI
- 26. czy drukować SI
 - 0 - nie
 - 1 - tak
- 27. liczba bloków aparaturowych
- 28. liczba bloków fikcyjnych
- 29. liczba różnych typów bloków / równa się ona liczbie wierszy FPB /
- 30. liczba kolumn PM /maksymalna liczba elementów typu integer w wierszu tablicy *BNPMPB zwiększona o 1/
- 31. liczba kolumn PB
- 32. czy drukować PB
 - = -1 - tak w formacie zadanym przez FPB
 - = 0 - nie
 - > 0 - tak w formacie jednolitym zadanym przez tę liczbę
- 33. liczba wierszy APB
- 34. liczba kolumn APB
- 35. czy drukować APB
 - = -1 - tak w formacie zadanym przez FAP
 - = 0 - nie
 - > 0 - tak w formacie jednolitym zadanym przez tę liczbę
- 36. liczba wierszy FAP
- 37. czy EP jest potrzebna i czy ją drukować
 - = 0 - nie jest potrzebna i nie drukować
 - > 0 - jest potrzebna i drukować w formacie zadanym przez tę liczbę

- 38. liczba wierszy LV
- 39. liczba kolumn LV
- 40. liczba wierszy ME
- 41. liczba kolumn ME
- 42. czy drukować ME
 - = -1 - tak w formacie zadanym dla kolumn przez FME
 - = 0 - nie
 - > 0 - tak w formacie jednolitym zadanym przez tą liczbę
- 43. liczba wierszy PCP / liczba substancji występujących w procesie technologicznym, a ściślej liczba substancji uwzględnionych w modelu tego procesu /
- 44. liczba kolumn PCP
- 45. format drukowania temperatur strumieni
- 46. format drukowania ciśnień strumieni
- 47. format drukowania molowych natężeń przepływu poszczególnych faz strumieni
- 48. format drukowania ułamków molowych poszczególnych składników strumieni
- 49. sygnalizacja błędu
 - < 0 - nieprawidłowa numeracja w tablicy / błąd występujący przy czytaniu danych /
 - = 0 - nie ma błędu
 - > 0 - błąd wykonania
- 50. sygnalizacja wykonywania skoku
 - 0 - nie wykonywać skoku
 - 1 - wykonać skok do bloku wskazanego przez NO
- 51. licznik par rekordów DSI, SI zapisanych do zbioru STORE
- 52. licznik par rekordów DME, ME zapisanych do zbioru STORE

CS - ciąg obliczeń

typ: integer

kod tablicy: 2



NB - liczba bloków zadania obliczeniowego

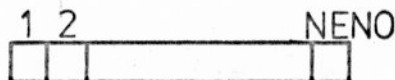
$NB \leq 999$

Elementami tablicy CS są numery bloków / wewnętrzne w segmencie głównym programu albo zewnętrzne w danych / ułożone wg kolejności obliczania odpowiadających im bloków. Każdy blok może wystąpić tylko jeden raz w CS. Elementy tablicy CS równe zero są pomijane przez program. Zmiana kolejności obliczeń niezbędna przy obliczeniach podsystemów informacyjnych, optymalizacji i obliczeniach ekonomicznych jest możliwa tylko na podstawie informacji zawartych w tablicy NO.

NO - nowy porządek

typ: integer

kod tablicy: 3



NENO - liczba elementów tablicy NO

Elementy tablicy NO opisują nam konieczne przy obliczeniach podsystemów informacyjnych, optymalizacji i obliczeniach eko-

nomicznych, zmiany przyjętej w tablicy CS kolejności obliczeń. Znaczenie elementów NO / w danych podajemy numery zewnętrzne bloków / jest następujące:

NO [1], NO [2]- po obliczeniu bloku o numerze zewnętrznym NO [1] /o numerze wewnętrznym CS[NO[1]]/przejdź do obliczania bloku o numerze zewnętrznym NO [2]/o numerze wewnętrznym CS [NO[2]] /

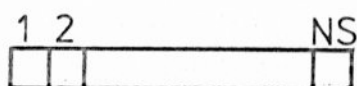
NO [3], NO [4]- po obliczeniu bloku o numerze zewnętrznym NO [3] /o numerze wewnętrznym CS[NO[3]]/przejdź do obliczania bloku o numerze zewnętrznym NO[4] /o numerze wewnętrznym CS[NO[4]]/.

⋮
itd.

SN - numery strumieni

typ: integer

kod tablicy: 4



NS - liczba strumieni w systemie

NS $\leq$ 999

Elementami tablicy SN są numery zewnętrzne strumieni poukładane w ten sposób, że dla $I < J$ mamy $SN[I] < SN[J]$. Zauważmy, że strumień o numerze zewnętrznym $SN[I]$ posiada numer wewnętrzny I.

IST - identyfikacja strumieni

typ: integer

kod tablicy: 5

	3	4		NCIS
1				
i				
NS				

NS - liczba strumieni w systemie

$NS \leq 999$

NCIS - maksymalny numer kolumny w IST

Tablica IST zawiera opisy wszystkich strumieni systemu

Strumieniowi o numerze wewnętrznym i odpowiada i -ty wiersz tablicy IST. Niech strumień ten ma k faz i l składników, Znaczenie poszczególnych kolumn IST dla i -tego wiersza jest wtedy następujące

3.	}	identyfikator strumienia
4.		
⋮		
$l+k+5.$		
$l+k+6.$	}	zera, jeżeli $l+k+5 < NCIS$
⋮		
NCIS.		

ST - strumienie

typ: real

kod tablicy: 6

	1	2		MNFS
1				
i				
NS				

NS - liczba strumieni w systemie

$NS \leq 999$

MNFS - maksymalna liczba cech strumienia w systemie

$MNFS \leq 99$

W tablicy ST pamiętane są wartości cech wszystkich strumieni systemu.

Strumieniowi o numerze wewnętrznym i odpowiada i-ty wiersz tablicy ST. Niech strumień ten ma k faz i l składników. Znaczenie poszczególnych kolumn ST dla i-tego wiersza jest wtedy następujące

- | | | |
|-------------|---|--------------------------------|
| 1. | } | charakterystyka strumienia |
| 2. | | |
| ⋮ | | |
| $k(l+1)+2.$ | | |
| $k(l+1)+3.$ | } | zera, jeżeli $k(l+1)+2 < MNSS$ |
| ⋮ | | |
| ⋮ | | |
| MNSS. | | |

IIS / IOS/ - identyfikacja strumieni wejściowych / wyjściowych/

typ: integer

kod tablic: 7/9/

	1	2		NCIS
1				
i				
MNIS(MNOS)				

MNIS - maksymalna liczba strumieni wchodzących do pojedynczego bloku

MNOS - maksymalna liczba strumieni wychodzących z pojedynczego bloku

NCIS - liczba kolumn IST

Tablica IIS / IOS/ zawiera opisy n strumieni wpływających do / wpływających z / danego bloku umieszczonych w pierwszych n wierszach i uporządkowanych wg wzrastających numerów wejść / wyjść / bloku. Znaczenie poszczególnych kolumn dla i-tego wiersza w którym opisany jest strumień o k fazach i l składnikach jest następujące:

1. numer wewnętrzny strumienia
2. numer wejścia / wyjścia / bloku z którym związany jest strumień
3. $\left. \begin{array}{l} \\ \\ l+k+5. \end{array} \right\}$ identyfikator strumienia

$$\left. \begin{array}{l} 1+k+6. \\ \\ NCIS. \end{array} \right\} \text{ zera, jeżeli } 1+k+5 < NCIS$$

IS / OS/ - strumienie wpływające do / wypływające z / danego bloku

typ: real

kod tablic: 8 / 10 /

	1	2		MNFS
1				
i				
MNIS(MNOS)				

MNIS - maksymalna liczba strumieni wpływających do danego bloku

MNOS - maksymalna liczba strumieni wypływających z danego bloku

MNFS - maksymalna liczba cech strumienia w systemie

$MNFS \leq 99$

W tablicy IS / OS/ pamiętane są wartości cech strumieni wpływających do / wypływających z / danego bloku umieszczone w pierwszych n wierszach, przy czym jeden wiersz odpowiada jednemu strumieniowi i wiersze uporządkowane są wg wzrastających numerów wejść / wyjść / bloku. Wierszowi o numerze i odpowiada strumień o numerze wewnętrznym IIS[i,1]/IOS[i,1]/ tzn. odpowiadamy ten sam strumień, który opisany jest w i-tym wierszu tablicy IIS/IOS/. Zna-

czenie poszczególnych kolumn dla i -tego wiersza, któremu odpowiada strumień o k fazach i l składnikach jest następujące:

1.	}	charakterystyka strumienia
2.		
⋮		
$k(l+1)+2$.	}	zera, jeżeli $k(l+1)+2 < MNSS$
$k(l+1)+3$.		
⋮		
$MNSS$.		

IIV - identyfikacja cech rozerwanych / iterowanych /

typ: integer

kod tablicy: 11

	1	2		NCIV
1				
⋮				
i				
NTS				

NTS - liczba strumieni zawierających cechy rozerwane

NCIV - liczba kolumn IIV

Tablica IIV zawiera numery kolumn tablicy ST, w których występują cechy rozerwane strumieni, przy czym jeden wiersz tablicy odpowiada jednemu strumieniowi. Znaczenie poszczególnych kolumny dla i -tego wiersza, który zawiera k cech rozerwanych jest następujące:

1. numer wewnętrzny strumienia

2. liczba cech rozerwanych występujących w strumieniu

3. $\left\{ \begin{array}{l} \vdots \\ \vdots \end{array} \right\}$ numery kolumn tablicy ST w których występują cechy ro-
 $\left\{ \begin{array}{l} \vdots \\ \vdots \end{array} \right\}$ zerwane danego strumienia, uporządkowane wg wielkości
 $k+2.$ /mniejsze numery kolumn poprzedzają numery większe/
 $k+3.$ $\left\{ \begin{array}{l} \vdots \\ \vdots \end{array} \right\}$ zera, jeżeli $k+2 < NCIV$
 $NCIV.$

LS - ograniczenia wartości cech strumieni

typ: real

kod tablicy: 12

	1	2		NLCS
1				
i				
NSL				

Tablica LS zawiera ograniczenia wartości cech strumieni. Ma ona między innymi zastosowanie w procedurach opisujących bloki: rozwiązywania układu równań i aparaturowe. Jeden wiersz tablicy odpowiada jednemu strumieniowi. Wiersze tablicy uporządkowane są wg numerów strumieni tzn. wiersze odpowiadające strumieniom o mniejszych numerach poprzedzają wiersze odpowiadające strumieniom o większych numerach. Najmniejsza wartość ograniczenia dolnego jaką można przyjąć jest równa zeru. Największa wartość ograniczenia

górnego jest natomiast $5,6 : 10^{76}$. Znaczenie poszczególnych kolumn i -tego wiersza odnoszącego się do strumienia scharakteryzowanego przez k cech jest następujące:

1. numer strumienia / wewnętrzny w programie a zewnętrzny w danych /
2. ograniczenie dolne cechy znajdującej się w 1-szej kolumnie ST
3. ograniczenie górne cechy znajdującej się w 1-szej kolumnie ST
4. ograniczenie dolne cechy znajdującej się w 2-ej kolumnie ST
5. ograniczenie górnej cechy znajdującej się w 2-ej kolumnie ST
- $\vdots$
- $2k$. ograniczenie dolne cechy znajdującej się w k -tej kolumnie ST
- $2k+1$. ograniczenie górne cechy znajdującej się w k -tej kolumnie ST
- $2k+2$.
 $\vdots$
 $\vdots$
 $NCLS$.
} zera, jeżeli $2k+1 < NCLS$

CE - błędy obliczeń

typ: real

kod tablicy: 13

	1	2		5
1				
i				

NRCE - liczba wierszy CE

Tablica CE znajduje zastosowanie w procedurach opisujących bloki rozwiązywania układu równań dla określenia dokładności z jaką muszą być obliczone cechy rozerwane strumieni. Jeden wiersz CE odpowiada jednemu strumieniowi. Wiersze tej tablicy są uporządkowane wg numerów strumieni tzn. wiersz któremu odpowiada strumień o mniejszym numerze poprzedza wiersz odpowiadający strumieniowi o numerze większym. Znaczenie poszczególnych kolumn tablicy jest następujące:

1. nr strumienia / wewnętrzny w programie, a zewnętrzny w danych /
2. dokładność z jaką należy obliczyć temperaturę strumienia
3. dokładność z jaką należy obliczyć ciśnienie strumienia
4. dokładność z jaką należy obliczyć molowe natężenia przepływu poszczególnych faz strumienia
5. dokładność z jaką należy obliczyć ułamki molowe składników występujących w poszczególnych fazach strumienia

Jeżeli dokładność jest podana ze znakiem plus to interpretujemy ją jako maksymalny błąd względny / nie wolno podawać go w procentach / między dwoma kolejnymi przybliżeniami, jeżeli natomiast jest podana ze znakiem minus to traktujemy ją / a ściślej jej moduł / jako maksymalny błąd bezwzględny między dwoma kolejnymi przybliżeniami. Jeżeli dokładności dla cech pewnego strumienia zostały podane w CE to nie można podać dokładności dla cech tego samego strumienia w ECE i na odwrót.

ECE - rozszerzone błędy obliczeń

typ: real

kod tablicy: 14

	1	2	NCEE
1			
i			
NREE			

NREE - liczba wierszy ECE

NCEE - liczba kolumn ECE

Tablica ECE znajduje zastosowanie w procedurach opisujących bloki rozwiązywania układu równań dla określenia dokładności z jaką muszą być obliczone cechy rozzerwane strumieni. Jeden wiersz ECE odpowiada jednemu strumieniowi. Wiersze tej tablicy są uporządkowane wg numerów strumieni tzn. wiersz któremu odpowiada strumień o mniejszym numerze poprzedza wiersz odpowiadający strumieniowi o numerze większym. Znaczenie poszczególnych kolumn dla i-tego wiersza, któremu odpowiada strumień scharakteryzowany przez k cech jest następujące:

1. nr strumienia / wewnętrzny w programie, a zewnętrzny w danych /
2. dokładność z jaką należy obliczyć cechę znajdującą się w 1-szej kolumnie ST
3. dokładność z jaką należy obliczyć cechę znajdującą się w 2-jej kolumnie ST

⋮

k+1. dokładność z jaką należy obliczyć cechę znajdującą się
w k-tej kolumnie ST

$\left. \begin{array}{l} k+2. \\ \vdots \\ NREE. \end{array} \right\} \text{zera, jeżeli } k+1 < NREE$

Jeżeli dokładność jest podana ze znakiem plus to interpretujemy ją jako maksymalny błąd względny /nie wolno podawać go w procentach /między dwoma kolejnymi przybliżeniami, jeżeli natomiast jest podana ze znakiem minus to jej moduł traktujemy jako maksymalny błąd bezwzględny między dwoma kolejnymi przybliżeniami. Jeżeli dokładności dla cech pewnego strumienia zostały podane w ECE to nie wolno podać dokładności dla cech tego samego strumienia w CE i na odwrót.

SI - magazyn iteracji /rozerwanych cech strumieni/

typ: real

kod tablicy: 15

	1	2		NCSI
1				
i				
NRSI				

NRSI - liczba wierszy SI

NCSI - liczba kolumn SI

Tablica SI znajduje zastosowanie w blokach rozwiązywania układów równań. Liczba wierszy tablicy jest równa większej z dwóch liczb:

a/ liczbie iteracji jaką trzeba znać aby można określić następne przybliżenia rozerwanych cech strumieni przy obliczeniach

σ - podsystemów informacyjnych

b/ maksymalnej liczbie iteracji jaką chcemy na raz zapamiętać w SI.

Jeden wiersz tablicy odpowiada jednej iteracji. W kolumnach znajdują się rozerwane cechy strumieni dla danego σ - podsystemu informacyjnego /liczba kolumn równa się maksymalnej liczbie rozerwanych cech strumieni dla obliczanych w zadaniu σ - podsystemów informacyjnych/, przy czym cechy rozerwane jednego strumienia są zgrupowane razem i uporządkowane wg numerów kolumn ST w których się znajdują tzn. cechy znajdujące się w kolumnie o numerze mniejszym poprzedza cechę znajdującą się w kolumnie o większym numerze. Natomiast kolejność samych strumieni jest zadana przez blok rozwiązywania układu równań. Jeżeli dla danego

σ - podsystemu informacyjnego cechy rozerwane występują w k strumieniach i jest ich l_1 w jednym strumieniu, l_2 w drugim strumieniu, ..., l_k w ostatnim strumieniu to znaczenia poszczególnych kolumn są następujące:

$$\left. \begin{array}{l} 1. \\ 2. \\ \vdots \\ l_1 \end{array} \right\} \begin{array}{l} \text{1-sza grupa / cechy rozerwane pierwszego strumienia} \\ \text{uporządkowane wg numerów kolumn ST /} \end{array}$$

$$\left. \begin{array}{l} l_1+1. \\ \vdots \\ l_1+l_2. \end{array} \right\} \begin{array}{l} \text{2-ga grupa / cechy rozerwane drugiego strumienia uporząd-} \\ \text{kowane wg numerów kolumn ST /} \end{array}$$

$$\vdots$$

$$1 + \left. \begin{array}{c} \sum_{j=1}^{k-1} 1_{j'} \\ \vdots \\ \sum_{j=1}^k 1_{j'} \end{array} \right\} \begin{array}{l} k\text{-ta grupa / cechy rozerwane ostatniego strumienia} \\ \text{uporządkowane wg numerów kolumn SI /} \end{array}$$

Kolejność występowania powyższych grup cech rozerwanych / tzn. sposób przydzielenia kolumn poszczególnym grupom przez blok rozwiązywania układu równań / jest nieistotna. Ważne jest tylko uporządkowanie wewnątrz grupy cech. Oczywiście raz przyjęty sposób rozmieszczenia grup cech przez blok rozwiązywania układu równań / tzn. dla danego σ -podsystemu informacyjnego / musi być zachowany dla każdej następnej iteracji. Tablica SI może być użytkowana przez kilka bloków rozwiązywania układu równań, ale tylko w ten sposób, że każdy następny blok może korzystać z SI gdy poprzedni zakończył obliczanie swojego układu równań /swojego σ -podsystemu informacyjnego/. Każdy blok rozwiązywania układu równań musi przed rozpoczęciem obliczeń σ -podsystemu informacyjnego wyzerować tablicę SI. Jeżeli blok rozwiązywania układu równań zapisuje w SI informacje potrzebne do określenia następnego przybliżenia rozerwanych cech strumieni na podstawie większej od 1 liczby iteracji to nowe przybliżenie wpisuje do pierwszego wolnego wiersza, a następnie przesuwając wszystkie wypełnione wiersze o jeden wiersz w górę wyrzucając wiersz pierwszy i zerując wiersz gdzie poprzednio zapisane było ostatnie przybliżenie. Jeżeli natomiast blok zapisuje w SI kolejne iteracje to z chwilą gdy SI zostanie wypełniona, przepisuje ją do zbioru STORE /patrz opis tego zbioru/, zeruje i kolejne iteracje zapamiętuje od góry. Możliwe są też inne sposoby użytkowania SI np. do pamiętania iteracji a jednocześnie blok może określać nowe przy-

bliżenia na podstawie pewnej liczby poprzednich. Zależy to wyłącznie od naszych potrzeb i inwencji programisty.

BN - numery bloków

typ: integer

kod tablicy: 16



NAB - liczba bloków aparaturowych

NB - liczba bloków

$NB \leq 999$

Elementami tablicy BN są numery zewnętrzne bloków poukładane w ten sposób, że dla $I < J$ mamy $BN[I] < BN[J]$. Zauważmy, że blok o numerze zewnętrznym $BN[I]$ posiada numer wewnętrzny I . Poza tym elementami o numerach od 1 do NAB są numery zewnętrzne bloków aparaturowych, a elementami o numerach od NAB+1 do NB są numery zewnętrzne bloków fikcyjnych.

PM - macierz procesu

typ: integer

kod tablicy: 17

NAB - liczba bloków aparaturowych

NB - liczba bloków

$NB \leq 999$

NCPM - liczba kolumn PM

	1	2		NCPM
1				
i				
NAB				
NAB+1			.	
NB				

Tablica PM zawiera opis struktury systemu. Znaczenie poszczególnych kolumn dla i -tego wiersza, któremu odpowiada blok o numerze wewnętrznym i mający k strumieni wejściowych i l strumieni wyjściowych jest następujące:

1. liczba strumieni wpływających do bloku
2. liczba strumieni wypływających z bloku
3. typ bloku / patrz opis procedury JUMPER /
4. nr wewnętrzny strumienia wpływającego do bloku
5. nr wejścia bloku do którego wpływa strumień z punktu 4
6. nr wewnętrzny strumienia wpływającego do bloku
7. nr wejścia bloku do którego wpływa strumień z punktu 6
- ⋮
- $2k+2$. nr wewnętrzny strumienia wpływającego do bloku
- $2k+3$. nr wejścia bloku do którego wpływa strumień z punktu $2k+2$
- $2k+4$. nr wewnętrzny strumienia wypływającego z bloku
- $2k+5$. nr wyjścia bloku z którego wypływa strumień z punktu $2k+4$

$2k+6$. nr wewnętrzny strumienia wypływającego z bloku

$2k+7$. nr wyjścia bloku z którego wpływa strumień z punktu $2k+6$

⋮

$2(k+1)+2$. nr wewnętrzny strumienia wypływającego z bloku

$2(k+1)+3$. nr wyjścia bloku z którego wpływa strumień
z punktu $2(k+1)+2$

$2(k+1)+4$.
⋮
⋮
NCPM. } zera, jeżeli $2(k+1)+3 < \text{NCPM}$

Numerы strumieni wejściowych w kolumnach $4, 6, \dots, 2k+2$ są uporządkowane wg wielkości tzn. numery mniejsze poprzedzają numery większe. Numerы strumieni wyjściowych w kolumnach $2k+4, 2k+6, \dots, 2(k+1)+2$ są również uporządkowane w ten sam sposób. Jeżeli pewnemu wierszowi odpowiada blok fikcyjny wówczas wszystkie elementy tego wiersza z wyjątkiem elementu trzeciego są równe zeru. Wiersze o numerach od 1 do NAB odpowiadają blokom aparaturowym, a wiersze od NAB+1 do NB blokom fikcyjnym.

PB - parametry bloków

typ: real

kod tablicy: 18

NAB - liczba bloków aparaturowych

NB - liczba bloków

NB $\leq$ 999

NCPB - liczba kolumn PB

NCPB $\leq$ 99

	1	2		NCPB
1				
i				
NAB				
NAB+1				
NB				

W tablicy PB znajdują się parametry bloków, przy czym i-temu wierszowi odpowiada blok o numerze wewnętrznym i. Wiersze o numerach od 1 do NAB odpowiadają blokom aparaturowym a od NAB+1 do NB blokom fikcyjnym. Zaleca się aby parametry bloków aparaturowych znajdujące się w tej tablicy charakteryzowały rzeczywiste własności aparatów lub zachodzących w nich procesów, tzn. żeby były cechami obiektów opisywanych przez te bloki.

FPB - formaty drukowania PB

typ: integer

kod tablicy: 19

	-4	-3		NCPB
1				
i				
NBT				

NBT - liczba różnych typów bloków

NCPB - liczba kolumn PB

NCPB $\leq$ 99

Tablica FPB zawiera formaty drukowania elementów tablicy PB.

Blokowi typu i odpowiada i -ty wiersz tablicy FPB. Znaczenie poszczególnych kolumn dla i -tego wiersza jest następujące:

- 4. kod 1-go znaku nazwy procedury opisującej blok typu i
- 3. kod 2-go znaku nazwy procedury opisującej blok typu i
- ...
- 0. kod 5-go znaku nazwy procedury opisującej blok typu i
- 1. format drukowania parametru znajdującego się w 1-szej kolumnie PB
- 2. format drukowania parametru znajdującego się w 2-ej kolumnie PB
- ...
- NCPB. format drukowania parametru znajdującego się w NCPB-tej kolumnie PB

Jeżeli nazwa procedury ma mniej niż 5 znaków to pozostałe miejsca przeznaczone na kody są wypełniane przez program kodami spacji. Spacje i nowe linie wewnątrz nazwy procedury nie są uwzględniane przez program /pomija on je przy czytaniu nazwy w danych/. Dla różnych bloków tego samego typu trzeba przyjąć wspólne formaty drukowania wartości parametrów.

APB - dodatkowe parametry bloków

typ: real

kod tablicy: 20

	1	2		NCAP
1				
i				
NRAP				

NRAP - liczba wierszy APB

NCAP - liczba kolumn APB

NCAP $\leq$ 99

W tablicy APB znajdują się dodatkowe parametry bloków. Jeden wiersz tej tablicy odpowiada jednemu blokowi. Znaczenie poszczególnych kolumn dla i-tego wiersza jest następujące:

1. nr bloku wewnętrzny w programie, a zewnętrzny w danych
2. } parametry bloku
- NCAP. }

Wiersze tablicy APB są uporządkowane wg numerów bloków tzn. wiersz odpowiadający blokowi o mniejszym numerze poprzedza wiersz odpowiadający blokowi o większym numerze. Zaleca się aby parametry bloków znajdujące się w tej tablicy nie odnosiły się do rzeczywistych własności aparatów lub zachodzących w nich procesów lecz do funkcjonowania procedury opisującej dany blok.

FAP - formaty drukowania APB

typ: integer

kod tablicy: 21

	1	2		NCAP
1				
i				
NRFA				

NRFA - liczba wierszy FAP / liczba różnych typów bloków opisanych w APB /

NCAP - liczba kolumn APB

Tablica FAP zawiera formaty drukowania elementów tablicy APB. Jednemu wierszowi odpowiadają bloki określonego tego samego typu. Znaczenie poszczególnych kolumn dla i-tego wiersza jest następujące:

1. typ bloku
2. format drukowania parametru znajdującego się w 2-ej kolumnie APB
3. format drukowania parametru znajdującego się w 3-ej kolumnie APB
- ⋮
- NCAP. format drukowania parametru znajdującego się w NCAP-tej kolumnie APB

Wiersze tablicy FAP są uporządkowane wg numerów typów bloków tzn. dla wierszy I, J takich, że $I < J$ mamy $FAP[I,1] < FAP[J,1]$. Dla różnych bloków tego samego typu trzeba przyjąć wspólne formaty drukowania wartości parametrów.

EP - parametry ekonomiczne bloków aparaturowych

typ: real

kod tablicy: 22

	1	2
1		
i		
NAB		

NAB - liczba bloków aparaturowych

Blokowi aparaturowemu o numerze wewnętrznym i odpowiada i-ty wiersz tablicy EP. Znaczenie poszczególnych kolumn dla tego wiersza jest następujące:

1. koszty eksploatacyjne aparatu opisanego przez blok
2. koszty inwestycyjne aparatu opisanego przez blok

LV - ograniczenia wartości cech bloków

typ: real

kod tablicy: 23

	1	2		NCLV
1				
i				
NRLV				

NRLV - liczba wierszy LV

NCLV - liczba wierszy LV

Tablica LV zawiera ograniczenia wartości cech bloków. Ma ona między innymi zastosowanie w blokach optymalizacyjnych i blokach aparaturowych przy projektowaniu. Jeden wiersz tablicy odpowiada jednemu blokowi. Wiersze są uporządkowane wg numerów bloków tzn. wiersze odpowiadające blokom o mniejszych numerach poprzedzają wiersze odpowiadające blokom o większych numerach. Najmniejsza wartość ograniczenia dolnego jaką można przyjąć jest równa $- 5,6 : 10^{76}$. Największa wartość ograniczenia górnego jest natomiast $5,6 : 10^{76}$. Znaczenie poszczególnych kolumn dla i-tego wiersza odnoszącego się do bloku aparaturowego scharakteryzowanego przez k cech jest następujące:

1. nr bloku / wewnętrzny w programie, a zewnętrzny w danych /
2. ograniczenie dolne cechy znajdującej się w 1-szej kolumnie PB
3. ograniczenie górne cechy znajdującej się w 1-szej kolumnie PB
4. ograniczenie dolne cechy znajdującej się w 2-ej kolumnie PB
5. ograniczenie górne cechy znajdującej się w 2-ej kolumnie PB
- ⋮
- 2k. ograniczenie dolne cechy znajdującej się w k-tej kolumnie PB
- 2k+1. ograniczenie górne cechy znajdującej się w k-tej kolumnie PB
- $\left. \begin{array}{l} 2k+2. \\ \vdots \\ NCLV. \end{array} \right\} \text{ zera, jeżeli } 2k+1 < NCLV$

Znaczenia poszczególnych kolumn dla wiersza odpowiadającego blokowi optymalizacyjnemu, który steruje k zmiennymi decyzyjnymi /są one cechami obiektów lub strumieni/ jest takie same jak dla wiersza odnoszącego się do bloku aparaturowego z tym, że w odpowiednich kolumnach tablicy PB /w wierszu odpowiadającym blokowi optymalizacyjnemu/ nie znajdują się cechy ale kody zmiennych decyzyjnych, dla których podane są te ograniczenia. Tak więc w podanym wyżej opisie kolumn tablicy LV tekst "ograniczenie dolne /górne/ cechy znajdujące się w j-tej kolumnie PB" trzeba w tym drugim przypadku zastąpić tekstem: ograniczenie dolne /górne/ zmiennej decyzyjnej, której kod znajduje się w j-tej kolumnie PB.

ME - pamięć

typ: real

kod tablicy: 24

	1	2		NCME
1				
i				
NRME				

NRME - liczba wierszy ME

NCME - liczba kolumn ME

Tablica ME znajduje następujące zastosowania w blokach optymalizacyjnych

1. zapamiętuje poszczególne kroki iteracyjne tzn. wartości zmiennych decyzyjnych i wartość funkcji celu, przy czym jeden wiersz odpowiada jednej iteracji, a kolumnom blok optymalizacyjny przyporządkowuje zmienne decyzyjne i wartość funkcji celu.
2. zapamiętuje wyniki uzyskane na poszczególnych etapach przy programowaniu dynamicznym

Liczba wierszy tablicy jest równa największej z trzech następujących liczb:

- a/ maksymalnej liczbie iteracji potrzebnych do określenia następnego przybliżenia przez dowolny blok optymalizacyjny zadania
- b/ maksymalnej liczbie iteracji jaką chcemy na raz zapamiętać w ME
- c/ maksymalnej liczbie wierszy jaka jest potrzebna przy programowaniu dynamicznych

Liczba kolumn tablicy jest równa większej z dwu liczb

- a/ maksymalnej liczbie zmiennych decyzyjnych występujących w zadaniach zwiększonej o 1
- b/ maksymalnej liczbie kolumn jaka jest potrzebna przy programowaniu dynamicznych

Tablica ME może być użytkowana przez kilka bloków optymalizacji wtedy gdy mamy do czynienia z:

1. kilkoma zadaniami optymalizacji ale takimi, że obliczenia każdego z nich nie wpływają na wyniki uzyskiwane przez pozostałe /nie ma przepływu informacji od jakiegoś obiektu ze zbioru rozszerzonego jednego zadania do pewnego obiektu ze zbioru fundamentalnego drugiego zadania/. Każdy z bloków optymalizacji korzysta wtedy z ME gdy poprzedni zakończył obliczenia z tym, że przed przystąpieniem do obliczeń musi wyzerować ME.
2. dwupoziomowym zadaniem optymalizacji. Wtedy zadanie nadrzędne i każde z zadań lokalnych musi mieć przydzielone odpowiednie kolumny w ME i żaden z bloków optymalizacyjnych wykonujących obliczenia tych zadań nie może zmieniać zapisów wykonanych w ME przez inne bloki. Przed rozpoczęciem obliczeń zadania dwupoziomowego blok optymalizacyjny zadania nadrzędnego musi wyzerować ME.

Niech zadanie optymalizacji ma 1 zmiennych decyzyjnych. Znaczenia poszczególnych kolumn ME /wyłączywszy przypadek programowania dynamicznego/ są następujące:

1. zmienna decyzyjna, której kod znajduje się w 1-szej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu
2. zmienna decyzyjna, której kod znajduje się w 2-jej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu
- ⋮
1. zmienna decyzyjna, której kod znajduje się w 1-tej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu
- 1+1. wartość funkcji celu, której kod znajduje się w 1+1-jej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu

Niech zadanie dwupoziomowe ma k zadań lokalnych. Niech zadanie nadrzędne ma l_0 zmiennych decyzyjnych, a zadania lokalne odpowiednio $l_1, l_2, \dots, l_k$ zmiennych decyzyjnych. Znaczenia poszczególnych kolumn ME / wyłączwszy przypadek programowania dynamicznego / są wtedy następujące:

1. zmienna decyzyjna, której kod znajduje się w 1-szej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu zadania nadrzędnego
- $\vdots$
- l_0 . zmienna decyzyjna, której kod znajduje się w l_0 -tej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu zadania nadrzędnego
- l_0+1 . wartość funkcji celu zadania dwupoziomowego, której kod znajduje się w l_0+1 -ej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu zadania nadrzędnego
- l_0+2 . zmienna decyzyjna, której kod znajduje się w 1-szej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu 1-go zadania lokalnego
- $\vdots$
- l_0+l_1+1 . zmienna decyzyjna, której kod znajduje się w l_1 -ej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu 1-go zadania lokalnego
- l_0+l_1+2 . wartość funkcji celu 1-go zadania lokalnego, której kod znajduje się w l_1+1 -ej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu 1-go zadania lokalnego
- $\vdots$
- $\sum_{i=0}^{k-1} (l_i+1)+1$ zmienna decyzyjna, której kod znajduje się w 1-tej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu k -tego zadania lokalnego
- $\vdots$

$\sum_{i=0}^{k-1} (l_i+1)+l_k$. zmienna decyzyjna, której kod znajduje się w l_k -tej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu k-tego zadania lokalnego

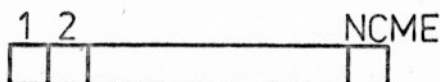
$\sum_{i=0}^k (l_i+1)$. wartość funkcji celu k-tego zadania lokalnego, której kod znajduje się w l_k+1 -ej kolumnie PB w wierszu odpowiadającym blokowi optymalizacyjnemu k-tego zadania lokalnego

Jeżeli blok optymalizacyjny zapisuje w ME informacje potrzebne do określenia następnego przybliżenia zmiennych decyzyjnych / na podstawie większej od 1 liczby iteracji / to nowe przybliżenie wpisuje do pierwszego wolnego wiersza, a następnie przesuwając wszystkie wypełnione wiersze w górę wyrzucając wiersz pionowy i zerując wiersz gdzie poprzednio zapisane było ostatnie przybliżenie. Jeżeli natomiast blok zapisuje w ME kolejne iteracje to z chwilą gdy ME zostanie wypełniona, przepisuje ją do zbioru STORE /patrz opis tego zbioru/, zeruje i kolejne iteracje zapamiętuje od góry. Możliwe są też inne sposoby użytkowania ME np. do pamiętania iteracji a jednocześnie blok może określać nowe przybliżenia na podstawie pewnej liczby poprzednich.

FME - formaty drukowania ME

typ: integer

kod tablicy: 25



NCME - liczba kolumn ME

Tablica FME zawiera formaty drukowania elementów tablicy ME, przy czym wszystkie elementy występujące w jednej kolumnie ME są drukowane w tym samym formacie. Znaczenie poszczególnych elementów tablicy FME jest następujące:

1. format drukowania 1-szej kolumny ME
2. format drukowania 2-ej kolumny ME
- ⋮
- NCME. format drukowania NCME-tej kolumny ME

PCP - parametry fizyko-chemiczne

typ: real

kod tablicy: 26

	1	2		NCPC
1				
i				
NRPC				

NRPC - liczba wierszy PCP

NCPC - liczba kolumn PCP

NCPC $\leq$ 60

Tablica PCP zawiera parametry do obliczania własności fizyko-chemicznych substancji. Jednej substancji odpowiada jeden wiersz tablicy, przy czym w pierwszej kolumnie znajduje się numer tej

substancji. Prace nad ustaleniem znaczenia pozostałych kolumn i procedurami obliczania własności fizyko-chemicznych są prowadzone przez mgr inż. St. Wasylkiewicza.

3. TABLICE WYSTĘPUJĄCE PRZY CZYTANIU DANYCH

* SNISTST - opis strumieni

	1	2			1	2		
1				*				*
				*				*
				⋮				⋮
				*				*
i				x				*
				x				*
				⋮				⋮
				*				*
				⋮				⋮
				*				*
NS				*				*
	A			B	C		D	

NS - liczba strumieni w systemie

Tablica * SNISTST zawiera dane dotyczące strumieni. Jeden wiersz tej tablicy odpowiada jednemu strumieniowi. Niech w i -tym wierszu opisany będzie strumień o k fazach i l składnikach. Znaczenie poszczególnych kolumn dla tego wiersza jest następujące:

A. 1. numer zewnętrzny strumienia

2. }
 : } identyfikator strumienia
 : }
 l+k+5. }

B. *

C. 1. }
 : } charakterystyka strumienia
 : }
 k(l+1)+2. }

D. *

Za nieznane cechy strumieni wstawiamy zera. Zera znajdujące się pomiędzy gwiazdką, a pierwszą liczbą większą od zera leżącą na lewo od gwiazdki, można pominąć zarówno w grupie kolumn A jak i w grupie kolumn C. Jeżeli w pewnym wierszu charakterystyka strumienia jest w ogóle nie znana to dwie gwiazdki będą stały obok siebie. Kolejność wierszy w tablicy jest dowolna.

* IIV - identyfikacja cech rozerwanych / iterowanych /

	1	2		
1				*
				*
				...
				*
i				*
				*
				...
				*
NTS				*

NTS - liczba strumieni zawierających cechy rozerwane

Tablica *IIV zawiera numery kolumn tablicy ST w których występują cechy rozerwane, przy czym jeden wiersz tablicy odpowiada jednemu strumieniowi. Znaczenie poszczególnych kolumn dla i-tego wiersza, który zawiera k cech rozerwanych jest następujące:

1. nr zewnętrzny strumienia
2. } numery kolumn tablicy ST w których występują cechy rozer-
- : } wane danego strumienia, wypisane w dowolnej kolejności
- : }
- k+1. }
- k+2. *

Kolejność wierszy w tablicy jest dowolna

* BNPMPB - opis bloków

		1 2			1 2		
1		*			*		*
		*			*		*
		⋮			⋮		⋮
		*			*		*
i		*			*		*
		*			*		*
		⋮			⋮		⋮
		*			*		*
NB		*			*		*
	A		B		C		D
	E		F				

NB - liczba bloków

Tablica *BNPMPB zawiera opis struktury systemu i parametry bloków, przy czym jeden wiersz tablicy odpowiada jednemu blokowi. Znaczenie poszczególnych kolumn dla i-tego wiersza, któremu odpowiada blok mający k strumieni wejściowych i l strumieni wyjściowych jest następujące:

- A. nazwa procedury opisującej blok /można pominąć/
- B. *
- C. 1. nr zewnętrzny bloku
2. typ bloku / patrz opis procedury JUMPER /
3. nr zewnętrzny strumienia ze znakiem plus jeżeli jest to strumień wejściowy albo ze znakiem minus jeżeli jest to strumień wyjściowy
4. nr wejścia lub wyjścia bloku, które jest związane ze strumieniem z punktu 3 odpowiednio do jego znaku
5. nr zewnętrzny strumienia ze znakiem plus jeżeli jest to strumień wejściowy albo ze znakiem minus jeżeli jest to strumień wyjściowy

6. nr wejścia lub wyjścia bloku, które jest związane ze strumieniem z punktu 5 odpowiednio do jego znaku.

⋮

$2(k+1)+1$. nr zewnętrzny strumienia ze znakiem plus jeżeli jest to strumień wejściowy albo ze znakiem minus jeżeli jest to strumień wyjściowy

$2(k+1)+2$. nr wejścia lub wyjścia bloku, które jest związane ze strumieniem z punktu $2(k+1)+1$ odpowiednio do jego znaku

D. *

E. 1. } parametry opisujące blok. Znaczenie kolumn /parametrów/ jest
2. } dokładnie takie samo jak dla odpowiednich kolumn w PB
⋮ } dla tego bloku.

F. *

Za nieznane parametry wstawiamy zera. Zera znajdujące się pomiędzy gwiazdką, a pierwszą liczbą różną od zera leżącą na lewo od gwiazdki, można pominąć. Jeżeli w pewnym wierszu parametry bloku w ogóle nie są znane, to dwie gwiazdki będą stały obok siebie. Kolejność wierszy w tablicy jest dowolna. Nr zewnętrzny dowolnego bloku fikcyjnego musi być większy od nr zewnętrznego każdego bloku aparaturowego.

* FPB - formaty drukowania PB

NBT - liczba różnych typów bloków

		1	2				
1		*					*
		*					*
		⋮					⋮
		*					*
i		*					*
		⋮					⋮
		*					*
NBT		*					*
	A	B		C		D	

Tablica *FPB zawiera formaty drukowania elementów tablicy PB. Jeden wiersz tablicy odpowiada jednemu typowi bloku. Znaczenie poszczególnych kolumn jest następujące:

A. nazwa procedury opisującej blok

B. *

C. 1. typ bloku

2. format drukowania parametru znajdującego się w 1-szej kolumnie PB

3. format drukowania parametru znajdującego się w 2-ej kolumnie PB

⋮

NCPB+1. format drukowania parametru znajdującego się w NCPB-tej kolumnie PB

D. *

Zera znajdujące się pomiędzy gwiazdką, a pierwszą liczbą większą od zera leżącą na lewo od gwiazdki, można pominąć. Dla różnych bloków tego samego typu trzeba przyjąć wspólne formaty drukowania wartości parametrów. Kolejność wierszy w tablicy jest dowolna.

* APB - dodatkowe parametry bloków

			1	2		
1		*				*
		*				*
		⋮				⋮
		*				*
i		*				*
		⋮				⋮
		*				*
NRAP		*				*
	A	B	C		D	

NRAP - liczba wierszy APB

W tablicy *APB znajdują się dodatkowe parametry bloków. Jeden wiersz tej tablicy odpowiada jednemu blokowi. Znaczenie poszczególnych kolumn jest następujące:

- A. nazwa procedury opisującej blok /można pominąć/
- B. *
- C. 1. nr zewnętrzny bloku
 - 2. } parametry dodatkowe bloku. Znaczenie kolumn /paramet-
 - 3. } rów/ dokładnie takie samo jak dla odpowiednich ko-
 - } lumn APB dla tego bloku
- D. *

Za nieznane parametry wstawiamy zera. Zera znajdujące się pomiędzy gwiazdką a pierwszą liczbą różną od zera leżącą na lewo od gwiazdki, można pominąć. Jeżeli w pewnym wierszu parametry dodatkowe bloku w ogóle nie są znane, to dwie gwiazdki będą stały obok siebie. Kolejność wierszy w tablicy jest dowolna.

*FAP - formaty drukowania APB

		1	2		
1		*			*
		*			*
		...			...
		*			*
i		*			*
		*			*
		...			...
		*			*
NRFA		*			*
A		B	C		D

NRFA - liczba wierszy FAP / liczba różnych typów bloków opisanych w APB/.

Tablica *FAP zawiera formaty drukowania elementów tablicy APB. Jeden wiersz tablicy odpowiada jednemu typowi bloku. Znaczniki poszczególnych kolumn jest następujące:

A. nazwa procedury opisującej blok /można pominąć/

B. *

C. 1. typ bloku

2. format drukowania parametru znajdującego się w 2-ej kolumnie APB

3. format drukowania parametru znajdującego się w 3-ej kolumnie APB

...

NCAP. format drukowania parametru znajdującego się w NCAP-tej kolumnie APB

D. *

Zera znajdujące się pomiędzy gwiazdką, a pierwszą liczbą większą od zera leżącą na lewo od gwiazdki, można pominąć. Dla różnych bloków tego samego typu trzeba przyjąć wspólne formaty drukowa-

V. OPIS ZBIORÓW NA TAŚMACH MAGNETYCZNYCH

Wszystkie opisane niżej zbiory są zorganizowane przy pomocy pakietu procedur dla pamięci taśmowej opisanych w [4].

ZBIORY ADDATA, LIBDATA-1 i LIBDATA-2

nr programowy zbioru ADDATA: 20

nr programowy zbioru LIBDATA-1: 21

nr programowy zbioru LIBDATA-2: 22

Informacje są wprowadzane do tych trzech zbiorów przy pomocy programu ARCHIVIST, a do zbioru ADDATA także przy pomocy programu BOSS przed rozpoczęciem właściwych obliczeń / na etapie czytania danych przez procedurę READ DATA/. Są one później odczytywane przez bloki. Zbiór ADDATA zawiera informacje zależące od konkretnego zadania obliczeniowego i jest tworzony w zależności od potrzeb. Natomiast zbiory LIBDATA-1 i LIBDATA-2 zawierają informacje niezależące od konkretnego zadania obliczeniowego, np. normy. Rekordem danych będziemy nazywali rekord odpowiadający n -wymiarowej tablicy typu real, gdzie $n \leq 5$. Rekord wymiaru jest to rekord odpowiadający jednowymiarowej n -elementowej tablicy typu real, której elementy mają następujące znaczenie:

1. górna granica 1-go wskaźnika tablicy odpowiadającej rekordowi danych
2. górna granica 2-go wskaźnika tablicy odpowiadającej rekordowi danych
- ...
- n. górna granica n -tego wskaźnika tablicy odpowiadającej rekordowi danych

Dolne granice wszystkich wskaźników są równe 1. Rekordy w każdym ze zbiorów poukładane są w następujący sposób:

1. a/ rekord wymiaru

b/ rekord danych

2. a/ rekord wymiaru

b/ rekord danych

⋮

Rekord wymiaru zawsze musi poprzedzać związany z nim rekord danych. Zgodnie z wymaganiami pakietu procedur dla pamięci taśmowej rekordy wymiaru i danych muszą posiadać swoje nazwy.

Przyjmujemy następującą składnię tych nazw

$\langle nr \rangle ::= \langle \text{liczba naturalna} \rangle$

$\langle \text{nazwa rekordu danych} \rangle ::= \langle \text{nazwa} \rangle - \langle nr \rangle$

$\langle \text{nazwa rekordu wymiaru} \rangle ::= \text{DIM} - \langle \text{nazwa rekordu danych} \rangle$

Znaczenie powyższych symboli jest podane w [2]. Za $\langle \text{nazwa} \rangle$ będziemy z reguły przyjmować nazwę procedury opisującej blok /szczególnie w zbiorze ADDATA/, a tylko w uzasadnionych przypadkach gdy ten sam rekord będzie czytany przez kilka różnych procedur przyjmujemy jakąś wspólną nazwę związaną z zawartością rekordu np. dla norm. Jeżeli procedura czyta rekordy z dwóch czy trzech zbiorów to nazwy rekordów w różnych zbiorach mogą być takie same. W razie potrzeby procedura może także czytać rekordy przeznaczone początkowo dla innych procedur. Dla procedury czytającej tablicę ze zbioru ADDATA zaleca się możliwość alternatywnego czytania jej z co najmniej dwóch rekordów o różnych nazwach. Sterowanie czytania odpowiedniego rekordu odbywa się wtedy przez umieszczenie w tablicy APB odpowiedniej liczby równej $\langle nr \rangle$ lub zeru w przypadku gdy nie trzeba czytać danej tablicy /jeżeli oczywiście przewidujemy taką możliwość/.

Przykład: Procedura REAKTOR ma czytać ze zbioru ADDATA dwuwymiarową tablicę A albo z rekordu o nazwie REAKTOR-3 albo z rekordu REAKTOR-4. Oczywiście wcześniej w zależności od tego z jakiego rekordu będzie czytana tablica A trzeba wprowadzić górne granice jej wskaźników odpowiednio z rekordu DIM-REAKTOR-3 albo z rekordu DIM-REAKTOR-4. Sterowanie czytaniem ma odbywać się przez umieszczenie w wierszu tablicy APB odpowiadającym danemu blokowi w drugiej kolumnie liczby 3 albo 4. Procedura REAKTOR może wyglądać np. tak:

```
procedure REAKTOR ( KEY, ..., APB, ... );  
  :  
  begin  
    integer I, J;  
    array V [1:2];  
    switch SW:= E,F;  
    :  
    for :=1 step 1 until KEY [33] do if KEY[I,1] = APB[I,1]  
    then go to D;  
    D:J:= APB [ 1,2]-2;  
    go to SW[J];  
    E: read binary ( 20,V, ((' DIM-REAKTOR-3')));  
    go to G;  
    F: read binary ( 20,V, ((' DIM - REAKTOR-4')));  
    G: begin  
      array A[1: V [1], 1: V [2]];  
      switch SC:= K,L;  
      :  
      go to SC [J];  
      K: read binary ( 20,A, ((' REAKTOR-3')));  
      go to M;
```

```
      L: read binary ( 20,A,('REAKTOR-4'));  
      M:  
      :  
      end;  
      :  
      :  
      end REAKTOR;
```

Przyporządkowanie jednej tablicy kilku nazw rekordów ze zbioru ADDATA jest celowe, gdyż w systemie może występować kilka takich samych bloków i każdy z nich może wymagać innych wartości elementów tablicy lub dla pewnego bloku chcemy zmienić wartości elementów tablicy i przy tym nie chcemy powtórnie tworzyć zbioru ADDATA. Gdy procedura czyta dane ze zbioru LIBDATA-1 lub LIBDATA-2 to jednej tablicy przyporządkowujemy z reguły jedną nazwę rekordu. Jeżeli jednak chcemy alternatywnie czytać różne tablice lub jeżeli zachodzi konieczność przyporządkowania jednej tablicy kilku nazw rekordów to wtedy sterowanie czytania odpowiednich rekordów umieszczamy w APB tak jak się to robi dla zbioru ADDATA.

ZBIÓR STORE

nr programowy zbioru: 23

Do zbioru STORE podczas przebiegu programu BOSS mogą być zapisywane tablice SI przez bloki rozwiązywania układu równań lub tablice ME przez bloki optymalizacyjne i związane z tymi tablicami dodatkowe informacje. Wszystko to odbywa się w celu późniejszego wydrukowania tablic SI i ME przez procedurę PRINT DATA. Przez rekord wyników będziemy rozumieli rekord odpowiadający tablicy SI lub ME przy czym będzie miał on nazwę SI lub ME odpowiednio.

Rekord opisu o nazwie DSI odpowiada jednowymiarowej KEY [25]+5 elementowej tablicy typu real, której znaczenie poszczególnych elementów przy założeniu, że cechy rozerwane dla danego σ - podsystemu informacyjnego występują w m strumieniach, jest następujące:

1. nr wewnętrzny bloku rozwiązywania układu równań
2. maksymalny nr zajętego wiersza w tablicy SI
3. maksymalny nr zajętej kolumny w tablicy SI
4. nr iteracji /większy lub równy zero / zapisanej w pierwszym wierszu tablicy SI
5. liczba określająca co którą iterację należy drukować
6. nr wewnętrzny strumienia, którego cechy rozerwane zajmują 1-szą grupę kolumn w tablicy SI
7. nr wewnętrzny strumienia, którego cechy rozerwane zajmują 2-gą grupę kolumn w tablicy SI
- ...
- m+5. nr wewnętrzny strumienia, którego cechy rozerwane zajmują m-tą grupę kolumn w tablicy SI
- m+6. { zera, jeżeli $m < \text{KEY}[25]$
- ... }
- KEY[25]+5. }

Rekord opisu o nazwie DME odpowiada jednowymiarowej KEY [41]+5 elementowej tablicy typu real, której znaczenie poszczególnych elementów przy założeniu, że blok optymalizacyjny zapisuje informacje do pierwszych m kolumn tablicy ME, jest następujące:

1. nr wewnętrzny bloku optymalizacyjnego
2. maksymalny nr zajętego wiersza w tablicy ME
3. maksymalny nr zajętej kolumny w tablicy ME

4. nr iteracji /większy lub równy zero / zapisanej w pierwszym wierszu tablicy ME albo -1 przy programowaniu dynamicznym
 5. liczba określająca co którą iterację /lub co który wiersz przy programowaniu dynamicznym / należy drukować
 6. kod cechy strumienia lub parametru bloku przyporządkowanego 1-szej kolumnie tablicy ME
 7. kod cechy strumienia lub parametru bloku przyporządkowanego 2-jej kolumnie tablicy ME
 - ⋮
 - m+5. kod cechy strumienia lub parametru bloku przyporządkowanego m-tej kolumnie tablicy ME
 - m+6.
 - ⋮
 - KEY[41]+5.
- } zera, jeżeli $m < \text{KEY}[41]$

Jeżeli blok optymalizacyjny zapisuje w k-tej kolumnie ME wielkość nie występującą w ST, PB lub APB wówczas w tablicy odpowiadającej rekordowi opisu do k+5-tego elementu wstawiamy zero. Rekordy w zbiorze STORE są poukładane w następujący sposób:

1. a/ rekord opisu
 b/ rekord wyników
2. a/ rekord opisu
 b/ rekord wyników
- ⋮

Rekord opisu zawsze musi poprzedzać związany z nim rekord wyników. Jeden blok może oczywiście kilkakrotnie zapisywać tablice /odpowiednio SI albo ME wraz z tablicami odpowiadającymi rekordowi opisu/. Ma to miejsce wtedy gdy tablica zawiera całkowicie nową informację w porównaniu do tej, która została zapisana

wcześniej lub gdy blok zakończył obliczenia. W tym ostatnim przypadku, jeżeli tablica zawiera częściowo stare informacje, przed zapisaniem jej do zbioru STORE blok przesuwą nowe informacje tak aby zaczynały się od pierwszego wiersza, a pozostałe niewykorzystane wiersze wypełnia zerami. Wszystkie rekordy tego samego typu zapisywane do zbioru przez dowolne bloki rozwiązywania układu równań / odpowiednio optymalizacyjne / mają te same nazwy. Liczba par rekordów DSI i SI zapisanych do zbioru jest pamiętana w KEY[51], przy czym blok rozwiązywania układu równań po zapisie każdej takiej pary rekordów musi wykonać instrukcję KEY[51]:= KEY[51]+1. Podobnie liczba par rekordów DME i ME zapisanych do zbioru jest pamiętana w KEY[52], przy czym blok optymalizacyjny po zapisie każdej takiej pary rekordów musi wykonać instrukcję KEY[52]:= KEY[52]+1.

ZBIÓR PHCHP

nr programowy zbioru: 19

Zbiór PHCHP zawiera rekordy o nazwie PCP odpowiadające tablicy PCP[1:30, 1:60]. Kolumny o najwyższych numerach, których znaczenia nie zostaną ustalone będą tworzyły rezerwę na wypadek konieczności poszerzenia PCP o dodatkowe parametry. Wiersze tablicy PCP odpowiadającej danemu rekordowi są uporządkowane wg wzrastających numerów substancji. Rekordy w zbiorze są poukładane w ten sposób, że pierwszy rekord zawiera parametry substancji o numerach 1 ÷ 30 drugi o numerach 31 ÷ 60, trzeci 61 ÷ 90 itd.

VI. OMÓWIENIE TYPOWYCH PROCEDUR

Wszystkie omawiane tutaj procedury muszą być napisane w języku ALGOL 1900. Nie mogą one używać podstawowych urządzeń wejścia i wyjścia. Wystąpienie podczas wykonywania procedury nietypowego błędu tzn. nie opisanego w [1] procedura sygnalizuje zapisując nr błędu w KEY [7].

PROCEDURA OPISUJĄCA BLOK APARATUROWY

Nagłówek takiej procedury wygląda następująco

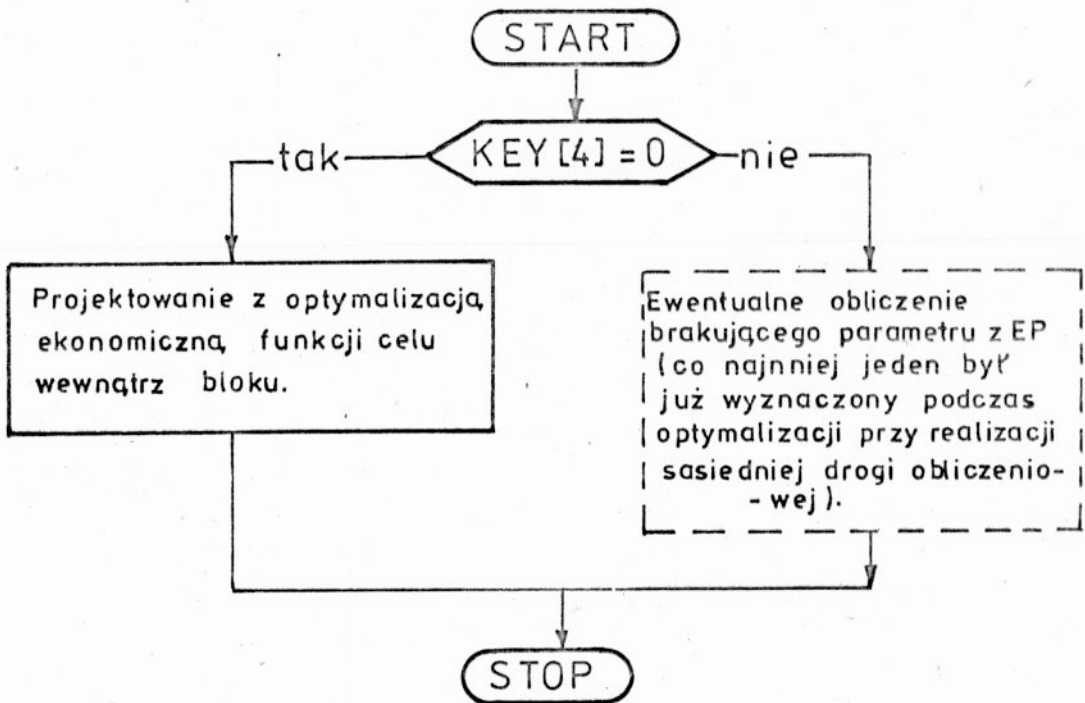
```
procedure nazwa ( KEY, IIS,IS,IOS, OS, LS, PB, APB,EP,LV, PCP);
```

```
    integer array KEY, IIS, IOS;
```

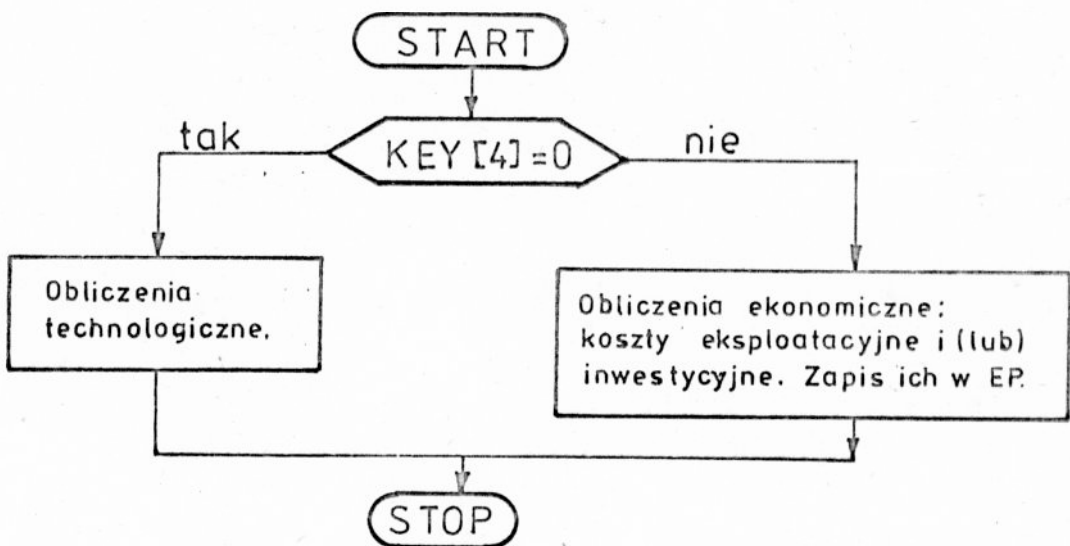
```
    array IS, OS, LS, PB, APB, EP, LV, PCP;
```

Procedura opisująca blok aparaturowy może zapisywać wyniki obliczeń do IS, OS oraz do wiersza w PB i APB odpowiadającego blokowi o numerze wewnętrznym KEY[1]. Procedura może czytać dane ze zbiorów ADDATA, LIBDATA-1 i LIBDATA-2. Na rysunkach VI.1 i VI.2 przedstawione są dwa spośród wielu możliwych, schematy blokowe procedur bloków aparaturowych.

Każda procedura bloku aparaturowego musi mieć dwie drogi obliczeń takie jak mają te z rysunków VI.1 i VI.2. Przypadek z rys. VI.1 odpowiada klasycznemu projektowaniu a z rys. VI.2 modelowaniu. Procedury bloku aparaturowego mogą realizować optymalizację wewnątrz bloku z tym, że zmiennymi decyzyjnymi nie mogą być cechy strumieni i cechy obiektów czyli zmiennymi decyzyjnymi nie mogą być elementy tablic IS, OS, PB i APB.



Rys. VI. 1



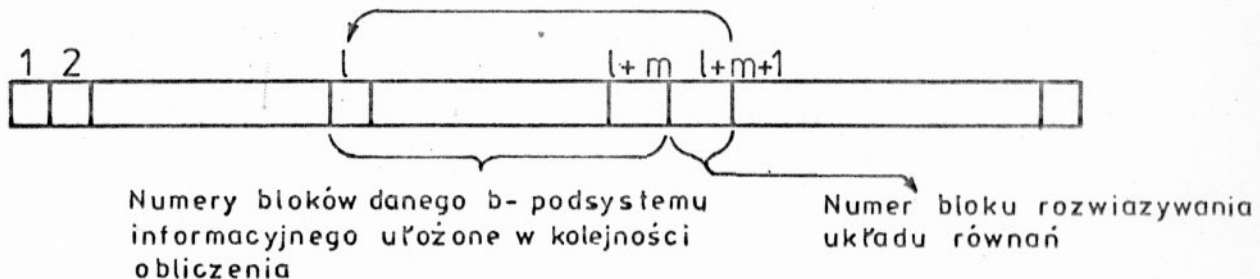
Rys. VI. 2

PROCEDURA OPISUJĄCA BLOK ROZWIĄZYWANIA UKŁADU RÓWNAŃ

Nagłówek procedury bloku rozwiązywania układu równań wygląda następująco

```
procedure nazwa (KEY, SN, IST, ST, IIV, LS, CE, ECE, SI, PB, APB);  
  integer array KEY, SN, IST, IIV;  
  array ST, LS, CE, ECE, SI, PB, APB;
```

Niech dla danego σ - podsystemu informacyjnego k strumieni zawiera cechy rozerwane. Numery zewnętrzne tych strumieni umieszczamy w wierszu tablicy PB odpowiadającym danemu blokowi rozwiązywania układu równań w kolumnach od 1 do k , zaś liczbę tych strumieni / jeżeli nie zlicza ich sama procedura / zapisujemy w 2-iej kolumnie w odpowiednim wierszu tablicy APB. Żadnych innych danych lub wyników związanych z blokiem rozwiązywania układu równań nie zaleca się zapisywać w PB. Numery rozerwanych cech strumieni znajdują się w IIV. Blok rozwiązywania układu równań musi być obliczany zaraz za ostatnim blokiem rozwiązywanego σ - podsystemu informacyjnego. W tablicy CS wygląda to następująco

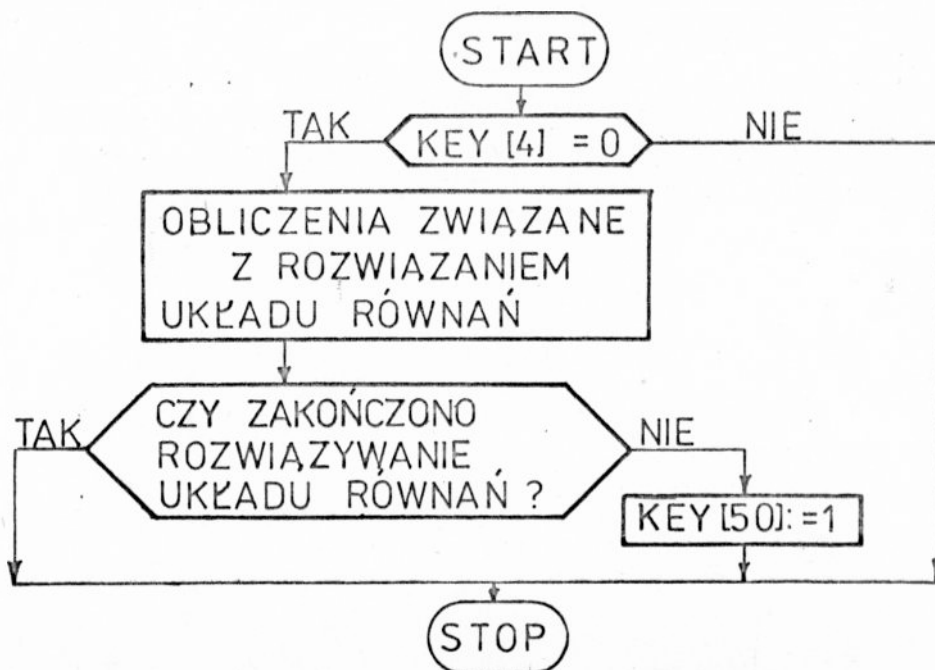


Jeżeli nie osiągnięto zadanej dokładności obliczeń, przy czym nie przekroczono jeszcze zadanej maksymalnej liczby iteracji

to procedura rozwiązywania układu równań może określić następne przybliżenie na podstawie pewnej liczby poprzednich iteracji i musi wykonać instrukcję $KEY[50] := 1$. Wtedy program organizacyjny posługując się tablicą NO przystąpi do powtórnego obliczenia danego σ -podsystemu informacyjnego zaczynając od bloku określonego przez NO.

Jeżeli procedura określa następne przybliżenie na podstawie poprzednich iteracji to dla danego strumienia suma tych jego ułamków molowych, które są cechami rozerwanymi nie może być większa od jedności pomniejszonej o sumę ułamków molowych tego strumienia, których wartości nie są obliczane przez żaden blok aparaturowy. Te ograniczające liczby umieszczamy w PB lub APB dla wszystkich strumieni zawierających cechy rozerwane.

Poza tym żadne cechy strumieni nie mogą przyjmować wartości ujemnych. Procedura może zapisywać tablice SI do zbioru STORE, a także jeżeli zachodzi taka potrzeba czytać rekordy ze zbioru ADDATA. Schemat blokowy procedury opisującej blok rozwiązywania układu równań jest przedstawiony na rys. VI.3.



PROCEDURA OPISUJĄCA BLOK OPTYMALIZACYJNY

Nagłówek procedury bloku optymalizacyjnego ma postać:

procedure nazwa (KEY,SN,IST,ST,BN,PB,APB,LV,ME);

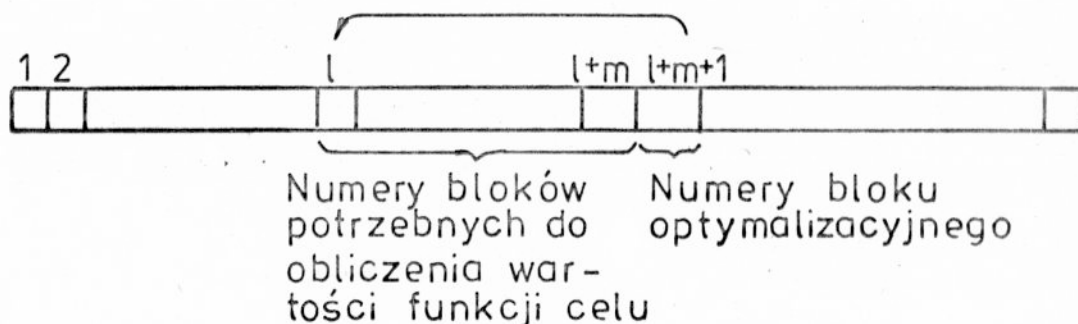
integer array KEY,SN,IST,BN;

array ST, PB,APB,LV,ME;

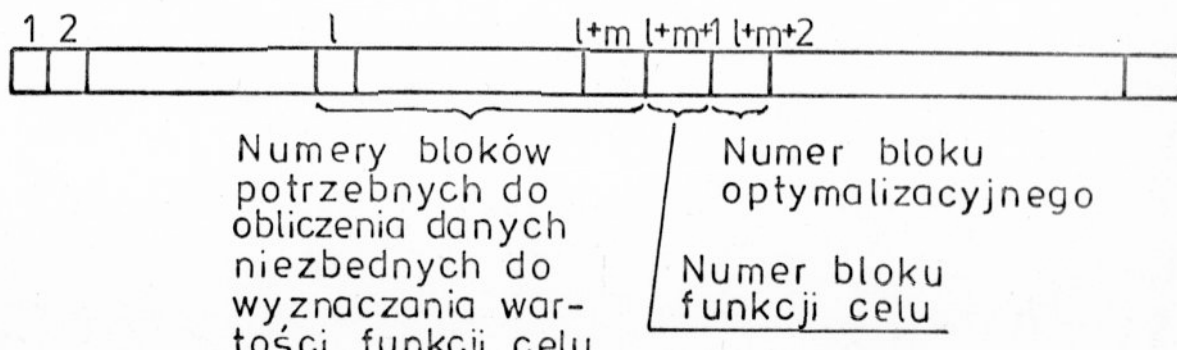
Zmiennymi decyzyjnymi zadania optymalizacji / patrz rozdział II punkt 3 / mogą być cechy strumieni lub obiektów, które nie są obliczane przez żaden blok, ale są wymagane jako dane. Wartość funkcji celu może być opisana przez wartość

1. cechy strumienia lub obiektu, która jest obliczona przez blok
2. parametru w PB lub APB, która jest liczona przez blok funkcji celu.

Ułożenie bloków w tablicy CS dla przypadku 1 jest następujące:

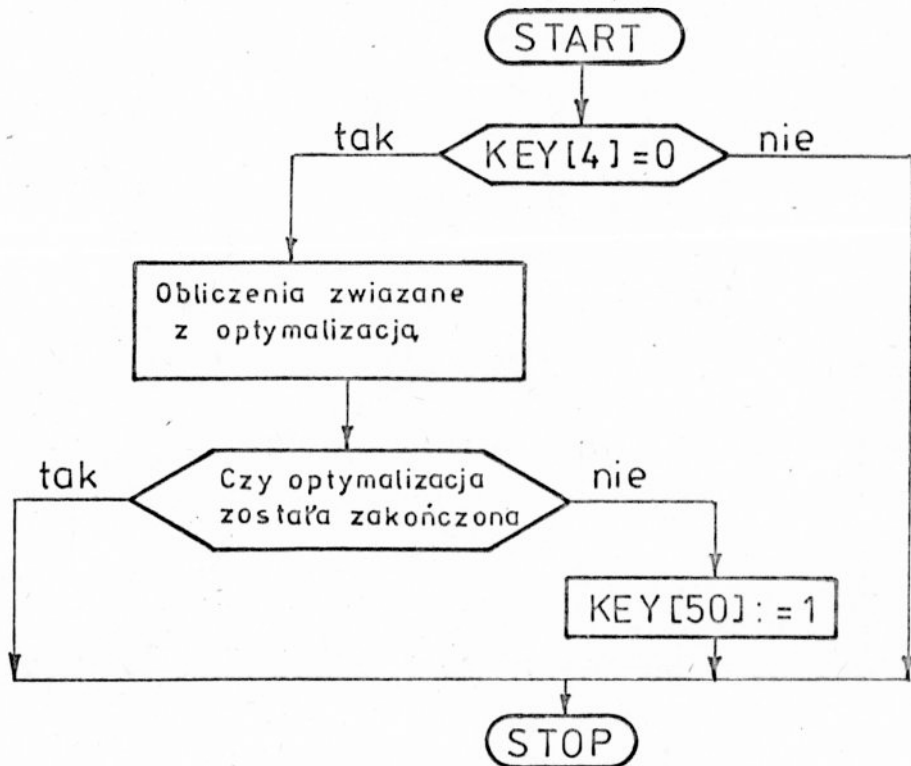


W przypadku 2 wygląda to następująco:



Przypadek 2 odnosi się między innymi do optymalizacji ekonomicznej, kiedy to do obliczenia wartości funkcji celu potrzebna jest znajomość kosztów eksploatacyjnych lub inwestycyjnych dla poszczególnych bloków aparaturowych. Wtedy rolę bloku funkcji celu pełni blok ekonomiczny. Niech dane zadanie optymalizacji ma k zmiennych decyzyjnych. Kody tych zmiennych decyzyjnych umieszczamy w wierszu tablicy PB odpowiadającym danemu blokowi optymalizacyjnemu w kolumnach od 1 do k , zaś kod funkcji celu zapisujemy w $k+1$ -szej kolumnie. Liczbę zmiennych decyzyjnych / jeżeli nie zlicza ich sama procedura / zapisujemy w 2-jej kolumnie w odpowiednim wierszu APB. Żadnych innych danych lub wyników związanych z blokiem optymalizacyjnym nie zaleca się zapisywać w PB. Jeżeli nie osiągnięto ekstremum funkcji celu, przy czym nie przekroczono jeszcze żądanej maksymalnej liczby iteracji to procedura musi określić nowe wartości zmiennych decyzyjnych i wykonać instrukcję `KEY [ 50 ] := 1`. Wtedy program organizacyjny posługując się tablicą NO przystąpi do powtórnego obliczania bloków uczestniczących w zadaniu optymalizacji zaczynając od bloku określonego przez NO. Procedura musi określać nowe przybliżenie w ten sposób, że suma zmiennych decyzyjnych będących ułamkami molowymi dla danego strumienia jest nie większa od jedności pomniejszonej o sumę ułamków molowych tego strumienia, których wartości nie są obliczone przez żaden blok aparaturowy i które nie są zmiennymi decyzyjnymi zadania optymalizacji związanego z danym blokiem optymalizacji. Te ograniczające liczby umieszczamy w PB lub APB dla wszystkich strumieni zawierających zmienne decyzyjne będące ułamkami molowymi. Procedura może zapisywać tablice ME do zbioru STORE, a także jeżeli zachodzi taka potrzeba czytać rekordy ze zbioru ADDATA. Jest wskazane przewi-

dzenie w procedurze optymalizacyjnej możliwości wyboru obliczania maksimum lub minimum funkcji celu. Schemat blokowy procedury opisującej blok optymalizacyjny jest przedstawiony na rys. VI.4.



Rys. VI.4

Uwaga: Wszystko to co powiedzieliśmy wyżej nie dotyczy optymalizacji wewnątrz bloku aparaturowego realizowanej przez sam blok aparaturowy.

PROCEDURA OPISUJĄCA BLOK EKONOMICZNY

Nagłówek procedury opisującej blok ekonomiczny ma postać

procedure nazwa (KEY,SN,IST,ST,PB,APB,EP);

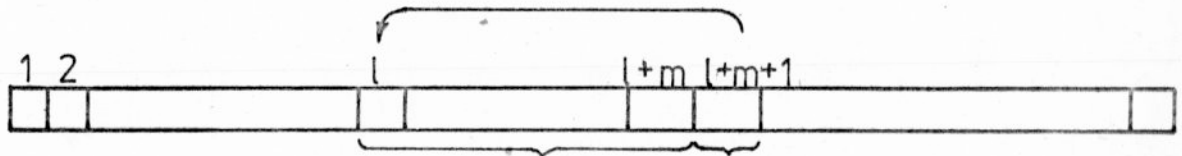
integer array KEY,SN,IST ;

array ST,PB,APB,EP;

Procedura taka steruje wykonywaniem obliczeń ekonomicznych przez bloki aparaturowe systemu w następujący sposób:

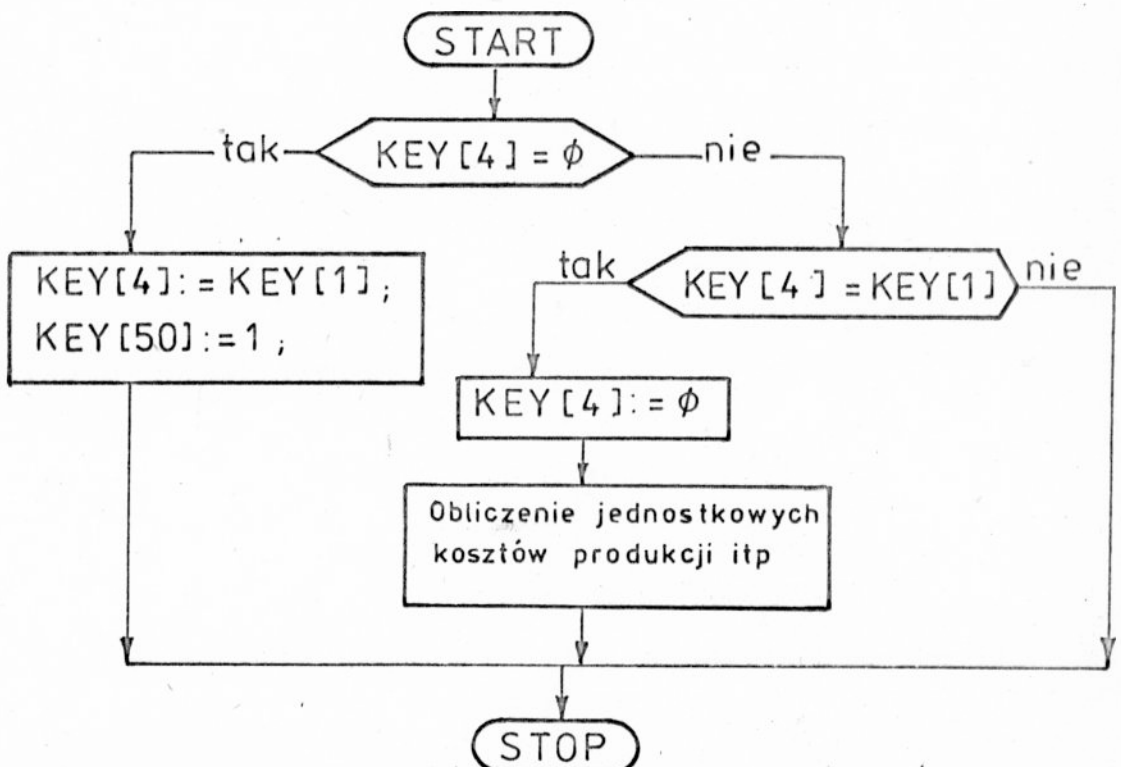
1. jeżeli $KEY[4] = 0$ to wykonuje instrukcje $KEY[4] := KEY[1]$ i $KEY[50] := 1$ a program organizacyjny posługując się tablicą NO przystąpi do obliczania bloków dla których mają być znane koszty eksploatacyjne i / lub / inwestycyjne poczynając od bloku określonego przez NO. Oczywiście obliczenia można przeprowadzić tylko dla bloków umieszczonych w tablicy CS przed danym blokiem ekonomicznym.
2. jeżeli $KEY[4] > 0$ i $KEY[4] = KEY[1]$ to znaczy, że bloki aparaturowe zakończyły obliczenia ekonomiczne sterowane przez dany blok ekonomiczny / o numerze wewnętrznym $KEY[1]$ /. Teraz trzeba wykonać instrukcję $KEY[4] := 0$ i dalej, jeżeli potrzeba, można obliczyć np. koszty jednostkowe wytwarzanych produktów. Potrzebne do tego wartości molowych natężeń przepływu surowców i produktów można odczytać z ST posługując się numerami zewnętrznymi strumieni wejściowych i wyjściowych systemu umieszczonymi w wierszu tablicy PB lub APB odpowiadającym danemu blokowi ekonomicznemu. Podobnie można liczyć koszty wytwarzania półproduktów wewnątrz systemu przez jakiś jego podsystem z tym, że wtedy w PB lub APB umieszczamy numery zewnętrzne strumieni wejściowych i wyjściowych rozpatrywanego podsystemu a poza tym potrzebna jest znajomość kosztów wpływających do niego surowców i wcześniej wytwarzanych półproduktów.
3. jeżeli $KEY[4] > 0$ i $KEY[4] \neq KEY[1]$ wtedy nie wykonuje się żadnych obliczeń przekazując sterowanie od razu do programu organizacyjnego.

Ułożenie bloku ekonomicznego i sterowanych przez niego bloków jest w tablicy CS następujące:



BŁOKI APARATUROWE
KTÓRYMI STERUJE BŁOK
EKONOMICZNY UMIE -
SZCZONY W l+m+1-SZYM
ELEMENTE TABLICY CS
I ZWIAZANE Z NIMI
BŁOKI FIKCYJNE

Procedura opisująca blok ekonomiczny może czytać rekordy ze zbioru ADDATA. Jej schemat blokowy jest przedstawiony na rys. VI.5.



Rys. VI.5

PROCEDURA OPISUJĄCA BLOK FUNKCJI CELU

Uwaga: W przypadku optymalizacji ekonomicznej rolę bloku funkcji celu pełni blok ekonomiczny / patrz str 116 / .

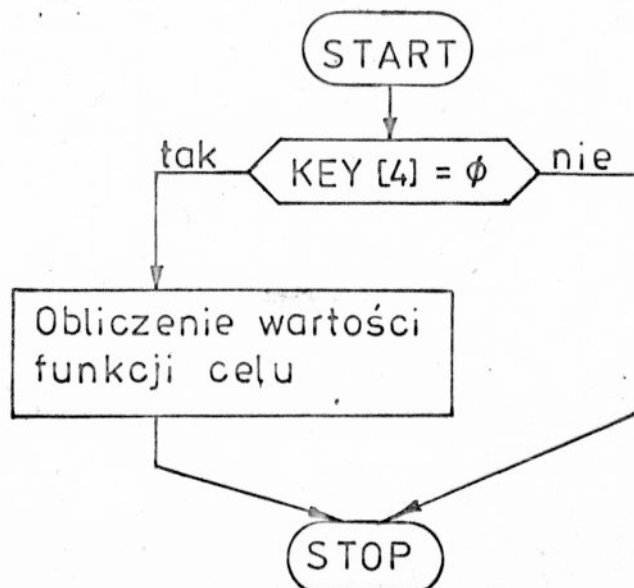
Nagłówek procedury opisującej blok funkcji celu ma postać

procedure nazwa (KEY,SN,ST,BN,PB,APB);

integer array KEY,SN,BN;

array ST,PB,APB;

Procedura oblicza wartość funkcji celu na podstawie zmiennych funkcji celu / patrz rozdział II punkt 3/. Niech funkcja celu ma k zmiennych. Kody tych zmiennych umieszczamy w wierszu tablicy PB odpowiadającym danemu blokowi funkcji celu w kolumnach od 1 do k . Żadnych innych danych lub wyników nie zaleca się zapisywać w PB. Liczbę zmiennych funkcji celu / jeżeli nie zlicza ich sama procedura / zapisujemy w 2-jej kolumnie w odpowiednim wierszu APB. Natomiast w 3-jej kolumnie tego wiersza procedura umieszcza obliczoną wartość funkcji celu. Schemat blokowy procedury opisującej blok funkcji celu przedstawiono na rys. VI.6.



Rys. VI. 6

PROCEDURA JUMPER

W procedurze JUMPER są zadeklarowane i wywoływane wszystkie procedury opisujące bloki aparaturowe i fikcyjne potrzebne do obliczenia danego systemu. Program organizacyjny po ustaleniu jak blok ma być aktualnie obliczany i przygotowaniu dla niego danych wywołuje, za pośrednictwem procedury JUMPER, procedurę opisującą dany blok.

Nagłówek procedury JUMPER jest postaci:

```
procedure JUMPER ( KEY,SN,IST,ST,IIS,IS,IOS,OS,IIV,LS,CE,ECE,SI,  
                    BN,PM,PB,APB,EP,LV,ME,PCP );  
integer array KEY,SN,IST,IIS,IOS,IIV,BN;  
array ST,IS,OS,LS,CE,ECE,SI,PB,APB,EP,LV,ME,PCP;
```

Opisy procedur bloków będą z reguły znajdować się w zbiorze bibliotecznym w postaci półskompilowanej. Jeżeli pewne opisy procedur dane są na kartach w postaci źródłowej to można je zamieścić bezpośrednio w treści procedury JUMPER lub poza nią między wierszami sterującymi continue i finish. Ten drugi sposób / tzn. poza procedurą JUMPER / jest wygodniejszy gdy treść danej procedury chcemy umieścić w nakładce. Wtedy przy określeniu struktury nakładkowej programu nakładanego za pomocą wierszy sterujących overlay, identyfikujemy procedurę bloku poprzez jej nazwę. We wszystkich przypadkach w których opis procedury bloku nie występuje wewnątrz procedury JUMPER umieszczamy tam zamiast niego nagłówek procedury zakończony symbolem podstawowym algot. W treści procedury JUMPER musimy także zamieścić opis przełącznika za pomocą którego będziemy się odwoływali do instrukcji procedur bloków. W związku z tym każda instrukcja procedury bloku musi być

poprzedzona odpowiednią etykietą. Wyrażeniem wskaźnikowym przełącznika / patrz [2] str. 21, 22 / musi być element KEY [5]. Przez typ bloku będziemy rozumieli taką wartość wyrażenia wskaźnikowego jaka jest potrzebna aby odwołać się do instrukcji procedury opisującej dany blok. Parametry formalne procedury JUMPER pełnią rolę parametrów aktualnych procedur bloków. Po każdej instrukcji procedury bloku z wyjątkiem ostatniej umieszczamy instrukcję skoku do etykiety poprzedzającej ostatni symbol podstawowy end w opisie procedury JUMPER.

Przykład

```
procedure JUMPER ( parametry formalne );  
    zbiór specyfikacji  
    beign  
        switch SDE := E1, E2, E3;  
        procedure REAKTOR ( parametry formalne );  
            zbiór specyfikacji  
            algol;  
        procedure WYMIENNIK CIEP ( parametry formalne );  
            zbiór specyfikacji  
            algol;  
        procedure KOLUMNA ( parametry formalne );  
            zbiór specyfikacji  
            algol;  
        go to SDE [ KEY [ 5 ] ];  
  
E1: REAKTOR ( parametry aktualne );  
    go to EK;
```

E2: WYMIENNIK CIEP (parametry aktualne);

go to EK;

E3: KOLUMNA (parametry aktualne);

EK: end JUMPER:

Aby można było wykonać instrukcję procedury REAKTOR wyrażenie wskaźnikowe musi przyjąć wartość 1. Tak więc wszystkie bloki obliczane przy pomocy procedury REAKTOR są typu 1. Podobnie wszystkie bloki obliczane przy pomocy procedury WYMIENNIK CIEP są typu 2, a wszystkie bloki obliczone przy pomocy procedury KOLUMNA są typu 3.

Uwaga 1: Zbiór specyfikacji zdefiniowany jest w [2] na str. 60, 61 i 155.

Uwaga 2: Stwierdzono praktycznie, że nie jest możliwe dołączenie do programu organizacyjnego BOSS procedur bloków napisanych w języku FORTRAN 1900.

VII. NAKŁADANIE PROCEDUR W PROGRAMIE ORGANIZACYJNYM

Programy obliczania systemów technologii chemicznej wymagają z reguły dużej pamięci operacyjnej komputera. Często może się zdarzyć, że pamięć, którą dysponujemy jest niewystarczająca. Metodą, która pozwala nam na ominięcie tej przeszkody jest nakładanie programu. Podstawowe wiadomości na ten temat są podane w [1,39].

Do kompilacji nakładanego programu organizacyjnego BOSS nadaje się jedynie kompilator #XALM. Nakładkami programu mogą być procedury READ KEY, READ DATA, PRINT DATA i procedury opisujące bloki. Struktura nakładkowa programu zależy od struktury obliczonego systemu. Powinna ona być tak określona aby liczba transmisji nakładek z pamięci zewnętrznej do jednostki centralnej była możliwie najmniejsza. Dlatego też trzeba ją tworzyć na nowo dla każdego konkretnego przypadku. Oczywiście nie należy zapominać, że przed przystąpieniem do obliczeń trzeba wczytać dane za pomocą procedur READ KEY i READ DATA, a po ich zakończeniu wydrukować wyniki używając procedury PRINT DATA. Te trzy procedury mogą tworzyć nakładki przyporządkowane temu samemu polu nakładania. /Uwaga - procedury READ KEY i READ DATA powinny wchodzić razem w skład tej samej nakładki/. Do tego pola wskazane jest również przypisanie części lub wszystkich /zależnie od konkretnej sytuacji/ nakładek z procedurami bloków gdyż podczas właściwych obliczeń ciągu technologii chemicznej nie korzysta się z procedur READ KEY, READ DATA i PRINT DATA. Natomiast w skład rezydenta muszą każdorazowo wejść procedury READ BINARY, WRITE BINARY, DATA SKIP, REWIND, FREEMT i READ. Osiągniemy to przez wymienienie ich w wierszu sterującym permanent.

Przykład

run

semicompiled (MT, LIBRAPROGORG, PROGORG)

library (MT,.BLOKI)

leaders

blocksizes

overlay program (BOSS)

overlay READ KEY, READ DATA: PRINT DATA: REAKTOR: WYMIENNIK
CIEP: KOLUMNA;

permanent READ BINARY, WRITE BINARY, DATA SKIP, REWIND,
FREEMT, READ;

priority 80

continue

segment główny programu

opis procedury JUMPER /patrz przykład z rozdziału VI/

opis procedury KOLUMNA

finish

Procedury READ KEY, READ DATA i PRINT DATA są zapamiętane w postaci półskompilowanej w podzbiorze PROGORG zbioru LIBRAPROGORG. Natomiast w podzbiorze BLOKI tego samego zbioru znajdują się procedury REAKTOR i WYMIENNIK CIEP również w postaci półskompilowanej. Opis procedury KOLUMNA w postaci źródłowej znajduje się na kartach i jest umieszczony poza procedurą JUMPER. Struktura nakładkowa programu jest następująca:

nakładka 1 pola 1 - procedura KOLUMNA

nakładka 2 pola 1 - procedura WYMIENNIK CIEP

nakładka 3 pola 1 - procedura REAKTOR

nakładka 4 pola 1 - procedura PRINT DATA

nakładka 5 pola 1 - procedury READ DATA i READ KEY

VIII. CZYTANIE DANYCH PRZEZ PROGRAM ORGANIZACYJNY

W rozdziale tym będziemy się powoływać na:

Uwaga 1: Końcowe nieokreślone lub równe zeru elementy tablicy można pominąć; program wstawi tam zera. Tablicę należy zakończyć *. Gwiazdka nie kończy liczby i dlatego między ostatnią cyfrą liczby a gwiazdką muszą wystąpić albo dwie spacje albo średnik albo przecinek albo nowa linia.

Uwaga 2: Każdy wiersz tablicy musi być zakończony *. Końcowe elementy każdego wiersza, które są nieokreślone lub równe zeru można pominąć; program wstawi tam zera. Kolejność wczytywania wierszy jest dowolna. Ponieważ gwiazdka nie kończy liczby, więc między ostatnią cyfrą liczby a gwiazdką muszą wystąpić albo dwie spacje albo przecinek albo średnik albo nowa linia.

Kolejność czytania danych przez program organizacyjny BOSS jest następująca:

1. czy procedury opisujące bloki potrzebują do obliczeń dane ze zbioru ADDATA

0 - nie

1 - tak przy czym nie trzeba wczytywać danych do zbioru

2- tak przy czym trzeba utworzyć zbiór i wczytać dane do niego

3 - tak przy czym trzeba wczytać dane do zbioru

2. czy procedury opisujące bloki potrzebują do obliczeń dane ze zbioru LIBDATA - 1

0 - nie

1 - tak

3. czy procedury opisujące bloki potrzebują do obliczeń dane ze zbioru LIBDATA-2

0 - nie

1 - tak

4. liczba elementów tablicy NO; jeżeli tablica NO nie występuje to wtedy zero.
5. liczba strumieni systemu
6. maksymalna liczba elementów w pojedynczym wierszu tablicy «SNISTST dla grupy kolumn A
7. maksymalna liczba cech strumienia w systemie
8. maksymalna liczba strumieni wpływających do pojedynczego bloku
9. maksymalna liczba strumieni wypływających z pojedynczego bloku
10. liczba wierszy tablicy «IIV; jeżeli tablica «IIV nie występuje to wtedy zero.

Jeżeli tablica «IIV występuje to

11. liczba kolumn tablicy «IIV

12. liczba wierszy tablicy LS; jeżeli tablica LS nie występuje to wtedy zero

Jeżeli tablica LS występuje to

13. liczba kolumn tablicy LS

14. liczba wierszy tablicy CE; jeżeli tablica CE nie występuje to wtedy zero
15. liczba wierszy tablicy ECE; jeżeli tablica ECE nie występuje to wtedy zero

Jeżeli tablica ECE występuje to

- 16. liczba kolumn tablicy ECE
- 17. liczba wierszy tablicy SI; jeżeli tablica SI nie występuje to wtedy zero

Jeżeli tablica SI występuje to

- 18. liczba kolumn tablicy SI
- 19. czy drukować tablicę SI
 - 0 - nie
 - 1 - tak
- 20. liczba bloków aparaturowych
- 21. liczba bloków fikcyjnych
- 22. maksymalna liczba elementów w pojedynczym wierszu tablicy «BNPMPB dla grupy kolumn C
- 23. liczba kolumn tablicy PB
- 24. czy drukować tablicę PB
 - =-1 - tak w formacie zadanym przez FPB
 - =0 - nie
 - >0,- tak w formacie jednolitym zadanym przez tę liczbę

Jeżeli trzeba drukować tablicę PB w formacie zadanym przez FPB to

- 25. liczba wierszy tablicy «FPB
- 26. liczba wierszy tablicy APB; jeżeli tablica APB nie występuje to wtedy zero

Jeżeli tablica APB występuje to

- 27. liczba kolumn tablicy APB
- 28. czy drukować tablicę APB

= -1 - tak w formacie zadanym przez FAP

= 0 - nie

> 0 - tak w formacie jednolitym zadanym przez tę
liczbę

Jeżeli trzeba drukować tablicę APB w formacie zadanym
przez FAP to

29. liczba wierszy tablicy FAP

30. czy tablica EP występuje i czy ją drukować

= 0 - nie występuje i nie drukować

> 0 - występuje i drukować w formacie zadanym przez
tę liczbę

31. liczba wierszy tablicy LV; jeżeli tablica LV nie występuje
to wtedy zero.

Jeżeli tablica LV występuje to

32. liczba kolumn tablicy LV

33. liczba wierszy tablicy ME; jeżeli tablica ME nie występuje
to wtedy zero

Jeżeli tablica ME występuje to

34. liczba kolumn tablicy ME

35. czy drukować tablicę ME

= -1 - tak w formacie zadanym dla kolumn przez
FME

= 0 - nie

> 0 - tak w formacie jednolitym zadanym przez
tę liczbę

36. liczba wierszy tablicy PCP; jeżeli tablica PCP nie wystę-
puje to zero

Jeżeli tablica PCP występuje to

37. liczba kolumn tablicy PCP

- 38. format drukowania temperatur strumieni
- 39. format drukowania ciśnień strumieni
- 40. format drukowania molowych natężeń przepływu poszczególnych faz strumieni
- 41. format drukowania ułamków molowych poszczególnych składników strumieni
- 42. tablica CS; patrz uwaga 1
- Jeżeli tablica NO występuje to
 - 43. tablica NO; patrz uwaga 1
- 44. tablica «SNISTST
- Jeżeli tablica «IIIV występuje to
 - 45. tablica «IIIV
- Jeżeli tablica LS występuje to
 - 46. tablica LS; patrz uwaga 2
- Jeżeli tablica CE występuje to
 - 47. tablica CE; patrz uwaga 2
- Jeżeli tablica ECE występuje to
 - 48. tablica ECE; patrz uwaga 2
- 49. tablica «BNPMPB
- Jeżeli trzeba drukować tablicę PB w formacie zadanym przez FPB to
 - 50. tablica «FPB
- Jeżeli tablica APB występuje to
 - 51. tablica «APB
 - Jeżeli trzeba drukować tablicę APB w formacie zadanym przez FAP to
 - 52. tablica «FAP
- Jeżeli tablica LV występuje to
 - 53. tablica LV; patrz uwaga 2

Jeżeli tablica ME występuje i trzeba ją drukować w formacie
zadany dla kolumn przez FME to

54. tablica FME; patrz uwaga 1

Jeżeli tablica PCP występuje to

55. skąd ma być czytana tablica PCP

1 - z kart

2 - z kart i ze zbioru PHCHP

3 - ze zbioru PHCHP

Jeżeli tablica PCP ma być czytana z kart i ze zbioru
PHCHP to

56. liczba wierszy tablicy PCP / liczba substancji /
która ma być czytana z kart

Jeżeli tablica PCP ma być czytana z kart lub z kart
i ze zbioru PHCHP to

57. wiersze tablicy PCP w odpowiedniej ilości;
patrz uwaga 2

Jeżeli tablica PCP ma być czytana z kart i ze zbioru
PHCHP lub ze zbioru PHCHP to

58. numery substancji dla których parametry trzeba
trzeba przeczytać ze zbioru PHCHP ułożone w
dowolnej kolejności i zakończone *.

Jeżeli do zbioru ADDATA trzeba wczytać dane to

59. liczba tablic z danymi, które trzeba wczytać do
zbioru ADDATA

60. a/ górne granice kolejnych wskaźników pierwszej
tablicy zakończone *

b/ elementy pierwszej tablicy ułożone "wierszami"

c/ łańcuch z nazwą rekordu wymiaru tj. podzbioru do którego mają być wpisane górne granice wskaźników pierwszej tablicy

d/ łańcuch z nazwą rekordu danych tj. podzbioru do którego ma być wpisana pierwsza tablica

61. a/ górne granice kolejnych wskaźników drugiej tablicy zakończone *

b/ elementy drugiej tablicy ułożone "wierszami"

c/ łańcuch z nazwą rekordu wymiaru tj. podzbioru do którego mają być wpisane górne granice wskaźników drugiej tablicy

d/ łańcuch z nazwą rekordu danych tj. podzbioru do którego ma być wpisana druga tablica.

...

IX. DRUKOWANIE WYNIKÓW PRZEZ PROGRAM ORGANIZACYJNY

Po bezbłędnym przebiegu obliczeń systemu program organizacyjny przystąpi do drukowania wyników. Najpierw wyprowadzane są wartości cech strumieni systemu. Przyjęto przy tym następujące oznaczenia

T - temperatura [K]

P - ciśnienie [N/m^2]

N - molowe natężenie przepływu [kmol/s]

X - ułamek molowy składnika strumienia [kmol/kmol]

W nawiasach kwadratowych przy X podawany jest numer substancji będącej danym składnikiem. Numery strumieni są numerami zewnętrznymi. Drukowane są także informacje o stanach skupienia poszczególnych faz strumienia. Patrz również opis tablic IST i ST w rozdziale IV.

Przykład:

SYSTEM STREAMS

T = [K]

P = [N/m^2]

N = [KMOL/S]

X = [KMOL/KMOL]

STREAM 2

T = 300.00 P = 1.500 & 5

LIQUID PHASE 1

N = 0,04 X[4] = 0,370 X[7] = 0,330 X[6] = 0,000

LIQUID PHASE 2

N = 0,01 X [4] = 0,000 X [7] = 0,000 X [6] = 0,300

STREAM 5

T = 520.00 P = 1.400 & 5

GAS PHASE 1

N = 0,05 X [6] = 0,925 X [1] = 0,000

SOLID PHASE 2

N = 0,01 X [6] = 0,000 X [1] = 0,065

Strumień 2 o temperaturze 300 K i ciśnieniu $1,5 : 10^5 \text{ N/m}^2$ składa się z dwóch faz ciekłych. Molowe natężenie przepływu 1-szej fazy wynosi 0,04 kmol/s. Ułamki molowe składników strumienia w tej fazie są równe:

dla substancji 4: 0,37 kmol/kmol

dla substancji 7: 0,33 kmol/kmol

dla substancji 6: 0 kmol/kmol

Analogicznie interpretuje się wydruk dla fazy 2-ej i dla następnych strumieni.

Następnie wyprowadzane są / jeżeli sobie tego życzymy / wartości parametrów poszczególnych bloków tzn. wartości elementów z odpowiednich wierszy tablic PB, APB i EP, poprzedzone pięcioletnim skrótem nazwy bloku / lub słowem BLOCK / i numerem zewnętrznym bloku. W nawiasach kwadratowych przy nazwie tablicy / PB lub APB / drukowany jest numer kolumny, w której występuje dany parametr. Wartości elementów tablicy

EP wyprowadzane tylko dla bloków aparaturowych poprzedzone są tekstami OPERATING COSTS i CAPITAL COSTS. Patrz również opisy tablic PB, APB i EP w rozdziale IV.

Przykład:

SYSTEM BLOCKS

MIESZ 3

PB [1] = 12,5 PB [2] = 0,37 PB [4] = 7,85

APB [2] = 113,7 APB [3] = 83,0

OPERATING COSTS = 3,70 & 4 CAPITAL COSTS = 1,13 & 5

REAKT 5

PB [1] = 3,2 & 2 PB [2] = 0,513 PB [3] = 873,0

APB [2] = 0,039 APB [4] = 25,3

OPERATING COSTS = 5,36 & 4 CAPITAL COSTS = 5,78 & 5

Wartości cech bloku MIESZ 3 są następujące

- parametr 1 w tablicy PB: 12,5
- parametr 2 w tablicy PB: 0,37
- parametr 4 w tablicy PB: 7,87
- parametr 2 w tablicy APB: 113,7
- parametr 3 w tablicy APB: 83

Koszt eksploatacyjny aparatu wynosi 37 000 jednostek pieniężnych a inwestycyjny 113 000 jednostek pieniężnych. W podobny sposób odczytujemy wyniki dla następnych bloków.

Z kolei mogą być drukowane tablice SI zapamiętane w zbiorze STORE. Wydruk zawartości tablic SI zapisanych przez dany blok rozwiązywania układu równań jest poprzedzony tekstem ARRAY SI OF THE BLOCK i numerem zewnętrznym tego bloku oraz listą, która przyporządkowuje poszczególnym kolumnom tablicy SI odpowiednie cechy strumieni /oznaczenia takie jak przy drukowaniu cech strumieni/. W nawiasach kwadratowych przy SI podawany jest numer kolumny tablicy SI. Każdy wiersz tablicy SI poprzedzony jest słowem ITERATION i numerem iteracji. Patrz również opis tablicy SI i zbioru STORE rozdziałach IV i V odpowiednio.

Przykład:

ARRAY SI OF THE BLOCK 6

SI [1] = T OF THE STREAM 13

SI [2] = N OF THE LIQUID PHASE 2 IN THE STREAM 13

SI [3] = P OF THE STREAM 5

SI [4] = X [2] OF THE GAS PHASE 1 IN THE STREAM 5

ITERATION 0

SI [1] = 300.0 SI [2] = 0,030 SI [3] = 1.500 & 5

SI [4] = 0,100

ITERATION 2

SI [1] = 317,2 SI [2] = 0,033 SI [3] = 1.527 & 5

SI [4] = 0,095

ITERATION 4

SI [1] = 321,5 SI [2] = 0,035 SI [3] = 1,531 & 5
SI [4] = 0,091

Pierwsza kolumna tablicy SI zbudowanej przez blok 6 odpowiada temperaturze strumienia 13 / wartość tej cechy dla iteracji 0 wynosi 300 K/, druga - molowemu natężeniu przepływu ciekłej fazy 2 strumienia 13 / 0,03 kmol/s dla iteracji 0/, trzecia - ciśnieniu strumienia 5 / $1,5 \cdot 10^5$ N/m² dla iteracji 0/ i czwarta - ułamkowi molowemu substancji 2 w gazowej fazie 1 strumienia 5 / 0,1 kmol/kmol dla iteracji 0/.

Dalej można wyprowadzić tablice ME zapamiętane również w zbiorze STORE. Wydruk zawartości tablic ME zapisanych przez dany blok optymalizacyjny jest poprzedzony tekstem ARRAY ME OF THE BLOCK i numerem zewnętrznym tego bloku oraz listą, która przyporządkowuje poszczególnym kolumnom tablicy ME odpowiednie cechy strumieni z tablicy ST lub parametry obiektów z tablic PB i APB / numery cech i parametrów są numerami kolumn w których one występują/. Jeżeli pewnej kolumnie tablicy ME przyporządkowano wielkość nie występującą w tablicach ST, PB i APB to w liście nie będzie ona wyszczególniona, mimo drukowania jej wartości. W nawiasach kwadratowych przy ME podawany jest numer kolumny tablicy ME. Każdy wiersz tablicy ME jest poprzedzony słowem ITERATION lub ROW i numerem iteracji lub wiersza odpowiednio. Patrz również opis tablicy ME i zbioru STORE w rozdziale IV i V.

Przykład:

ARRAY ME OF THE BLOCK 7

ME [1] = FEATURE 1 OF THE STREAM 8

ME [2] = FEATURE 5 OF THE BLOCK 4 IN THE ARRAY PB

ME [3] = FEATURE 4 OF THE BLOCK 2 IN THE ARRAY APB

ITERATION 1

ME[1] = 300,0 ME [2] = 12,5 ME [3] = 1,57 ME [4] = 11

ITERATION 2

ME [1] = 315,7 ME [2] = 10,7 ME [3] = 1,69 ME [4] = 10

Pierwszej kolumnie tablicy ME zbudowanej przez blok 7 odpowiada cecha 1 / tj. temperatura / strumienia 8 / wartość tej cechy dla iteracji 1 wynosi 300 K/, drugiej - parametr 5 bloku 4 z tablicy PB / 12,5 dla iteracji 1/, trzeciej - parametr 4 bloku 2 z tablicy APB / 1,57 dla iteracji 1/ i czwartej - pewna wielkość nie występująca w tablicach ST, PB i APB / 11 dla iteracji 1/.

Jeżeli podczas przebiegu programu wystąpią błędy będą one sygnalizowane następująco:

1. INCORECT NUMERATION IN THE ARRAY i kod tablicy

W tablicy o podanym kodzie / patrz opisy tablic w rozdziale IV / występuje numer zewnętrzny strumienia lub bloku, który nie wystąpił w tablicy *SNISTST lub *BNPMPB odpowiednio.

2. IT IS UNKNOWN WHERE TO GO FROM THE BLOCK i numer zewnętrzny bloku. Po obliczeniu bloku o podanym numerze nie wiadomo jaki następny blok ma być obliczany gdyż albo nie wprowadzono w danych tablicy NO albo tablica NO nie

zawiera numeru obliczonego już bloku.

3. ERROR numer błędu

BLOCK numer zewnętrzny bloku

Podczas obliczeń bloku wystąpił niestandardowy błąd.

Patrz opis błędów związanych z danym blokiem.

4. NEGATIVE NUMBER IN KEY [4] AFTER CALCULATION OF BLOCK
numer zewnętrzny bloku.

Przed obliczeniem podanego bloku mieliśmy $KEY [4] \geq 0$,

a po obliczeniu jest $KEY [4] < 0$. Prawdopodobnie procedura opisująca blok jest błędna.

5. ERROR TYPE numer błędu

BLOCK numer zewnętrzny bloku

W czasie obliczeń danego bloku wystąpił standardowy błąd wykonania. / Patrz opis błędów wykonania w [1]/. Jeżeli błąd ten wystąpił podczas obliczeń ekonomicznych jest dodatkowo wyprowadzony tekst: ECONOMICAL CALCULATION CONTROLLED BY BLOCK i numer zewnętrzny bloku ekonomicznego sterującego aktualnie obliczeniami.

Jeżeli blok w którym wystąpił błąd miał strumienie wejściowe i wyjściowe to wartości cech tych strumieni będą drukowane wraz z poprzedzającymi je następującymi tekstami:

INLET STREAMS OF THE BLOCK numer zewnętrzny bloku

i

OUTLET STREAMS OF THE BLOCK numer zewnętrzny bloku
odpowiednio.

Po wystąpieniu błędów opisanych w punktach 2 ÷ 5 oprócz informacji o nich będą wyprowadzane:

1. jeżeli w programie były używane tablice SI i ME to zostaną wydrukowane z pamięci operacyjnej aktualne ich obrazy bez list przyporządkowujących poszczególnym kolumnom cechy strumieni i parametry bloków. Będą one poprzedzone tekstami ARRAY SI i ARRAY ME odpowiednio. Każdy wiersz poprzedzony będzie słowem ROW i numerem wiersza.
2. wszystkie wielkości, które miały być drukowane przy poprawnym przebiegu programu.

X. OGÓLNA CHARAKTERYSTYKA PROGRAMU ANALIZY SYSTEMÓW

Aby program organizacyjny BOSS mógł sterować obliczeniami systemu musimy wcześniej uzyskać pewne dodatkowe informacje poprzez analizę tego systemu. Do analizy systemów służy program ANALYZER napisany w języku ALGOL 1900. Spełnia on następujące funkcje:

- buduje system informacyjny dla danego systemu
- wydziela wszystkie σ -podsystemy informacyjne
- sprawdza czy jednopoziomowe zadanie optymalizacji systemu zostało poprawnie postawione
- wydziela grupy σ -podsystemów informacyjnych, które można obliczać niezależnie od siebie
- ustala kolejność obliczania σ -podsystemów informacyjnych
- znajduje optymalne zbiory cech iterowanych / optymalne zbiory przecinające / dla σ -podsystemów informacyjnych i odpowiadające im kolejności obliczeń bloków
- bada, które cechy strumieni mogą być zmiennymi decyzyjnymi, a które mogą być używane do obliczenia wartości funkcji celu.

Program ANALYZER jest ułożony w oparciu o rozważania przeprowadzone w rozdziale II. Poza tym wykorzystano algorytmy: wyznaczania wszystkich dróg w grafie - paths, wyznaczania konturów w grafie zorientowanym - cprintcircuits / z pewnymi modyfikacjami i zmienioną nazwą na KONTURY / opublikowane w [22], wyznaczania optymalnych zbiorów przecinających opisany w [45] /patrz rozdział II punkt 5/, obliczania elementów tablicy strumieni /zmodyfikowany/ oraz obliczania elementów tablicy

powiązań przedstawione w [17].

W dodatku 2 przedstawiono schemat blokowy programu ANALYZER.

XI. CZYTANIE DANYCH PRZEZ PROGRAM ANALIZY SYSTEMÓW

1. TABLICE WYSTĘPUJĄCE PRZY CZYTANIU DANYCH

«BNPM - macierz procesu

	1	2	
1			✖
			✖
			⋮
			✖
i			✖
			⋮
			✖
NA			✖

NA - liczba bloków aparaturowych

Tablica «BNPM zawiera opis struktury systemu. Jeden wiersz tablicy odpowiada jednemu blokowi aparaturowemu. Znaczenie poszczególnych kolumn dla i-tego wiersza, któremu odpowiada blok mający k strumieni wejściowych i l strumieni wyjściowych jest następujące:

1. nr zewnętrzny bloku
2. nr zewnętrzny strumienia ze znakiem plus jeżeli jest to strumień wejściowy albo ze znakiem minus jeżeli jest to strumień wyjściowy
3. nr wejścia lub wyjścia bloku, które jest związane ze strumieniem z punktu 2 odpowiednio do jego znaku
4. nr zewnętrzny strumienia ze znakiem plus jeżeli jest to strumień wejściowy albo ze znakiem minus jeżeli jest to strumień wyjściowy

5. nr wejścia lub wyjścia bloku, które jest związane ze strumieniem z punktu 4 odpowiednio do jego znaku

:

$2(k+1)$. nr zewnętrzny strumienia ze znakiem plus jeżeli jest to strumień wejściowy albo ze znakiem minus jeżeli jest to strumień wyjściowy

$2(k+1)+1$. nr wejścia lub wyjścia bloku, które jest związane ze strumieniem z punktu $2(k+1)$ odpowiednio do jego znaku

$2(k+1)+2$. *

Gwiazdka nie kończy liczby i dlatego między ostatnią cyfrą liczby a gwiazdką muszą wystąpić albo dwie spacje albo średnik albo przecinek albo nowa linia. Kolejność wierszy w tablicy dowolna.

MSNS - wymiary strumieni

	1	2
1		
i		
NS		

NS - liczba strumieni systemu

Tablica MSNS zawiera wymiary strumieni systemu. Znaczenie poszczególnych kolumn jest następujące:

1. nr zewnętrzny strumienia

2. wymiar strumienia / liczba cech strumienia / z punktu 1

$\pi F E / \pi F B /$ - wartości funkcjonału F dla cech strumieni
wejściowych / wyjściowych/bloku

	1	2	
1			*
			*
			*
			*
			*
			*
			*
			*
NIS(NOS)			*

NIS - liczba strumieni wejściowych bloku

NOS - liczba strumieni wyjściowych bloku

Tablica $\pi F E / \pi F B /$ zawiera wartości funkcjonału F dla cech strumieni wejściowych / wyjściowych / danego bloku / patrz str.9/ a więc opisuje kierunki przepływu informacji dla tych cech. Znaczenie poszczególnych kolumn dla i -tego wiersza, któremu odpowiada strumień wejściowy / wyjściowy / opisany przez k cech, jest następujące

1. nr wejścia / wyjścia / bloku

2. wartość funkcjonału F dla 1-szej cechy strumienia

0 - gdy blok sam oblicza wartość 1-szej cechy

1 - gdy blok wymaga wartości 1-szej cechy jako danej

3. wartość funkcjonału F dla 2-ej cechy strumienia

0 - gdy blok sam oblicza wartość 2-ej cechy

1- gdy blok wymaga wartości 2-ej cechy jako danej

k+1. wartość funkcjonału F dla k-tej cechy strumienia

0 - gdy blok sam oblicza wartość k-tej cechy

1 - gdy blok wymaga wartości k-tej cechy jako danej

k+2. x

Jeżeli wyniki analizy systemu mają być wykorzystane przez program organizacyjny BOSS to kolejność cech strumieni musi być zgodna z charakterystyką strumienia / patrz str. 59,60%.

Zera znajdujące się pomiędzy gwiazdką, a pierwszą liczbą różną od zera leżącą na lewo od gwiazdki, można pominąć. Gwiazdka nie kończy liczby i dlatego między ostatnią cyfrą liczby a gwiazdką muszą wystąpić albo dwie spacje albo średnik albo przecinek albo nowa linia.

Kolejność wierszy w tablicy dowolna.

DS / DO/ - zmienne decyzyjne będące cechami strumieni / obiektów/

	1	2
1		
i		
NDS(NDO)		

NDS - liczba zmiennych decyzyjnych będących cechami strumieni

NDO - liczba zmiennych decyzyjnych będących cechami obiektów

Tablica DS / DO / określa, które spośród cech strumieni / obiektów / są zmiennymi decyzyjnymi dla danego zadania optymalizacji. Znaczenie poszczególnych kolumn dla i-tego wiersza jest następujące:

1. nr zewnętrzny strumienia / obiektu /
2. nr cechy strumienia / obiektu / z punktu 1 spełniającej rolę zmiennej decyzyjnej

Kolejność wierszy w tablicy jest dowolna.

Uwaga: Przez nr cechy strumienia rozumiemy pozycję tej cechy w charakterystyce strumienia / patrz str.59,60/ lub co na jedno wychodzi nr kolumny tablicy ST / patrz str.63/ w której znajduje się ta cecha. Numery cech obiektów nie mają dla programu ANALYZER żadnego znaczenia i dlatego każdy może przyjąć własną konwencję numeracji np. dla cechy umieszczonej w tablicy PB / patrz str.82,83/ nr kolumny w której znajduje się ta cecha, a dla cechy umieszczonej w tablicy APB / patrz str.84,85/ nr kolumny w której znajduje się ta cechy pomnożony przez 100.

VS / VO/- zmienne funkcji celu będące cechami strumieni /obektów/

	1	2
1		
i		
NVS(NVO)		

NVS - liczba zmiennych funkcji celu będących cechami strumieni

NVO - liczba zmiennych funkcji celu będących cechami obiektów

Tablica VS / VO/ określa, które spośród cech strumieni / obiektów są zmiennymi funkcji celu dla danego zadania optymalizacji.

Znaczenie poszczególnych kolumn dla i-tego wiersza jest następujące

1. nr zewnętrzny strumienia / obiektu /
2. nr cechy strumienia / obiektu / z punktu 1 spełniającej rolę zmiennej funkcji celu

Kolejność wierszy w tablicy jest dowolna. Patrz także uwaga w opisie tablic DS i DO.

2. KOLEJNOŚĆ CZYTANIA DANYCH

1. Czy kierunki przepływu informacji są zgodne dla wszystkich cech strumieni z kierunkami przepływu substancji?

0 - nie

1 - tak

2. Co trzeba znaleźć?

1 - uporządkowanie σ -podsystemów informacyjnych przy nieobecności zadania optymalizacji systemu

2 - uporządkowanie σ -podsystemów informacyjnych z uwzględnieniem zadania optymalizacji systemu

3 - które spośród cech strumieni mogą pełnić rolę zmiennych decyzyjnych, a które zmiennych funkcji celu

Jeżeli trzeba znaleźć uporządkowanie σ -podsystemów informacyjnych to

3 . Czy dla każdego σ -podsystemu informacyjnego zawierającego informacyjne pętle sprzężenia zwrotnego znaleźć wszystkie możliwe optymalne zbiory cech iterowanych i odpowiadające im kolejności obliczania bloków

0 - nie

1 - tak

4. Liczba bloków aparaturowych systemu

5. Liczba kolumn tablicy $\#BNPM$ / maksymalna liczba elementów w pojedynczym wierszu tablicy bez uwzględnienia gwiazdki/

6. Liczba strumieni systemu

7. Tablica $\#BNPM$

8. Tablica $\#SNS$

Jeżeli kierunki przepływu informacji nie są zgodne dla wszystkich cech strumieni z kierunkami przepływu substancji to

9. Dla wszystkich bloków aparaturowych systemu trzeba wczytać kolejno

a/ nr zewnętrzny bloku aparaturowego

b/ tablica $\#Fd$ dla bloku z punktu a

c/ tablica $\#FB$ dla bloku z punktu a

Jeżeli trzeba znaleźć uporządkowanie σ -podsystemów informacyjnych z uwzględnieniem zadania optymalizacji to

10. Liczba zmiennych decyzyjnych będących cechami strumieni

11. Liczba zmiennych decyzyjnych będących cechami obiektów

12. Liczba zmiennych funkcji celu będących cechami strumieni

13. Liczba zmiennych funkcji celu będących cechami strumieni

Jeżeli liczba zmiennych decyzyjnych będących cechami strumieni jest większa od zera to

14. Tablica DS

Jeżeli liczba zmiennych decyzyjnych będących cechami obiektów jest większa od zera to

15. Tablica DO

Jeżeli liczba zmiennych funkcji celu będących cechami strumieni jest większa od zera to

16. Tablica VS

Jeżeli liczba zmiennych funkcji celu będących cechami obiektów jest większa od zera to

17. Tablica VO

Uwaga: Wszystkie tablice czytane są wierszami.

XII. DRUKOWANIE WYNIKÓW PRZEZ PROGRAM ANALIZY SYSTEMÓW

Jeżeli program ANALYZER ma znaleźć uporządkowanie σ -podsystemów informacyjnych to na początku będzie drukowana lista σ -podsystemów informacyjnych rozpoczynająca się nagłówkiem THE LIST OF SUBSYSTEMS. Zawiera ona opisy poszczególnych σ -podsystemów informacyjnych składające się z łańcucha SUBSYSTEMS i numeru σ -podsystemu informacyjnego / nadawanego przez program / oraz numerów zewnętrznych bloków wchodzących w skład tego σ -podsystemu informacyjnego poprzedzone słowem BLOCK lub BLOCKS i dwukropkiem.

Przykład:

THE LIST OF SUBSYSTEMS

SUBSYSTEM 1

BLOCK: 2

SUBSYSTEM 2

BLOCKS: 3 5 11

System składa się w tym przypadku z dwóch σ -podsystemów informacyjnych. W skład pierwszego z nich wchodzi blok o numerze zewnętrznym 2, a w skład drugiego bloki 3, 5 i 11.

Następnie będzie drukowany nagłówek THE SEQUENCE OF SUBSYSTEM CALCULATION i kolejność obliczeń σ -podsystemów informacyjnych /tylko jedna, nawet wtedy gdy jest ich więcej / przy czym dla każdego σ -podsystemu informacyjnego wyprowadzone są następujące informacje: nr kolejności obliczania, łańcuch THE SUBSYSTEM i nr σ -podsystemu informacyjnego z listy podsystemów oraz ewentualnie tekst WITH THE BLOCK OF SOLVING OF NONLINEAR

EQUATION SYSTEM / tzn. z blokiem rozwiązywania układu równań nieliniowych / jeżeli w tym σ -podsystemie informacyjnym występują informacyjne pętle sprzężenia zwrotnego. Jeżeli program porządkuje σ -podsystemy informacyjne z uwzględnieniem zadania optymalizacji to numer kolejności obliczania każdego σ -podsystemu informacyjnego uczestniczącego w obliczeniach optymalizacyjnych / tzn. takich, że wykonywane są iteracje zmiennych decyzyjnych / będzie poprzedzony znakiem *. W kolejności obliczeń uwzględnia się także blok funkcji celu i blok optymalizacyjny /patrz str. 114 / poprzez wydrukowanie poprzedzonych gwiazdkami i numerami kolejności obliczenia łańcuchów THE GOAL

FUNCTION BLOCK (IF IT IS NECESSARY) i THE OPTIMIZATION BLOCK odpowiednio. Ponadto na końcu zostanie wyprowadzony tekst THE FAMILY OF SUBSYSTEMS AND BLOCKS WHICH ARE DESIGNATED BY * IS CALCULATED ITERATIVELY DURING OPTIMIZATION tzn. rodzina podsystemów i bloków oznaczonych * jest obliczana iteracyjnie podczas optymalizacji. W wypadku gdy pewne grupy σ -podsystemów informacyjnych można obliczać niezależnie od siebie to wówczas dla każdej takiej grupy kolejność obliczeń zostanie ustalona niezależnie od innych grup i każda grupa zostanie poprzedzona łańcuchem THE GROUP i numerem grupy nadanym przez program. Dodatkowo na końcu zostanie wyprowadzony tekst

THE SEQUENCE OF GROUP CALCULATION IS OPTIONAL

EACH GROUP CAN BE CALCULATED INDEPENDENT OF REMAINING ONES
THAT IS SEPARATELY

tzn.

kolejność obliczania grup jest dowolna

Każdą grupę można obliczać niezależnie od pozostałych tzn.
oddzielnie.

Przykład

THE SEQUENCE OF SUBSYSTEM CALCULATION

GROUP 1

1. THE SUBSYSTEM 2
2. THE SUBSYSTEM 3 WITH THE BLOCK OF SOLVING OF NONLINEAR EQUATION SYSTEM

GROUP 2

- * 1. THE SUBSYSTEM 1
- * 2. THE SUBSYSTEM 4 WITH THE BLOCK OF SOLVING OF NONLINEAR EQUATION SYSTEM
- * 3. THE GOAL FUNCTION BLOCK (IF IT IS NECESSARY)
- * 4. THE OPTIMIZATION BLOCK
5. THE SUBSYSTEM 5

THE SEQUENCE OF GROUP CALCULATION IS OPTIONAL

EACH GROUP CAN BE CALCULATED INDEPENDENT OF REMAINING ONES
THAT IS SEPARATELY

THE FAMILY OF SUBSYSTEMS AND BLOCKS WHICH ARE DESIGNATED BY *
IS CALCULATED ITERATIVELY DURING OPTIMIZATION

Mamy tutaj do czynienia z dwiema grupami σ -podsystemów informacyjnych, które można obliczać niezależnie od siebie w dowolnej kolejności. Przyjźyjmy się bliżej drugiej grupie. Podsystem 4 zawiera informacyjne pętle sprzężenia zwrotnego i do jego obliczenia potrzebny jest blok rozwiązywania układu równań nieliniowych. Podsystemy i bloki oznaczone * biorą udział w obliczeniach optymalizacyjnych.

Kolejność obliczeń jest następująca

1. podsystem 1
2. podsystem 4
3. blok funkcji celu, jeżeli występuje
4. blok optymalizacyjny

jeżeli ekstremum jeszcze nie zostało znalezione to wracamy do punktu 1, a w przeciwnym razie przechodzimy do punktu 5

5. podsystem 5

Wydruki wyników otrzymanych na następnym etapie obliczeń, wykonywanym tylko dla σ -podsystemów informacyjnych zawierających informacyjne pętle sprzężenia zwrotnego, rozpoczynają się od tekstu THE OPTIMAL TEARING SETS AND THE SEQUENCE OF BLOCK CALCULATION ADEQUATE FOR THEM tzn. optymalne zbiory przecinające i odpowiadające im kolejności obliczeń bloków. Dla poszczególnych σ -podsystemów informacyjnych drukuje się wszystkie otrzymane rozwiązania. Najpierw wyprowadza się łańcuch THE SUBSYSTEM i nr podsystemu z listy podsystemów. Jeżeli teraz kierunki przepływu informacji są zgodne dla wszystkich cech strumieni z kierunkami przepływu substancji, to drukowany jest łańcuch THE TORN STREAMS: i numery zewnętrzne strumieni rozerwanych /iterowanych/. W przeciwnym wypadku wyprowadzony jest tekst THE TORN FEATURES: i dla każdego strumienia zawierającego cechy rozerwane /iterowane/ drukowane są: łańcuch STREAM, nr zewnętrzny strumienia, dwukropek i numery cech rozerwanych w tym strumieniu. Dalej drukuje się napis THE SEQUENCE OF BLOCK CALCULATION i kolejność obliczania bloków przy czym dla każdego bloku wyprowadzane są następujące infor-

macje: nr kolejności obliczania, łańcuch THE BLOCK i nr zewnętrzny bloku. Na końcu poprzedzony również numerem kolejności obliczania, drukowany jest łańcuch THE BLOCK OF SOLVING OF NONLINEAR EQUATION SYSTEM tzn. blok rozwiązywania układu równań nieliniowych.

Przykład

THE OPTIMAL TEARING SETS AND THE SEQUENCE OF BLOCK CALCULATION ADEQUATE FOR THEM

THE SUBSYSTEM 2

THE TORN FEATURES:

STREAM 1: 2 3

STREAM 3: 4 5 7

THE SEQUENCE OF BLOCK CALCULATION

1. THE BLOCK 6
2. THE BLOCK 5
3. THE BLOCK OF SOLVING OF NONLINEAR EQUATION SYSTEM

Dla podsystemu 2 cechami iterowanymi / rozerwanymi / z optymalnego zbioru przecinającego są cechy 2 i 3 strumienia 1 oraz cechy 4, 5 i 7 strumienia 3. Kolejność obliczeń bloków jest następująca

1. blok 6

2. blok 5

3. blok rozwiązywania układu równań nieliniowych

jeżeli nie osiągnięto zadanej dokładności obliczenia cech rozerwanych i nie przekroczono zadanej maksymalnej liczby

iteracji to wracamy do punktu 1; w przeciwnym wypadku kończymy obliczenia podsystemu

Jeżeli program ANALYZER bada, które spośród cech strumieni mogą pełnić rolę zmiennych decyzyjnych, a które zmiennych funkcji celu to wyprowadzenie wyników rozpoczyna się od tekstu THE DIVISION OF THE SYSTEM STREAM FEATURES WITH REGARD ON THE PART THEY MAY PLAY DURING OPTIMIZATION tzn. podział cech strumieni systemu ze względu na role jakie mogą pełnić przy optymalizacji. Jeżeli teraz kierunki przepływu informacji są zgodne dla wszystkich cech strumieni z kierunkami przepływu substancji to drukowany jest łańcuch THE DECISION VARIABLES MAY BE THE FEATURES OF INLET STREAMS OF THE SYSTEM co znaczy, że zmiennymi decyzyjnymi mogą być cechy strumieni wejściowych systemu. W przeciwnym wypadku wyprowadzamy jest tekst THE DECISION VARIABLES MAY BE FOLLOWING FEATURES: / tzn. zmiennymi decyzyjnymi mogą być następujące cechy / i dla każdego strumienia zawierającego cechy, które mogą być zmiennymi decyzyjnymi drukowane są: łańcuch STREAM, nr zewnętrzny strumienia, dwukropek i numery cech tego strumienia mogące pełnić rolę zmiennych decyzyjnych. Na końcu zostaną wyprowadzone teksty THE FEATURES USED FOR CALCULATION OF GOAL FUNCTION VALUE MAY BE REMAINING STREAM FEATURES / tzn. cechami używanymi do obliczenia wartości funkcji celu mogą być pozostałe cechy strumieni / i IN THE SIMILAR WAY THE OBJECT FEATURES OF THE SYSTEM CAN BE EASILY DIVIDED / tzn. w podobny sposób można łatwo podzielić cechy obiektów systemu /.

Przykład

THE DIVISION OF THE SYSTEM STREAM FEATURES WITH REGARD ON
THE PART THEY MAY PLAY DURING OPTIMIZATION

THE DECISION VARIABLES MAY BE FOLLOWING FEATURES:

STREAM 2 : 1 3 4

STREAM 4 : 2 3 5

STREAM 5 : 1 2 3 4

THE FEATURES USED FOR CALCULATION OF GOAL FUNCTION VALUE MAY
BE REMAINING STREAM FEATURES

IN THE SIMILAR WAY THE OBJECT FEATURES OF THE SYSTEM CAN BE
EASILY DIVIDED

Rolę zmiennych decyzyjnych mogą pełnić cechy 1, 3 i 4 strumienia 2, cechy 2, 3 i 5 strumienia 4 oraz cechy 1, 2, 3 i 4 strumienia 5. Pozostałe cechy strumieni systemu mogą być używane jako zmienne funkcje celu.

Jeżeli podczas przebiegu programu wystąpią nietypowe błędy to będą one sygnalizowane następująco:

1. THE DIRECTIONS OF INFORMATION FLOW ARE IN CONTRADICTION

FOR:

THE FEATURE k OF THE STREAM m

THE FEATURE l OF THE STREAM n

⋮

gdzie k, l są numerami cech, a m, n numerami zewnętrznymi strumieni. Kierunki przepływu informacji są sprzeczne dla cechy k strumienia m , cechy l strumienia n itd. Oznacza to, że nie jest spełniony warunek $B^{-1}((0,0)) = \emptyset$ /patrz str 9/ .

2. THE DECISION VARIABLES WERE NOT DEFINED

Nie zostały zdefiniowane zmienne decyzyjne. W danych liczba zmiennych decyzyjnych będących cechami strumieni i liczba zmiennych decyzyjnych będących cechami obiektów są równe zeru.

3. THERE WERE NOT DEFINED THE FEATURES WHICH ARE USED FOR CALCULATION OF GOAL FUNCTION VALUE

Nie zostały zdefiniowane cechy służące do obliczenia wartości funkcji celu. W danych liczba zmiennych funkcji celu będących cechami strumieni i liczba zmiennych funkcji celu będących cechami obiektów są równe zeru.

4. THE FEATURE k IS NOT INVOLVED IN THE STREAM m

gdzie k jest numerem cechy, a m numerem zewnętrznym strumienia. Cecha k nie wchodzi w skład strumienia m . W tablicy DS lub VS podano dla strumienia m numer cechy k większy od wymiaru tego strumienia lub w tablicy «SNS wymiar strumienia m jest za mały.

5. THE FEATURE k OF THE STREAM m IS NOT ALLOWED TO BE A DECISION VARIABLE

gdzie k jest numerem cechy, a m numerem zewnętrznym strumienia. Cecha k strumienia m nie może być zmienną decyzyjną /nie może pełnić roli zmiennej decyzyjnej/.

6. THE FEATURE k OF THE STREAM m IS NOT ALLOWED TO BE USED FOR CALCULATION OF GOAL FUNCTION VALUE

gdzie k jest numerem cechy, a m numerem zewnętrznym strumienia. Cecha k strumienia m nie może być użyta do obliczenia wartości funkcji celu /nie może pełnić roli zmiennej funkcji celu/.

7. THE FEATURE k OF THE STREAM m IS AN APPARENT DECISION VARIABLE BECAUSE DOES NOT INFLUENCE THE GOAL FUNCTION VALUE

gdzie k jest numerem cechy, a m numerem zewnętrznym strumienia. Cecha k strumienia m jest pozorną zmienną decyzyjną gdyż nie wpływa na wartość funkcji celu.

8. THE FEATURE k OF THE OBJECT m IS AN APPARENT DECISION VARIABLE BECAUSE DOES NOT INFLUENCE THE GOAL FUNCTION VALUE

gdzie k jest numerem cechy, a m numerem zewnętrznym obiektu. Cecha k obiektu m jest pozorną zmienną decyzyjną gdyż nie wpływa na wartość funkcji celu.

9. THE FEATURE k OF THE STREAM m EMPLOYED FOR CALCULATION OF GOAL FUNCTION VALUE IS INDEPENDENT OF DECISION VARIABLES

gdzie k jest numerem cechy, a m numerem zewnętrznym strumienia. Cecha k strumienia m używana do obliczania wartości funkcji celu nie zależy od zmiennych decyzyjnych.

10. THE FEATURE k OF THE OBJECT m EMPLOYED FOR CALCULATION OF GOAL FUNCTION VALUE IS INDEPENDENT OF DECISION VARIABLES

gdzie k jest numerem cechy, a m numerem zewnętrznym obiektu. Cecha k obiektu m używana do obliczania wartości funkcji celu nie zależy od zmiennych decyzyjnych.

11. THE OPTIMIZATION TASK CONSIDERED CONSISTS IN FACT OF n MUTUALLY INDEPENDENT OPTIMIZATION TASKS

gdzie n jest liczbą zadań optymalizacji. Rozważane zadanie optymalizacji w rzeczywistości składa się z n wzajemnie

od siebie niezależnych zadań optymalizacji. Dla każdego z niezależnych zadań optymalizacji drukowane jest ponadto:

- a/ łańcuch THE OPTIMIZATION TASK i nr zadania optymalizacji
- b/ tekst THE DECISION VARIABLES i wszystkie zmienne decyzyjne
- c/ tekst THE FEATURES EMPLOYED FOR CALCULATION OF GOAL FUNCTION VALUE i wszystkie cechy służące do obliczania wartości funkcji celu / tzn. wszystkie zmienne funkcji celu /

Wszystkie zmienne decyzyjne i zmienne funkcji celu drukowane są w następujący sposób: THE FEATURE k OF THE STREAM m lub THE FEATURE k OF THE OBJECT m w zależności od tego czy dana zmienna jest cechą strumienia czy też obiektu / k jest numerem cechy, a m numerem zewnętrznym strumienia lub obiektu odpowiednio/.

12. INCORECT NUMERATION IN THE ARRAY i nazwa tablicy / może nią być: *BNPM, DS, DO, VS lub VO/.

Znaczenie błędu jest następujące:

- a/ dla tablicy *BNPM - w tablicy tej występuje ujemny nr. zewnętrzny obiektu
- b/ dla tablic DS, DO, VS, VO - w tablicy wystąpił nr zewnętrzny strumienia^(lub) obiektu, który nie występuje w tablicy *BNPM lub *SNS.

XIII. OPIS PROGRAMU WPROWADZAJACEGO DANE DO ZBIORÓW NA TAŚMACH MAGNETYCZNYCH

Program ARCHIVIST zapisuje dane do zbiorów ADDATA, LIBDATA-1 i LIBDATA-2 znajdujących się na taśmach magnetycznych. Organizację tych zbiorów opisano na str. 103+106. Kolejność wczytywania danych przez program jest następująca

1. czy zbiór już istnieje?

true - tak

false - nie

2. łańcuch z nazwą zbioru

3. liczba tablic z danymi, które trzeba wczytać do zbioru

4. a/ górne granice kolejnych wskaźników pierwszej tablicy
zakończone *

b/ elementy pierwszej tablicy ułożone "wierszami"

c/ łańcuch z nazwą pierwszego rekordu wymiaru tj. podzbioru do którego mają być wpisane górne granice wskaźników pierwszej tablicy

d/ łańcuch z nazwą pierwszego rekordu danych tj. podzbioru do którego ma być wypisana pierwsza tablica

5. a/ górne granice kolejnych wskaźników drugiej tablicy
zakończone *

b/ elementy drugiej tablicy ułożone wierszami

c/ łańcuch z nazwą drugiego rekordu wymiaru tj. podzbioru do którego mają być wpisane górne granice wskaźników drugiej tablicy

d/ łańcuch z nazwą drugiego rekordu danych tj. podzbioru do którego ma być wpisana druga tablica

⋮

Po zapisaniu na taśmie magnetycznej każdego rekordu wymiaru lub danych drukowany jest łańcuch THE DIMENSION RECORD k IS STORED ON THE MAGNETIC TAPE lub THE DATA RECORD k IS STORED ON THE MAGNETIC TAPE odpowiednio, gdzie k jest kolejnym numerem pary rekordów w skład której wchodzi dany rekord.

XIV. PODSUMOWANIE

Wprowadzenie pojęcia systemu i odpowiadającego mu systemu informacyjnego pozwoliło w sposób konsekwentny zbudować teoretyczne podstawy analizy systemów. Zdefiniowano σ -podsystemy informacyjne / tj. grupy obiektów powiązane informacyjnymi sprzężeniami zwrotnymi / i znaleziono porządkujące je relacje zarówno z uwzględnieniem jedno albo dwupoziomowego zadania optymalizacji systemu jak też w jego nieobecności. Sformułowano warunki dostateczne poprawności postawienia zadania optymalizacji systemu. Rozpatrzono w jakiej sytuacji system informacyjny rozpada się na niezależne od siebie części, które można obliczać osobno. Na podstawie rezultatów uzyskanych z rozważań teoretycznych opracowano program analizy systemów ANALYZER. Wyniki otrzymane z tego programu służą jako dane do programu organizacyjnego BOSS sterującego obliczeniami obiektów dowolnego systemu technologii chemicznej. Program BOSS jest elastyczny, gdyż strukturę systemu podaje się w danych a z procedur opisujących bloki dołącza się do programu tylko te, które są potrzebne do obliczenia danego systemu. Poza tym ze względu na dynamiczną rezerwację tablic zajmuje tylko tyle pamięci operacyjnej komputera ile jest potrzebne dla danego zadania.

XV. LITERATURA

1. ALGOL kompilatory zeszyt 1, 1301201/1, WZE ELWRO 1973
2. ALGOL opis języka, 13046, WZE ELWRO 1973
3. ALGOL procedury pomocnicze zeszyt 1, 1301202/1, WZE ELWRO 1973
4. ALGOL procedury pomocnicze zeszyt 2, 1301202/2, WZE ELWRO 1973
5. Boyadjiev Ch.B., Stateva R.R., On the structural analysis of large scale process systems, materiały z sympozjum "Computers in Chemical Engineering", Wysokie Tatry 1977, tom 2, str. 398-400.
6. Concept MARK 3, Computer Aided Design Centre, Cambridge
7. Design Aids for Chemical Engineer, Computer Aided Design Centre
8. Evans L.B., Steward D.G., Sprague C.R., Computer - Aided Chemical Process Design, Chem.Eng.Progr., 64/4/, 39-46, 1968
9. Fidkowski Z., Królikowski L., Kuciel E., Kuźniar J., Badania, modelowanie i projektowanie aparatów kolumnowych do wymiany masy, raporty Inst.Inż.Chem. i Urz.Ciepl. P.Wr. nr 38, 1977
10. Findeisen Wł., Syzmankowski J., Wierzbiński A., Teoria i metody obliczeniowe optymalizacji, WN, Warszawa 1977
11. Findeisen Wł., Wielopoziomowe układy sterowania, PWN, Warszawa 1974
12. Forder G.J., Hutchison H.P., The analysis of chemical plant flowsheets, Chem.Eng.Sci., 24, 771-785, 1969
13. Gratarola M., Pianzola B., Teconi M., Implementation of a flowsheeting program, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques au developpement du genie chimique et de la chimie industrielle", Paryż 1978, C11, str. 49-52.

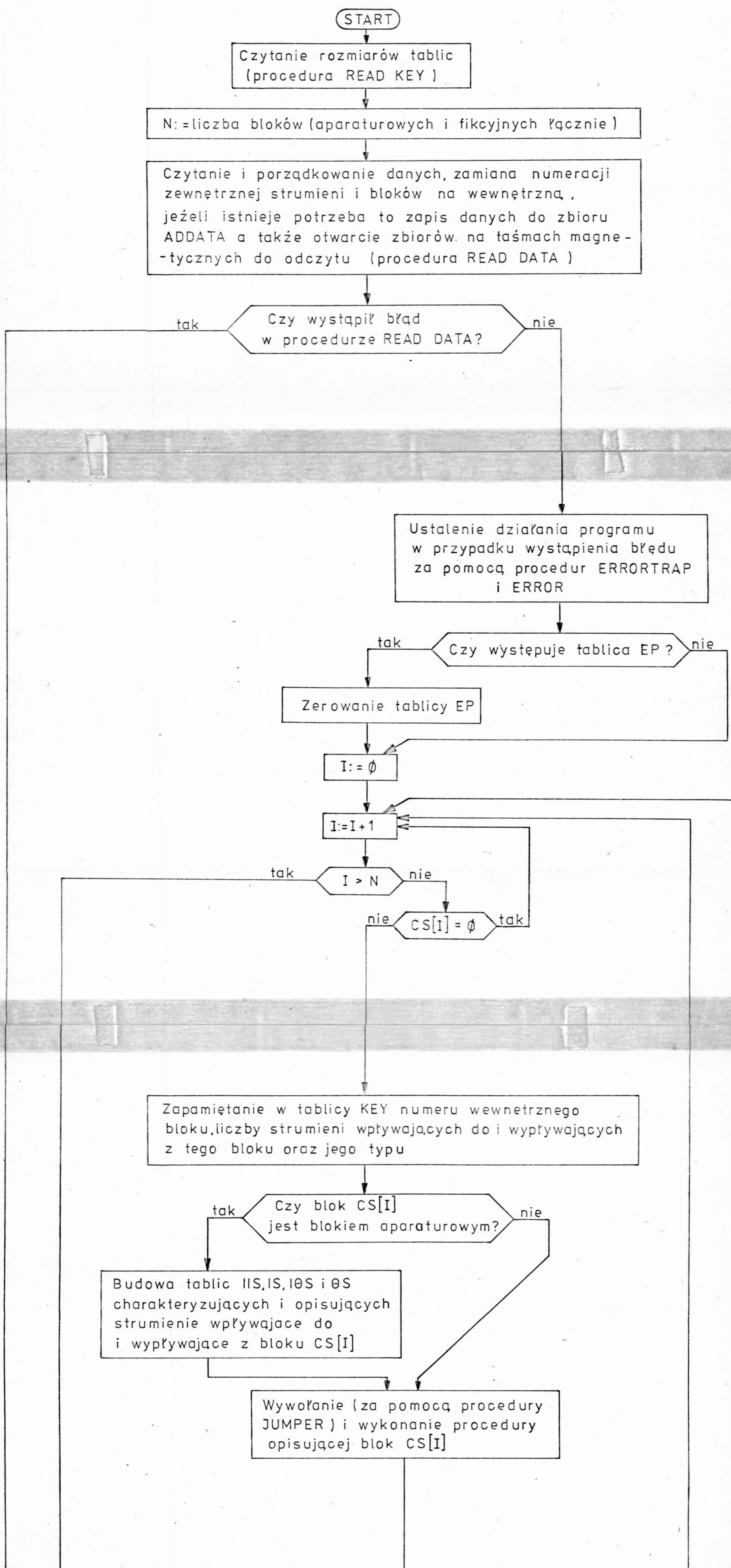
14. Himmelblau D.M., Morphology of decomposition, str. 1-13
w książce Decomposition of Large - Scale Problems pod redakcją D.M. Himmelblau, North Holland / American Elsevier, Amsterdam 1973
15. Husain A., Computers in chemical engineering. Design-part I, Chem. Age Ind., 18 / 1/, 17-23, 1967 .
16. Husain A., Varaprased S.S., Computers in chemical engineering Design - part III, Chem. Age Ind., 18 / 4/, 289-294, 1967
17. Kaczmarek A., Schematy blokowe modelowania, projektowania ciągów technologicznych przy pomocy EMC, pracy dyplomowa, Inst.Inż.Chem. i Urz. Ciepl. P.Wr. Wrocław 1975.
18. Kevorkian A.K., Snoek J., Decomposition in Large Scale System Theory and Applications of Structural Analysis in Partitioning Disjoiting and Constructing Hierarchical Systems, str. 467-490 w książce Decomposition of Large - Scale Problems pod redakcją D.M. Himmelblau, North Holland / Americal Elsevie Amsterdam 1973
19. Klumpar J.V., Proces Predesign by Computer, Chem. Eng. 22, 114-122, 1969
20. Krop E.J., The new flowsheeting system PRODAS, materiały z międzynarodowego kongresu "Contribution des Calculatures electroniques an developpment du genie chimique et de la chimie industrielle", Paryż 1978, C7, str. 34-38.
21. Krou K., Gamilec A., Hoffman T., Dzonson A. Wuds D. S^zennon P. Matematyczeskoje modelirowanie chemiczeskich proizwodstw, Mir, Moskwa 1973
22. Kucharczyk J., Sysło M., Algorytmy optymalizacji w języku Algol 60, PWN, Warszawa 1975.

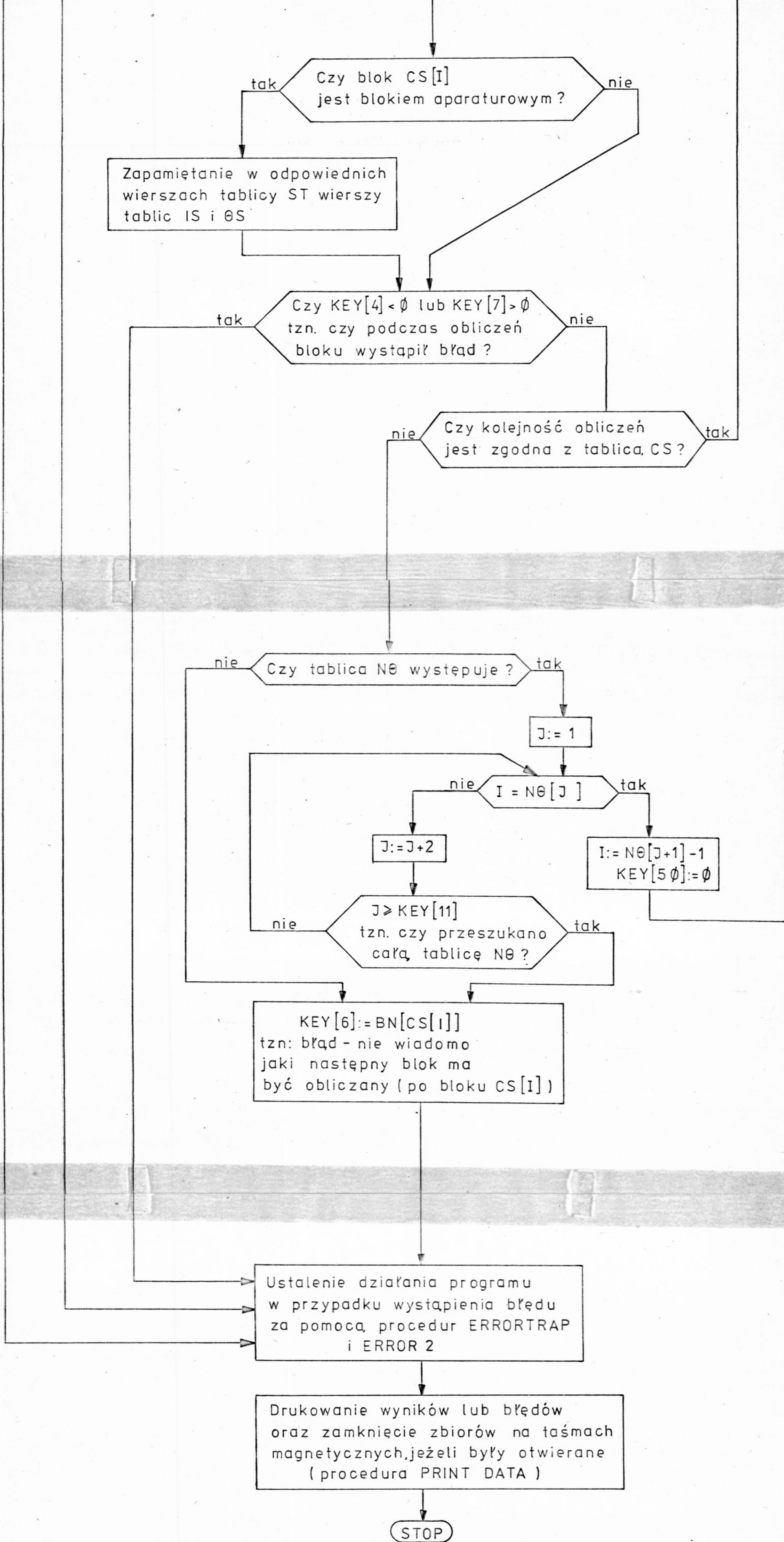
23. Kulikowski R., Sterowanie w wielkich systemach, WNT, Warszawa 1974
24. Ledet W.D., Himmelblau D.M., Decomposition Procedures for Solving a Large Scale Systems, Adv. Chem. Eng., 8, 185-254, 1970
25. Ławrow S.S., ALGOL 60. Wykład popularny z ćwiczeniami, WNT, Warszawa 1970
26. Maejima T., Shindo A., Computer - aided process engineering system in engineering firm, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques an developpment du genie chimique et de la chimie industrielle" Paryż 1978, C10, str. 44-48
27. Motard R.L., Shachm M., Rosen E.M., Steady state chemical process simulation, AIChE J., 21 / 3/, 417-436, 1975
28. Ogólna teoria systemów. Tendencje rozwojowe. WNT, Warszawa 1976
29. Oguchi G., Kano O., Mitsunaga M., Process flowsheet simulation by PSX, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques an developpment du genie chimique et de la chimie industrielle", Paryż 1978, C9 str. 39-43.
30. Ostrovsky G.M., Berezhinsky T.A., Belyaeva A.R., Quadratic methods of descent in problems of optimization of chemical engineering processes, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques an developpment du genie chimique et de la chimie industrielle", Paryż 1978, F6, str. 17-21

31. Ostrovsky G.M., Volin Yu.M., Problems of optimization of complex chemical and technological systems, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques an developpment du genie chimique et de la chimie industrielle", Paryż 1978, F2, str. 1-7.
32. Ostrowski G.M., Wolin J.M., Optymalizacja złożonych systemów technologii chemicznej, WNT, Warszawa 1974
33. Pho T.K., Lapidus L., Topics in Computer Aided Design; Part 1 An Optimum Tearing Algorithm for Recycle Systems, AIChE J., 19, 1170-1181, 1973
34. Pierucci S., Biardi G., Ranzi E., Dente M., A multipurpose computing programme for process flow - sheets calculation, materiały z sympozjum "Computers in chemical engineering", Wysokie Tatry 1977, tom 2, str. 401-409
35. Pierucci S., Biardi G., Ranzi E., Dente M., Computer aided design of chemical plants: base design and simulation by integrated approach, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques an developpment du genie chimique et de la chimie industrielle", Paryż 1978, C4, str. 18-25
36. Pierucci S., Biardi G., Ranzi E., Dente M., Large systems analysis: relationships among phenomenological, logical and numerical aspects, materiały z międzynarodowego kongresu "Contribution des Calculateurs electroniques an developpment du genie chimique et de la chimie industrielle", Paryż 1978, F9, str. 33-37
37. Sargent R.W.H., Westerberg A.W., "Speed-up" in chemical engineering design, Trans. Instn. Chem. Engrs, 42, 190-197, 1964 .

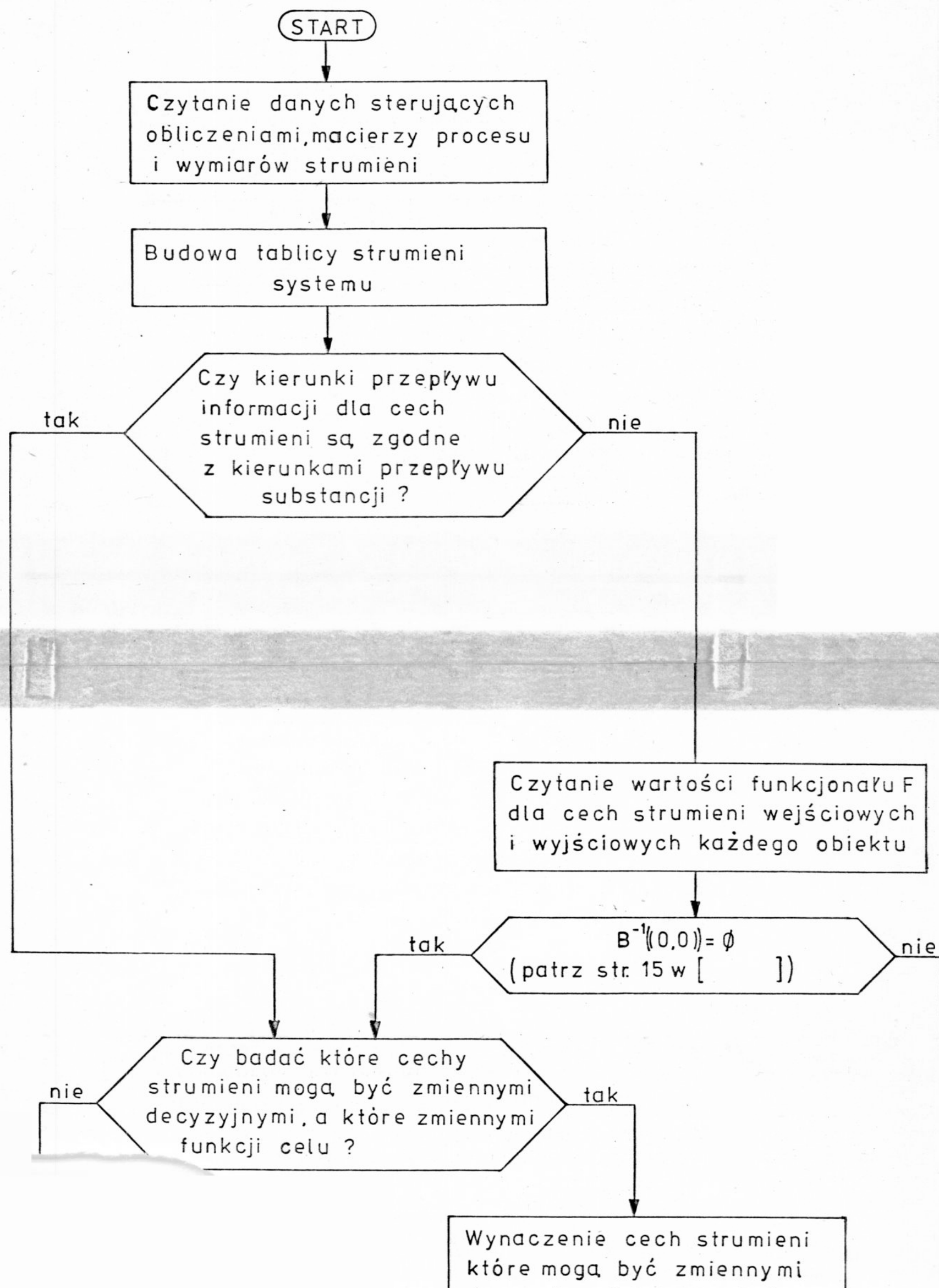
38. Scimus Systems Library, Scimus Information Technology Ltd.
39. Surma T., Programy nakładane w językach PLAN-3, FORTRAN 1900, ALGOL 1900, P. Wr., Wrocław 1976.
40. Szamkołowicz L., Teoria grafów skończonych, Ossolineum, Wrocław 1971
41. Upadhye R.S., Grens E.A., An Efficient Algorithm for Optimum Decomposition of Recycle Systems, AIChEJ., 18 / 3/, 533-539, 1972
42. Upadhye R.S., Grens E.A., Selection of decompositions for Process Simulation, AIChEJ., 21, 136-143, 1975
43. Veress G.E., Almasy G.A., Pallai I.M., On the solution of large scale problems by decomposition, str. 141-151 w książce Decomposition of Large-Scale Problems pod redakcją D.M. Himmelblau, North Holland / American Elsevier, Amsterdam 1973
44. Westerberg A.W., Decomposition Methods for Solving Steady State Process Design Problems, str. 379-397 w książce Decomposition of Large - Scale Problems pod redakcją D.M. Himmelblau, North Holland / American Elsevier, Amsterdam 1973
45. Wolin Ju.M., Ostrowskij G.M., Hansel K., Sauer D., Schnitzzahlminimierung in komplexen verfahrenstechnischen Schemata, Chem. Eng.Sci., 28, 2045-2049, 1973

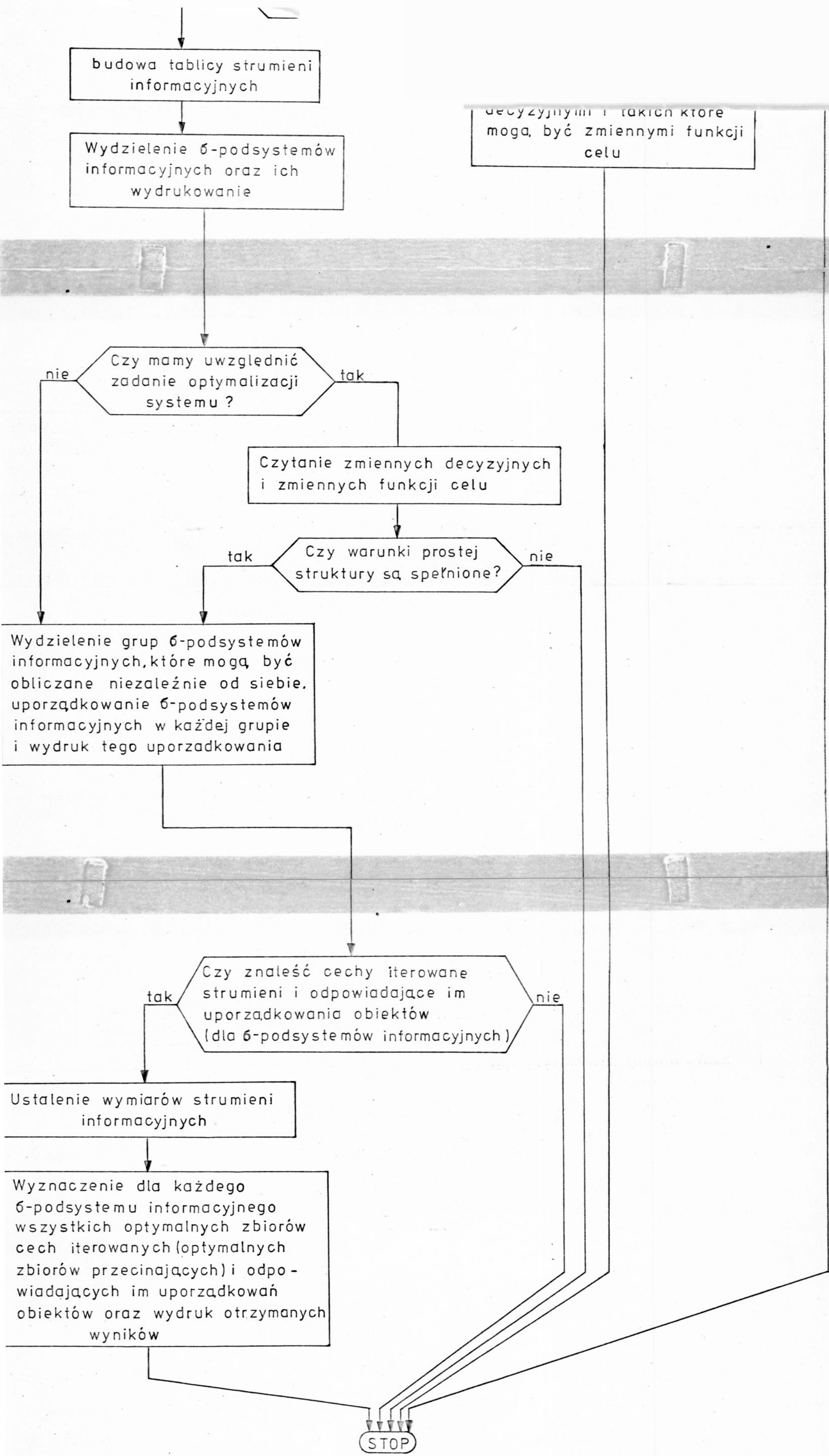
DODATEK 1.
SCHEMAT BLOKOWY PROGRAMU ORGANIZACYJNEGO BOSS





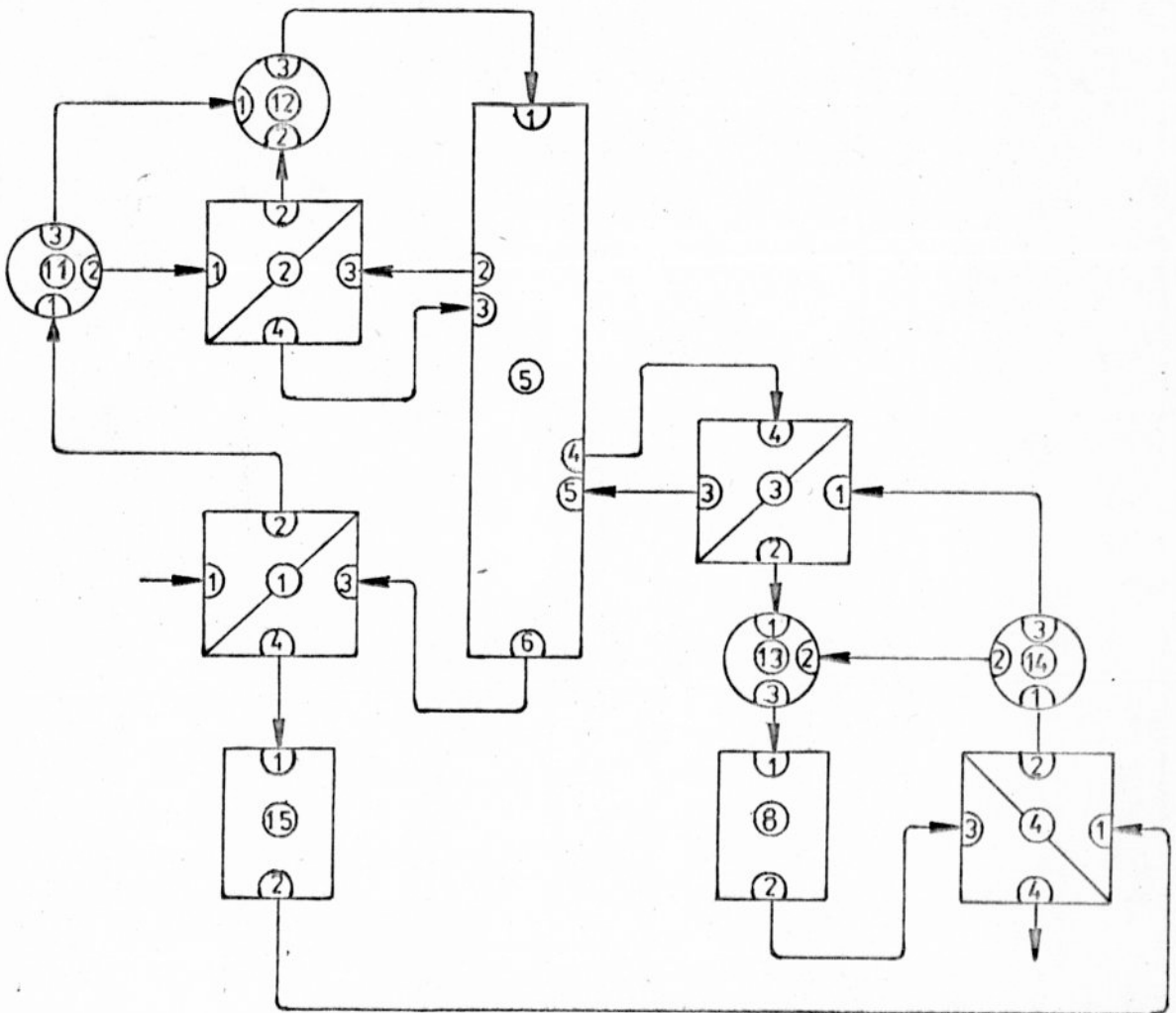
DODATEK 2.
SCHEMAT BLOKOWY PROGRAMU ANALIZY SYSTEMÓW





DODATEK 3

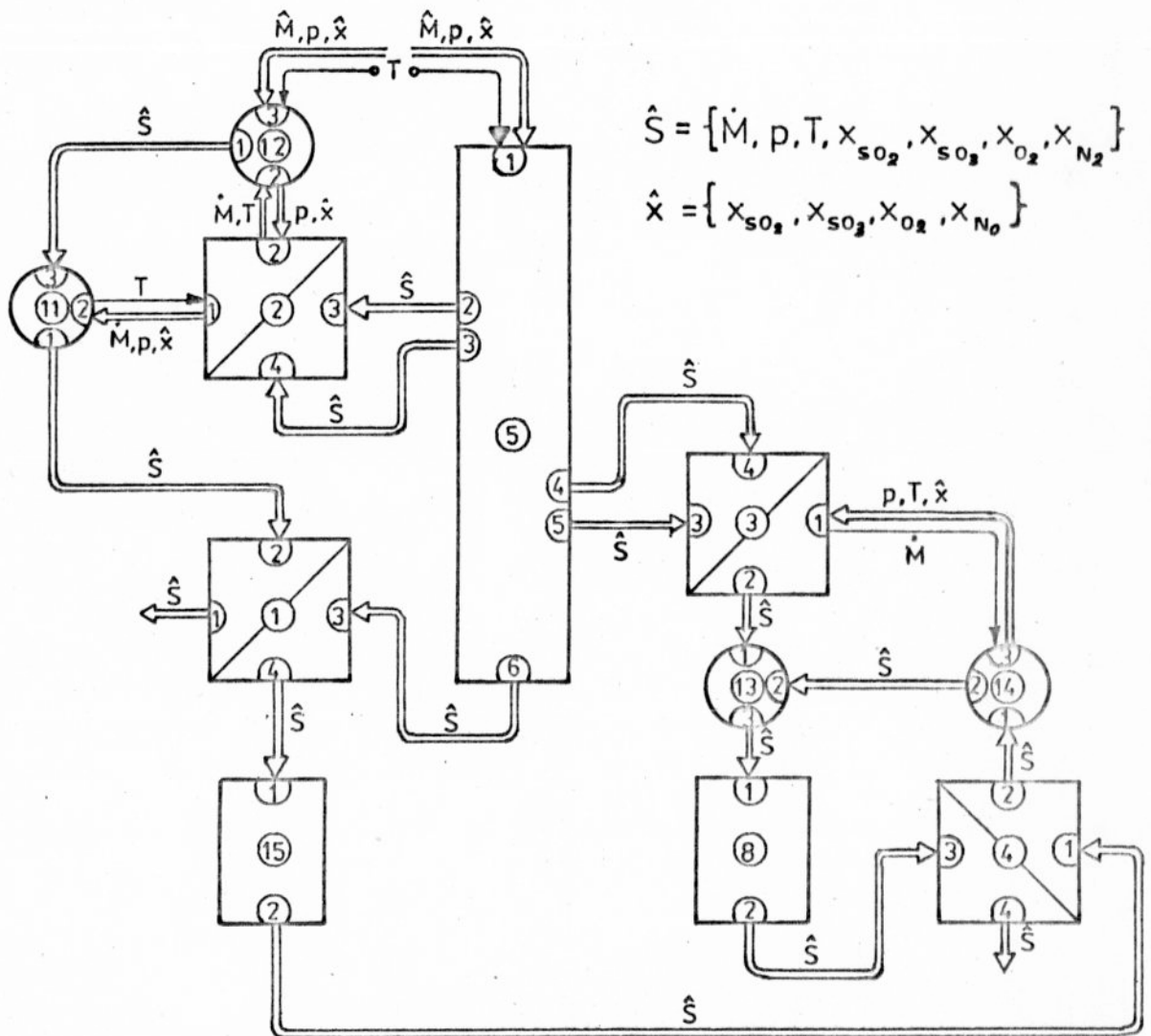
PRZYKŁAD ANALIZY SYSTEMU TECHNOLOGII CHEMICZNEJ



Rys. 3.1

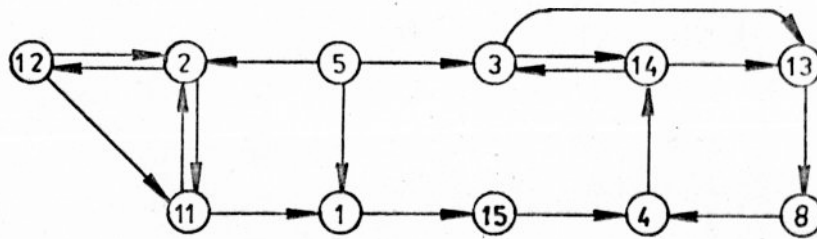
Rozważmy system pokazany na rys. 3.1. Dwutlenek siarki utleniany jest tutaj do trójtlenku siarki. Pierwszy stopień konwersji tworzą: trójstopniowy aparat kontaktowy 5 i trzy wy-

mienniki ciepła 1,2 oraz 3. Z absorbera 15 gdzie został usunięty SO_3 , gaz płynie do drugiego stopnia konwersji składającego się z jednostopniowego reaktora 8 i wymiennika ciepła 4. Katalizator z pierwszego stopnia konwersji musi być wymieniony. Wymagamy aby objętość nowego katalizatora była jak najmniejsza, a zarazem udział molowy SO_2 na wylocie 6 z aparatu kontaktowego

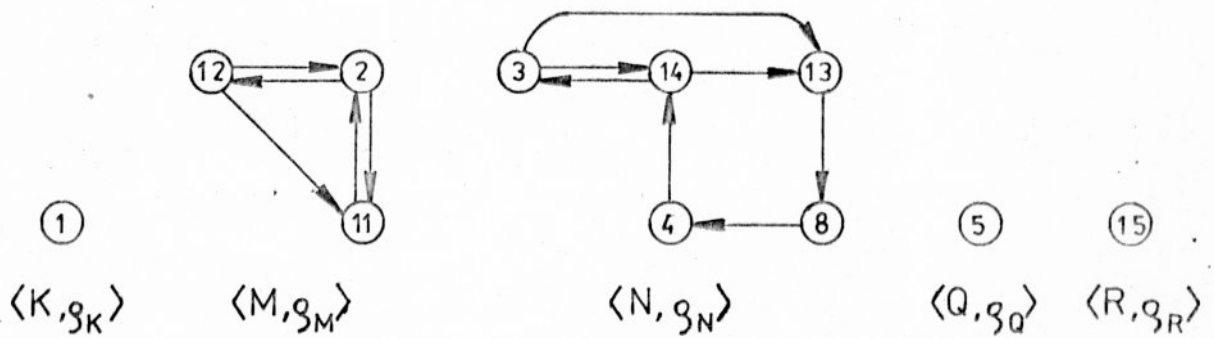


Rys. 3,2

był mniejszy od a. Algorytmy projektowania i modelowania aparatu kontaktowego były przedstawione w literaturze^{n/}. Na ich podstawie możemy zaproponować wartości funkcjonału $F(c)$ dla reaktorów 5 i 8.



Rys. 3.3



Rys. 3.4

^{n/}Kuciel E., Królikowski L., Modelling and designing of a catalytic reactor for sulphur dioxide oxidation, materiały z międzynarodowego kongresu "Contribution des Calculateurs électroniques au développement du génie chimique et de la chimie industrielle". Parvż 1978. C23. str. 95-99.

System informacyjny odpowiadający danemu systemowi technologicznemu jest pokazany na rys. 3.2. Jako zmienną decyzyjną przyjęto temperaturę gazu na wlocie 1 do reaktora 5. Wymienniki ciepła są wystarczająco duże aby zapewnić wymaganą wymianę ciepła. Wartości parametrów geometrycznych wszystkich aparatów są określone. Założono wartości natężenia przepływu, stężeń składników i ciśnienia na wlocie 1 reaktora 5. Te same cechy występują na wylocie 3 trójkąta 12. Graf odpowiadający systemowi informacyjnemu jest pokazany na rys. 3.3, a podgrafy opisujące strukturę σ -podsystemów informacyjnych na rys. 3.4.

Kolejność obliczania σ -podsystemów informacyjnych powinna być następująca

1. σ -podsystem informacyjny o strukturze $\langle Q, q_Q \rangle$
2. σ -podsystem informacyjny o strukturze $\langle M, q_M \rangle$
3. σ -podsystem informacyjny o strukturze $\langle K, q_K \rangle$
4. σ -podsystem informacyjny o strukturze $\langle R, q_R \rangle$
5. σ -podsystem informacyjny o strukturze $\langle N, q_N \rangle$

Rozwiązanie na początku σ -podsystemu informacyjnego o strukturze $\langle Q, q_Q \rangle$ /minimalna wartość funkcji celu została osiągnięta/ umożliwia nam obliczenie pozostałych σ -podsystemów informacyjnych dla poprawnych wartości cech strumieni wylotowych reaktora 5. Jest oczywiste, że σ -podsystemy informacyjne o strukturze $\langle M, q_M \rangle$ i $\langle N, q_N \rangle$ będą obliczane iteracyjnie.

DODATEK 4

RELACJE

Relację ϱ nazywamy relacją równoważności w zbiorze X , jeżeli ϱ jest relacją zwrotną, symetryczną i przechodnią tzn.

$$(z) \quad x \varrho x \quad \text{dla każdego } x \in X$$

$$(s) \quad x \varrho y \Rightarrow y \varrho x \quad \text{dla każdego } x, y \in X$$

$$(p) \quad (x \varrho y \wedge y \varrho z) \Rightarrow x \varrho z \quad \text{dla każdego } x, y, z \in X$$

Niech ϱ będzie dowolną relacją równoważności w $X \neq \emptyset$. Dla każdego elementu $x \in X$ oznaczmy

$$\|x\| = \{y \in X : x \varrho y\}$$

Zbiory $\|x\|$ dla $x \in X$ nazywamy klasami równoważności relacji ϱ w X . Zbiór wszystkich klas równoważności relacji ϱ w X oznaczamy symbolem X/ϱ .

Twierdzenie: Niech ϱ będzie dowolną relacją równoważności w zbiorze $X \neq \emptyset$. Spełnione są wówczas następujące warunki dla każdego $x, y \in X$:

$$(1) \quad x \in \|x\|$$

$$(2) \quad \|x\| = \|y\| \quad \text{wtedy i tylko wtedy gdy} \quad x \varrho y$$

$$(3) \quad \text{jeżeli } \|x\| \neq \|y\|, \text{ to klasy } \|x\| \text{ i } \|y\| \text{ są rozłączne}$$

Twierdzenie: Dowolna relacja równoważności ϱ w zbiorze $X \neq \emptyset$ ustala podział tego zbioru na niepuste i rozłączne podzbiory, mianowicie na klasy równoważności tej relacji, w taki sposób, że dwa elementy x, y zbioru X należą do tej samej klasy równoważności wtedy i tylko wtedy gdy $x \varrho y$.

Relację ϱ nazywamy porządkującą zbiór X jeżeli jest zwrotna, przechodnia i antysymetryczna tzn.

$$(z) \quad x \varrho x \quad \text{dla każdego } x \in X$$

$$(p) \quad (x \varrho y \wedge y \varrho z) \Rightarrow x \varrho z \quad \text{dla każdego } x, y, z \in X$$

$$(a) \quad (x \varrho y \wedge y \varrho x) \Rightarrow x = y \quad \text{dla każdego } x, y \in X$$

DODATEK 5

GRAFY

Grafem zorientowanym nazywamy parę uporządkowaną $\langle X, \rho \rangle$, gdzie X jest dowolnym skończonym zbiorem elementów zwanych wierzchołkami, a ρ relacją dwuargumentową określoną dla elementów zbioru X . Liczbę wszystkich wierzchołków grafu nazywamy rzędem grafu. Parę $[x, y]$ taką, że $x \rho y$ nazywamy łukiem. Ciąg wierzchołków grafu $[x_{k_1}, x_{k_2}, \dots, x_{k_{t+1}}]$ taki, że dla $i = 1, 2, \dots, t$ zachodzi $x_{k_i} \rho x_{k_{i+1}}$ nazywamy drogą / oznaczamy ją też symbolem $L(x_{k_1}, x_{k_{t+1}})$ /, a jeżeli $x_{k_i} \rho x_{k_{i+1}}$ lub $x_{k_{i+1}} \rho x_{k_i}$ - łańcuchem grafu. Drogę zamkniętą, tzn. taką, że $x_{k_1} = x_{k_{t+1}}$ nazywamy konturem / jeżeli poza tym żadne dwa inne wierzchołki nie są identyczne to mówimy wówczas o konturze prostym /.

Za długość drogi w grafie przyjmujemy liczbę łuków tej drogi. Graf nazywamy spójnym, jeżeli dla każdej pary różnych wierzchołków istnieje łączący je łańcuch. Podgrafem grafu $\langle X, \rho \rangle$ nazywamy dowolny graf $\langle K, \rho_K \rangle$ gdzie $K \subset X$, a relacja ρ_K jest relacją ρ ograniczoną do elementów zbioru K . Rodzinę podgrafów $\langle K, \rho_K \rangle$ grafu $\langle X, \rho \rangle$ taką, że $K \in \mathcal{K} \subset 2^X / 2^X$ - rodzina wszystkich podzbiorów zbioru X / oznaczamy $\mathcal{G}(\mathcal{K})$. Zauważmy, że dany graf $\langle X, \rho \rangle$ możemy w sposób równoważny określić jako parę uporządkowaną $\langle X, U \rangle$ gdzie $U = \{[x, y] : x, y \in X \wedge x \rho y\}$ jest zbiorem łuków grafu. Grafem częściowym grafu $\langle X, U \rangle$ nazywamy dowolny graf $\langle X, Z \rangle$ gdzie $Z \subset U$.

08/11/37
STATEMENT

26/07/78

ZAKŁADNIK 1
COMPILED BY XAIT MK. 5A

```
'BEGIN'
'INTEGER' I,J,K,N;
'BOOLEAN' B;
'INTEGER' 'ARRAY' KEY[1:521];
'PROCEDURE' READ KEY(V);
'INTEGER' 'ARRAY' V;
'ALGOL';
SELECT INPUT(3);
READ KEY(KEY);
N:=KEY[271]+KEY[281];
'BEGIN'
'INTEGER' 'ARRAY' NO[1:'IF' KEY[111]>0 'THEN' KEY[111] 'ELSE' 1],
SN[1:KEY[121]],
IST[1:KEY[121],3:KEY[131]],IIS[1:KEY[151],1:KEY[131]],
IOS[1:KEY[161],1:KEY[131]],IIV[1:'IF' KEY[171]>0 'THEN'
KEY[171] 'ELSE' 1,1:KEY[181]],CS,BN[1:N],PM[1:N,1:KEY[301]],
FPB[1:KEY[291],-4:'IF' KEY[321]=-1 'THEN' KEY[311] 'ELSE' -4],
FAP[1:KEY[361],1:'IF' KEY[351]=-1 'THEN' KEY[341] 'ELSE' 1],
FME[1:'IF' KEY[421]=-1 'THEN' KEY[411] 'ELSE' 1];
'ARRAY' ST[1:KEY[121],1:KEY[141]],IS[1:KEY[151],1:KEY[141]],OS[1:KEY[161],
1:KEY[141]],LS[1:'IF' KEY[191]>0 'THEN' KEY[191] 'ELSE' 1,1:KEY[201]],
CE[1:'IF' KEY[211]>0 'THEN' KEY[211] 'ELSE' 1,1:51],
ECE[1:'IF' KEY[221]>0 'THEN' KEY[221] 'ELSE' 1,1:KEY[231]],
SI[1:'IF' KEY[241]>0 'THEN' KEY[241] 'ELSE' 1,1:KEY[251]],
PB[1:N,1:KEY[311]],APB[1:'IF' KEY[331]>0 'THEN' KEY[331] 'ELSE' 1,
1:KEY[341]],EP[1:'IF' KEY[371]>0 'THEN' KEY[271] 'ELSE' 1,1:21],
LV[1:'IF' KEY[381]>0 'THEN' KEY[381] 'ELSE' 1,1:KEY[391]],
ME[1:'IF' KEY[401]>0 'THEN' KEY[401] 'ELSE' 1,1:KEY[411]],
PCP[1:'IF' KEY[431]>0 'THEN' KEY[431] 'ELSE' 1,1:KEY[441]];
'PROCEDURE' READ DATA(KEY,CS,NO,SN,IST,ST,IIV,LS,CE,ECE,BN,PM,PB,FPB,
APB,FAP,IIV,FME,PCP,EA);
'INTEGER' 'ARRAY' KEY,CS,NO,SN,IST,IIV,BN,PM,FPB,FAP,FME;
'ARRAY' ST,LS,CE,ECE,PB,APB,LV,PCP;
'LABEL' EA;
'ALGOL';
'PROCEDURE' JUMPER(KEY,SN,IST,ST,IIS,IS,IOS,OS,IIV,LS,CE,ECE,SI,BN,PB,
APB,EP,LV,ME,PCP);
'INTEGER' 'ARRAY' KEY,SN,IST,IIS,IOS,IIV,BN;
'ARRAY' ST,IS,OS,LS,CE,ECE,SI,PB,APB,EP,LV,ME,PCP;
'ALGOL';
'PROCEDURE' PRINT DATA(KEY,CS,SN,IST,ST,IS,OS,IIV,SI,BN,PM,PB,FPB,APB,
FAP,EP,ME,FME);
'INTEGER' 'ARRAY' KEY,CS,SN,IST,IIV,BN,PM,FPB,FAP,FME;
'ARRAY' ST,IS,OS,SI,PB,APB,EP,ME;
'ALGOL';
'PROCEDURE' ERRORTRAP(P);
'PROCEDURE' P;
'EXTERNAL';
'PROCEDURE' ERROR(J);
'INTEGER' J;
'IF' J>0 'THEN'
```



```

30 KEYL49J:=J;
32 'GO TO' L7;

32 'END' ERROR;
32 'PROCEDURE' ERROR2(J);
34 'INTEGER' J;
35 'IF' J>0 'THEN' KEY[49]:=J;
35 READ DATA(KEY,CS,NO,SN,IST,ST,IIV,LS,CE,ECE,BN,PM,PB,FPB,APB,FAP,LV,FME,
36 PCP,I7);
37 ERRORTRAP(ERROR);
38 'IF' KEY[37]>0 'THEN'
38 'FOR' I:=1 'STEP' 1 'UNTIL' KEY[27] 'DO'
39 'FOR' J:=1,2 'DO'
40 EP[I,J]:=0;
41 L1:I:=0;
42 L2:I:=I+1;
43 'IF' I>N 'THEN' 'GO TO' L7;
44 'IF' CS[I]=0 'THEN' 'GO TO' L2;
45 KEY[11]:=CS[I];
46 'FOR' J:=2,3 'DO' KEY[J]:=PM[CS[I],J-1];
48 KEY[5]:=PM[CS[I],31];
49 B:=KEY[11]>KEY[27];
50 'IF' B 'THEN' 'GO TO' L2A;
51 'FOR' J:=1 'STEP' 1 'UNTIL' KEY[2] 'DO'
52 'BEGIN'
52 'FOR' K:=1,2 'DO' IIS[J,K]:=PM[CS[I],2*J+K+1];
55 'FOR' K:=3 'STEP' 1 'UNTIL' KEY[13] 'DO' IIS[J,K]:=IST[IIS[J,1],K];
57 'FOR' K:=1 'STEP' 1 'UNTIL' KEY[14] 'DO' IS[J,K]:=ST[IIS[J,1],K]
58 'END' J;
59 'FOR' J:=KEY[2]+1 'STEP' 1 'UNTIL' KEY[15] 'DO'
60 'BEGIN'
60 'FOR' K:=1 'STEP' 1 'UNTIL' KEY[13] 'DO' IIS[J,K]:=0;
63 'FOR' K:=1 'STEP' 1 'UNTIL' KEY[14] 'DO' IS[J,K]:=0
64 'END' J;
65 'FOR' J:=1 'STEP' 1 'UNTIL' KEY[3] 'DO'
66 'BEGIN'
66 'FOR' K:=1,2 'DO' IOS[J,K]:=PM[CS[I],2*(KEY[2]+J)+K+1];
69 'FOR' K:=3 'STEP' 1 'UNTIL' KEY[13] 'DO' IOS[J,K]:=IST[IOS[J,1],K];
71 'FOR' K:=1 'STEP' 1 'UNTIL' KEY[14] 'DO' OS[J,K]:=ST[IOS[J,1],K]
72 'END' J;
73 'FOR' J:=KEY[3]+1 'STEP' 1 'UNTIL' KEY[16] 'DO'
74 'BEGIN'
74 'FOR' K:=1 'STEP' 1 'UNTIL' KEY[13] 'DO' IOS[J,K]:=0;
77 'FOR' K:=1 'STEP' 1 'UNTIL' KEY[14] 'DO' OS[J,K]:=0
78
78 'END' J;
79 L2A:JUMPER(KEY,SN,IST,ST,IIS,IS,IOS,OS,IIV,LS,CE,ECE,SI,BN,PB,APB,EP,LV,
79 NE,PCP);
80 'IF' B 'THEN' 'GO TO' L2B;
81 'FOR' J:=1 'STEP' 1 'UNTIL' KEY[3] 'DO'
82 'FOR' K:=1 'STEP' 1 'UNTIL' IOS[J,3] 'DO'
83 ST[IOS[J,1],K]:=OS[J,K];
84 'FOR' J:=1 'STEP' 1 'UNTIL' KEY[2] 'DO'
85 'FOR' K:=1 'STEP' 1 'UNTIL' IIS[J,3] 'DO'
86 ST[IIS[J,1],K]:=IS[J,K];
87 L2B:'IF' KEY[4]<0 'OR' KEY[7]>0 'THEN' 'GO TO' L7;
88 'IF' KEY[50]=0 'THEN' 'GO TO' L2;
89 'IF' KEY[11]=0 'THEN' 'GO TO' L4;
90 J:=1;
91 L3:'IF' I=NO[J] 'THEN'
91 'BEGIN'
91 I:=NO[J+1]-1;
93 KEY[50]:=0;

94 'GO TO' L2
94 'END';
95 J:=J+2;
96 'IF' J<KEY[11] 'THEN' 'GO TO' L3;
97 L4:KEY[6]:=BN[CS[I]];
98 L7:ERRORTRAP(ERROR2);
99 PRINT DATA(KEY,CS,SN,IST,ST,IS,OS,IIV,SI,BN,PM,PB,FPB,APB,FAP,EP,NE,FME)
99 'END'
99 'END'

```

ZAŁĄCZNIK 2

12/12/46
STATEMENT

0
0
0

09/09/78

COMPILED BY XALM MK. 15C

```
'SEND TO'(MT,SCRATCH TAPE,LIBRAPROGORG,PROGORG)
'OVERLAY SEGMENTS,(VEGA)
'CONTINUE'
```

12/13/07
STATEMENT

0
2
3
3
3
6
7
8
9
10
12
13
14
15
16
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
35
36

09/09/78

COMPILED BY XALM MK. 15C

```
'PROCEDURE' READ KEY(V);
'INTEGER' ARKAY V;
'BEGIN'
'INTEGER' I;
'FOR' I:=8 'STEP' 1 'UNTIL' 17 'DO' V[I]:=READ;
V[13]:=V[13]*1;
V[18]:='IF' V[17]>0 'THEN' 1+READ 'ELSE' 1;
V[19]:=READ;
V[20]:='IF' V[19]>0 'THEN' READ 'ELSE' 1;
'FOR' I:=21,22 'DO' V[I]:=READ;
V[23]:='IF' V[22]>0 'THEN' READ 'ELSE' 1;
V[24]:=READ;
V[25]:='IF' V[24]>0 'THEN' READ 'ELSE' 1;
V[26]:='IF' V[24]>0 'THEN' READ 'ELSE' 0;
'FOR' I:=27,28,30,31,32 'DO' V[I]:=READ;
V[29]:='IF' V[32]=-1 'THEN' READ 'ELSE' 1;
V[30]:=V[30]*1;
'IF' V[30]<5 'THEN' V[30]:=5;
V[33]:=READ;
V[34]:='IF' V[33]>0 'THEN' READ 'ELSE' 1;
V[35]:='IF' V[33]>0 'THEN' READ 'ELSE' 0;
V[36]:='IF' V[35]=-1 'THEN' READ 'ELSE' 1;
V[37]:=READ;
V[38]:=READ;
V[39]:='IF' V[38]>0 'THEN' READ 'ELSE' 1;
V[40]:=READ;
V[41]:='IF' V[40]>0 'THEN' READ 'ELSE' 1;
V[42]:='IF' V[40]>0 'THEN' READ 'ELSE' 0;
V[43]:=READ;
V[44]:='IF' V[43]>0 'THEN' READ 'ELSE' 1;
'FOR' I:=45,46,47,48 'DO' V[I]:=READ;
'FOR' I:=4,6,7,49,50,51,52 'DO' V[I]:=0
'END' READ KEY;
```

SEGMENT READKEY

LENGTH 575

STATEMENT

```

0      'PROCEDURE' READ DATA(KEY,CS,NO,SN,IST,ST,II,V,LS,CE,ECE,BN,PM,PB,FPB,
1      APB,FAP,LV,FME,PCP,EA);
2      'INTEGER' 'ARRAY' KEY,CS,NO,SN,IST,II,V,BN,PM,FPB,FAP,FME;
3      'ARRAY' ST,LS,CE,ECE,PB,APB,LV,PCP;
4      'LABEL' EA;
5      'BEGIN'
6      'INTEGER' D,F,G,I,J,K,L,M,N,P,R,SP,NL,SR;
7      'INTEGER' 'ARRAY' X[1:'IF'KEY[12]>KEY[27]+KEY[28]'AND'KEY[12]>KEY[18]-2
8      'AND'KEY[14]>KEY[43]'THEN'KEY[12]'ELSE''IF'KEY[18]-2>
9      KEY[27]+KEY[28]'AND'KEY[18]-2>KEY[43]'THEN'KEY[18]-2
10     'ELSE''IF'KEY[27]+KEY[28]>KEY[43]'THEN'KEY[27]+KEY[28]
11     'ELSE'KEY[43]];
12     V[1:5],
13     Y,ZL1:(KEY[50]-3)/2,1:2];
14     'PROCEDURE' CREATE(N,S);
15     'VALUE' N;
16     'INTEGER' N;
17     'STRING' S;
18     'EXTERNAL';
19     'PROCEDURE' USE(N,S);
20     'VALUE' N;
21     'INTEGER' N;
22     'STRING' S;
23     'EXTERNAL';
24     'PROCEDURE' INPUT(N,S);
25     'VALUE' N;
26     'INTEGER' N;
27     'STRING' S;
28     'EXTERNAL';
29     'PROCEDURE' DATA SKIP(N);
30     'VALUE' N;
31     'INTEGER' N;
32     'EXTERNAL';
33     'PROCEDURE' KESER(N,B);
34     'VALUE' N,B;
35     'INTEGER' N,B;
36     'EXTERNAL';
37     'PROCEDURE' WRITE BINARY(N,A,S);
38     'VALUE' N;
39     'INTEGER' N;
40     'ARRAY' A;
41     'STRING' S;
42     'EXTERNAL';
43     'PROCEDURE' READ BINARY(N,A,S);
44     'VALUE' N;
45     'INTEGER' N;
46     'ARRAY' A;
47     'STRING' S;
48     'EXTERNAL';
49     'PROCEDURE' KEWIN(N);
50     'VALUE' N;
51     'INTEGER' N;
52     'EXTERNAL';
53     'PROCEDURE' FREEMT(N);
54     'VALUE' N;
55     'INTEGER' N;
56     'EXTERNAL';
57     'PROCEDURE' ABV(V,N);
58     'VALUE' N;
59     'INTEGER' N;
60     'INTEGER' 'ARRAY' V;
61     'BEGIN'
62     'INTEGER' I,M,P,X;

```

```

56 'BEGIN'
56 P:=0;
58 'FOR' I:=2 'STEP' 1 'UNTIL' M 'DO'
59 'IF' V[I-1]>V[I] 'THEN'
59 'BEGIN'
59 X:=V[I-1];
61 V[I-1]:=V[I];
62 V[I]:=X;
63 P:=1
63 'END'
63 'END' M
63 'END' ABV;
63 'PROCEDURE' ABVMI(V,M,NR,LC,UC,T);
65 'VALUE' NR,LC,UC,T;
66 'INTEGER' NR,LC,UC,T;
67 'INTEGER' 'ARRAY' V,M;
68 'BEGIN'
68 'INTEGER' I,N,P,R;
68 'IF' T=1 'THEN' 'FOR' N:=1 'STEP' 1 'UNTIL' NR 'DO' V[N]:=M[N,1];
71 'FOR' N:=NR,N-1 'WHILE' P>0 'DO'
72 'BEGIN'
72 P:=0;
74 'FOR' I:=2 'STEP' 1 'UNTIL' N 'DO'
75 'IF' V[I-1]>V[I] 'THEN'
75 'BEGIN'
75 P:=V[I-1];
77 V[I-1]:=V[I];
78 V[I]:=P;
79 'FOR' R:=LC 'STEP' 1 'UNTIL' UC 'DO'
80 'BEGIN'
80 P:=M[I-1,R];
82 M[I-1,R]:=M[I,R];
83 M[I,R]:=P
83 'END' R;
84 P:=1
84 'END'
84 'END' N
84 'END' ABVMI;
84 'PROCEDURE' ABVMR(V,M,NR,NC,T);
86 'VALUE' NR,NC,T;
87 'INTEGER' NR,NC,T;
88 'INTEGER' 'ARRAY' V;
89 'ARRAY' M;
90 'BEGIN'
90 'INTEGER' I,N,P,R;
90 'REAL' X;
91 'IF' T=1 'THEN' 'FOR' N:=1 'STEP' 1 'UNTIL' NR 'DO' V[N]:=M[N,1];
94 'FOR' N:=NR,N-1 'WHILE' P>0 'DO'
95 'BEGIN'
95 P:=0;
97 'FOR' I:=2 'STEP' 1 'UNTIL' N 'DO'
98 'IF' V[I-1]>V[I] 'THEN'
98 'BEGIN'
98 P:=V[I-1];
100 V[I-1]:=V[I];
101 V[I]:=P;
102 'FOR' R:=1 'STEP' 1 'UNTIL' NC 'DO'
103 'BEGIN'
103 X:=M[I-1,R];
105 M[I-1,R]:=M[I,R];
106 M[I,R]:=X
106 'END' R;
107 P:=1
107 'END'

```

```

107 'END' ABVMR;
107 'PROCEDURE' INV(X,N,V,NE,AN,EB);
109 'VALUE' N,NE,AN;
110 'INTEGER' N,NE,AN;
111 'INTEGER' 'ARKAY' X,V;
112 'LABEL' EB;
113 'BEGIN'
113 'INTEGER' I,J;
113 'FOR' I:=1 'STEP' 1 'UNTIL' NE 'DO'
115 'IF' V[I]#0 'THEN'
115 'BEGIN'
115 'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
117 'IF' V[I]=X[J] 'THEN'
117 'BEGIN'
117 V[I]:=J;
119 'GO TO' ER
119 'END';
120 KEY[49]:=-AN;
121 'GO TO' EB;
122 ER: 'END'
122 'END' INV;
122 'PROCEDURE' INMI(X,N,M,NR,AN,EB);
124 'VALUE' N,NR,AN;
125 'INTEGER' N,NR,AN;
126 'INTEGER' 'ARKAY' X,M;
127 'LABEL' EB;
128 'BEGIN'
128 'INTEGER' I,J;
128 J:=0;
130 'FOR' I:=1 'STEP' 1 'UNTIL' NR 'DO'
131 'BEGIN'
131 ET:J:=J+1;
133 'IF' M[I,1]>X[J] 'THEN'
133 'BEGIN'
133 'IF' J<N 'THEN' 'GO TO' ET;
135 EU:KEY[49]:=-AN;
136 'GO TO' EB
136 'END'
136 'ELSE'
136 'IF' M[I,1]=X[J] 'THEN' M[I,1]:=J 'ELSE' 'GO TO' EU
136 'END' I
136 'END' INMI;
136 'PROCEDURE' INMR(X,N,M,NR,AN,EB);
138 'VALUE' N,NR,AN;
139 'INTEGER' N,NR,AN;
140 'INTEGER' 'ARKAY' X;
141 'ARRAY' M;
142 'LABEL' EB;

143 'BEGIN'
143 'INTEGER' I,J;
143 J:=0;
145 'FOR' I:=1 'STEP' 1 'UNTIL' NR 'DO'
146 'BEGIN'
146 EV:J:=J+1;
148 'IF' M[I,1]>X[J] 'THEN'
148 'BEGIN'
148 'IF' J<N 'THEN' 'GO TO' EV;
150 EW:KEY[49]:=-AN;
151 'GO TO' EB
151 'END'
151 'ELSE'
151 'IF' M[I,1]=X[J] 'THEN' M[I,1]:=J 'ELSE' 'GO TO' EW
151 'END' I
151 'END' INMR;

```

```

152      'BEGIN'
152      'INTEGER' C;
152      EC:C:=NEXTCH;
154      'IF' C=SP 'OR' C=NL 'THEN'
154      'BEGIN'
154      SKIPCH;
156      'GO TO' EC
156      'END';
157      TRC:=C
157      'END' TRC;
157      'PROCEDURE' TRS;
158      'BEGIN'
158      ED:'IF' SR#NEXTCH 'THEN'
159      'BEGIN'
159      SKIPCH;
161      'GO TO' ED
161      'END';
162      SKIPCH;
163      'END' TRS;
163      'PROCEDURE' KV(V,N);
165      'VALUE' N;
166      'INTEGER' N;
167      'INTEGER' 'ARRAY' V;
168      'BEGIN'
168      'INTEGER' J,L;
168      L:=1;
170      'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
171      'BEGIN'
171      'IF' L=1 'THEN'
172      'BEGIN'
172      'IF' TRC=SR 'THEN' L:=0
173      'END';
174      VIJJ:='IF' L=1 'THEN' READ 'ELSE' 0
174      'END' J;
175      TRS
175      'END' RV;
175      'PROCEDURE' KRI(M,I,LC,UC);
177      'VALUE' I,LC,UC;
178      'INTEGER' I,LC,UC;
179      'INTEGER' 'ARRAY' M;
180      'BEGIN'
180      'INTEGER' J,L;
180      L:=1;

182      'FOR' J:=LC 'STEP' 1 'UNTIL' UC 'DO'
183      'BEGIN'
183      'IF' L=1 'THEN'
184      'BEGIN'
184      'IF' TRC=SR 'THEN' L:=0
185      'END';
186      MII,JJ:='IF' L=1 'THEN' READ 'ELSE' 0
186      'END' J;
187      TRS
187      'END' RRI;
187      'PROCEDURE' KRR(M,I,N);
189      'VALUE' I,N;
190      'INTEGER' I,N;
191      'ARRAY' M;
192      'BEGIN'
192      'INTEGER' J,L;
192      L:=1;
194      'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
195      'BEGIN'
195      'IF' L=1 'THEN'
196      'BEGIN'

```



```

197      'END';
198      MCI,JJ:='IF' L=1 'THEN' READ 'ELSE' ,0
198      'END' J;
199      TRS
199      'END' RRR;
199      'PROCEDURE' STG(W,A,ND);
201      'VALUE' ND;
202      'INTEGER' ND;
203      'ARRAY' W,A;
204      'BEGIN'
204      'FOR' I:=1 'STEP' 1 'UNTIL' ND 'DO' W[I]:=V[I];
207      WRITE BINARY(20,W,('3')));
208      WRITE BINARY(20,A,('3')));
209      'GO TO' EX
209      'END' STG;
209      SP:=CODE('(% % )');
211      NL:=CODE('(EL )');
212      SR:=CODE('( * )');
213      K:=KEY[27]+KEY[28];
214      RV(CS,K);
215      'IF' KEY[11]>0 'THEN'
215      'BEGIN'
215      RV(NO,KEY[11]);
217      INV(CS,K,NO,KEY[11],3,EA);
218      'FOR' I:=1 'STEP' 2 'UNTIL' KEY[11] 'DO'
219      'IF' NOL[I]#0 'AND' NOL[I+1]=0 'OR' NOL[I]=0 'AND' NOL[I+1]#0 'THEN'
219      'BEGIN'
219      KEY[49]:=-3;
221      'GO TO' EA
221      'END'
221      'END';
222      'FOR' I:=1 'STEP' 1 'UNTIL' KEY[12] 'DO'
223      'BEGIN'
223      SN[I]:=X[I]:=READ;
225      'IF' SN[I] LE 0 'THEN'
225      'BEGIN'
225      KEY[49]:=-4;
227      'GO TO' EA
227      'END';
228      RRI(IST,I,3,KEY[13]);
229      RRR(ST,I,KEY[14])
229      'END' I;
230      ABVMI(SN,IST,KEY[12],3,KEY[13],0);
231      ABVMR(X,ST,KEY[14],KEY[14],0);
232      'IF' KEY[17]>0 'THEN'
232      'BEGIN'
232      'FOR' I:=1 'STEP' 1 'UNTIL' KEY[17] 'DO'
234      'BEGIN'
234      RRI(IIV,I,2,KEY[18]);
236      IIV[I,1]:=IIV[I,2];
237      IIV[I,2]:=0;
238      'FOR' J:=3 'STEP' 1 'UNTIL' KEY[18] 'DO'
239      'IF' IIV[I,J]>0 'THEN'
239      'BEGIN'
239      IIV[I,2]:=IIV[I,2]+1;
241      X[J-2]:=IIV[I,J]
241      'END'
241      'ELSE'
241      'GO TO' E0;
242      E0:ABV(X,IIV[I,2]);
243      'FOR' J:=1 'STEP' 1 'UNTIL' IIV[I,2] 'DO' IIV[I,J+2]:=X[J]
244      'END' I;
245      ABVMI(X,IIV,KEY[17],1,KEY[18],1);
246      INMI(SN,KEY[12],IIV,KEY[17],11,EA)

```

```

247 'IF' KEY[19]>0 'THEN'
247 'BEGIN'
247 'FOR' I:=1 'STEP' 1 'UNTIL' KEY[19] 'DO' RRR(LS,I,KEY[20]);
250 ABVMR(X,LS,KEY[19],KEY[20],1);
251 INMR(SN,KEY[12],LS,KEY[19],12,EA)
251 'END';
252 'IF' KEY[21]>0 'THEN'
252 'BEGIN'
252 'FOR' I:=1 'STEP' 1 'UNTIL' KEY[21] 'DO' RRR(CE,I,5);
255 ABVMR(X,CE,KEY[21],5,1);
256 INMR(SN,KEY[12],CE,KEY[21],13,EA)
256 'END';
257 'IF' KEY[22]>0 'THEN'
257 'BEGIN'
257 'FOR' I:=1 'STEP' 1 'UNTIL' KEY[22] 'DO' RRR(ECE,I,KEY[23]);
260 ABVMR(X,ECE,KEY[22],KEY[23],1);
261 INMR(SN,KEY[12],ECE,KEY[22],14,EA)
261 'END';
262 'FOR' I:=1 'STEP' 1 'UNTIL' K 'DO'
263 'BEGIN'
263 TRS;
265 BN[I]:=X[I]:=READ;
266 'IF' BN[I] LE 0 'THEN'
266 'BEGIN'
266 KEY[49]:=-16;
268 'GO TO' EA
268 'END';
269 RRI(PM,I,3,KEY[30]);
270 RRR(PB,I,KEY[31])
270 'END' I;
271 ABVM1(BN,PM,K,3,KEY[30],0);
272 ABVMR(X,PB,K,KEY[31],0);
273 'FOR' I:=1 'STEP' 1 'UNTIL' K 'DO'

274 'BEGIN'
274 PM[I,1]:=PMLI,2]:=0;
276 J:=4;
277 E1: 'IF' PM[I,J]>0 'THEN'
277 'BEGIN'
277 PM[I,1]:=PMLI,1]+1;
279 Y[PM[I,1],1]:=PMLI,J];
280 Y[PM[I,1],2]:=PMLI,J+1]
280 'END'
280 'ELSE'
280 'IF' PMLI,J] < 0 'THEN'
280 'BEGIN'
280 PM[I,2]:=PMLI,2]+1;
282 Z[PMLI,2],1]:=-PMLI,J];
283 Z[PMLI,2],2]:=PMLI,J+1]
283 'END';
284 J:=J+2;
285 'IF' J<KEY[30] 'THEN' 'GO TO' E1;
286 ABVM1(X,Y,PMLI,1],1,2,1);
287 ABVM1(X,Z,PMLI,2],1,2,1);
288 INMI(SN,KEY[12],Y,PMLI,1],17,EA);
289 INMI(SN,KEY[12],Z,PMLI,2],17,EA);
290 'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,1] 'DO'
291 'BEGIN'
291 L:=Y[J,1];
293 Y[J,1]:=Y[J,2];
294 Y[J,2]:=L
294 'END' J;
295 'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,2] 'DO'
296 'BEGIN'
296 L:=Z[J,1];

```

```

299 Z[J,2]:=L
299 'END' J;
300 ABVMI(X,Y,PMI,1,1,2,1);
301 ABVMI(X,Z,PMI,2,1,2,1);
302 'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,1] 'DO'
303 'BEGIN'
303 PMI,2*(J+1)J:=Y[J,2];
305 PMI,2*J+3:=Y[J,1]
305 'END' J;
306 L:=2*(PMI,1J+1);
307 'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,2] 'DO'
308 'BEGIN'
308 PMI,2*J+L:=Z[J,2];
310 PMI,2*J+L+1:=Z[J,1]
310 'END' J
310 'END' I;
311 INV(BN,K,CS,K,2,EA);
312 'IF' KEYL32=-1 'THEN'
312 'BEGIN'
312 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL29 'DO'
314 'BEGIN'
314 J:=-5;
316 E2:L:=TRC;
317 'IF' L#SR 'THEN'
317 'BEGIN'
317 J:=J+1;
319 FPBLI,JJ:=L;
320 SKIPCH;

321 'IF' J<0 'THEN' 'GO TO' E2 'ELSE' TRS
321 'END'
321 'ELSE'
321 'BEGIN'
321 'FOR' J:=J+1 'STEP' 1 'UNTIL' 0 'DO' FPBLI,JJ:=SP;
324 SKIPCH
324 'END';
325 X[IJ]:=READ;
326 RRI(FPB,I,1,KEYL31)
326 'END' I;
327 ABVMI(X,FPB,KEYL29,-4,KEYL31,0)
327 'END';
328 'IF' KEYL33>0 'THEN'
328 'BEGIN'
328 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL33 'DO'
330 'BEGIN'
330 TRS;
332 RRR(APB,I,KEYL34)
332 'END' I;
333 ABVMR(X,APB,KEYL33,KEYL34,1);
334 INMR(BN,K,APB,KEYL35,20,EA);
335 'IF' KEYL35=-1 'THEN'
335 'BEGIN'
335 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL36 'DO'
337 'BEGIN'
337 TRS;
339 RRI(FAP,I,1,KEYL34)
339 'END' I;
340 ABVMI(X,FAP,KEYL36,1,KEYL34,1)
340 'END'
340 'END';
341 'IF' KEYL38>0 'THEN'
341 'BEGIN'
341 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL38 'DO' RRR(LV,I,KEYL39);
344 ABVMR(X,LV,KEYL38,KEYL39,1);
345 INMR(BN,K,LV,KEYL38,23,EA)

```



```

346 'IF' KEY[42]=-1 'THEN' RV(FME,KEY[41]);
347 'IF' KEY[43]LE'0 'THEN' 'GO TO' E8;
348 K:=READ;
349 'IF' K=2 'THEN' L:=READ 'ELSE'
349 'IF' K=1 'THEN' L:=KEY[43] 'ELSE'
349 'BEGIN'
349 L:=0;
351 'GO TO' E3
351 'END';
352 'FOR' I:=1 'STEP' 1 'UNTIL' L 'DO'
353 'BEGIN'
353 RRR(PCP,I,KEY[44]);
355 'IF' PCPLI,1LE'0 'THEN'
355 'BEGIN'
355 KEY[49]:=-26;
357 'GO TO' EA
357 'END'
357 'END' I;
358 'IF' K=1 'THEN' 'GO TO' E7;
359 E3:J:=KEY[43]-L;
360 RV(X,J);
361 'FOR' I:=1 'STEP' 1 'UNTIL' J 'DO'
362 'IF' X[I]LE'0 'THEN'

362 'BEGIN'
362 KEY[49]:=-26;
364 'GO TO' EA
364 'END';
365 ABV(X,J);
366 D:=30;
367 F:=60;
368 G:=2*D*F/(D+1);
369 'BEGIN'
369 'ARRAY' LD[1:D,1:F];
369 INPUT(19,('(PHCHP')));
371 M:=-1;
372 R:=0;
373 E5:R:=R+1;
374 N:=X[R]/(D+1)*G;
375 'IF' M#N 'THEN'
375 'BEGIN'
375 RESET(19,N);
377 READ BINARY(19,LD,((PCP)));
377 'END';
378 P:=X[R]-X[R]/(D+1)*D;
379 'FOR' I:=1 'STEP' 1 'UNTIL' KEY[44] 'DO' PCP[L+R,I]:=LD[P,I];
381 'IF' R=J 'THEN' 'GO TO' E6;
382 M:=N;
383 'GO TO' E5
383 'END';
384 E6:REWIND(19);
385 FREEMT(19);
386 E7:'IF' K#3 'THEN' ABVMR(X,PCP,KEY[43],KEY[44],1);
387 E8:'IF' KEY[0]=1 'THEN' INPUT(20,('(ADDATA')));
388 'IF' KEY[8]>1 'THEN'
388 'BEGIN'
388 'IF' KEY[8]=4 'THEN' CREATE(20,('(ADDATA')) 'ELSE'
389 'BEGIN'
389 USE(20,('(ADDATA')));
391 DATA SKIP(20)
391 'END';
392 F:=READ;
393 'FOR' D:=1 'STEP' 1 'UNTIL' F 'DO'
394 'BEGIN'
394 RV(V,5);

```

```

396 'BEGIN';
396 'ARRAY' W[1:5],A[1:V[1]],1:V[2],1:V[3],1:V[4],1:V[5]];
396 'FOR' I:=1 'STEP' 1 'UNTIL' V[1] 'DO';
398 'FOR' J:=1 'STEP' 1 'UNTIL' V[2] 'DO';
399 'FOR' K:=1 'STEP' 1 'UNTIL' V[3] 'DO';
400 'FOR' L:=1 'STEP' 1 'UNTIL' V[4] 'DO';
401 'FOR' M:=1 'STEP' 1 'UNTIL' V[5] 'DO';
402 A[I,J,K,L,M]:=READ;
403 STG(W,A,5);
403 'END';
404 'IF' V[4]>0 'THEN';
404 'BEGIN';
404 'ARRAY' W[1:4],A[1:V[1]],1:V[2],1:V[3],1:V[4]];
404 'FOR' I:=1 'STEP' 1 'UNTIL' V[1] 'DO';
406 'FOR' J:=1 'STEP' 1 'UNTIL' V[2] 'DO';
407 'FOR' K:=1 'STEP' 1 'UNTIL' V[3] 'DO';
408 'FOR' L:=1 'STEP' 1 'UNTIL' V[4] 'DO';
409 A[I,J,K,L]:=KEAD;

410 STG(W,A,4);
410 'END';
411 'IF' V[3]>0 'THEN';
411 'BEGIN';
411 'ARRAY' W[1:3],A[1:V[1]],1:V[2],1:V[3]];
411 'FOR' I:=1 'STEP' 1 'UNTIL' V[1] 'DO';
413 'FOR' J:=1 'STEP' 1 'UNTIL' V[2] 'DO';
414 'FOR' K:=1 'STEP' 1 'UNTIL' V[3] 'DO';
415 A[I,J,K]:=READ;
416 STG(W,A,3);
416 'END';
417 'IF' V[2]>0 'THEN';
417 'BEGIN';
417 'ARRAY' W[1:2],A[1:V[1]],1:V[2]];
417 'FOR' I:=1 'STEP' 1 'UNTIL' V[1] 'DO';
419 'FOR' J:=1 'STEP' 1 'UNTIL' V[2] 'DO';
420 A[I,J]:=READ;
421 STG(W,A,2);
421 'END';
422 'IF' V[1]>0 'THEN';
422 'BEGIN';
422 'ARRAY' W[1:1],A[1:V[1]]];
422 'FOR' I:=1 'STEP' 1 'UNTIL' V[1] 'DO';
424 A[I]:=READ;
425 STG(W,A,1);
425 'END';
426 EX:=END; D;
427 REWIND(20);
428 'IF' KEY[8]=3 'THEN';
428 'BEGIN';
428 FREEMT(20);
430 INPUT(20,('ADDATA')));
430 'END';
430 'END';
431 'IF' KEY[9]=1 'THEN' INPUT(21,('LIBDATA-1')));
432 'IF' KEY[10]=1 'THEN' INPUT(22,('LIBDATA-2')));
433 'IF' KEY[26]=1 OR KEY[42]=-1 'THEN' CREATE(23,('STORE')));
434 FREE INPUT
434 'END' READ DATA;
SEGMENT READDATA LENGTH 5930

```

12/14/37 09/09/78 COMPILED BY XALM MK. 15C

STATEMENT

```

0 'PROCEDURE' PRINT DATA(KEY,CS,SN,IST,ST,IS,OS,IIV,SI,BN,PM,PB,FPB,APB,
1 FAP,EP,ME,FME);
2 'INTEGER' 'ARRAY' KEY,CS,SN,IST,IIV,BN,PM,FPB,FAP,FME;

```

```

4      'BEGIN'
4      'INTEGER' H,I,J,K,L,M,N,P,R,SP,SU,T,U,V,W;
4      'BOOLEAN' CON;
5      'INTEGER' 'ARRAY' DSL1: 'IF' KEY[26]=1 'AND' KEY[51]>0 'THEN' KEY[25]+5
6          'ELSE' 1;
6      DML1: 'IF' KEY[42]=-1 'AND' KEY[52]>0 'THEN' KEY[41]+5 'ELSE'
6          1;
6      FSL1: 'IF' KEY[26]=1 'AND' KEY[51]>0 'THEN' KEY[25] 'ELSE' 1;
6      'ARRAY' DSIL1: 'IF' KEY[26]=1 'AND' KEY[51]>0 'THEN' KEY[25]+5 'ELSE' 1;
7      DML1: 'IF' KEY[42]=-1 'AND' KEY[52]>0 'THEN' KEY[41]+5 'ELSE' 1;
7      'PROCEDURE' ON(N);
9      'VALUE' N;
10     'INTEGER' N;
11     'EXTERNAL';
11     'PROCEDURE' OFF(N);
13     'VALUE' N;
14     'INTEGER' N;
15     'EXTERNAL';
15     'PROCEDURE' READ BINARY(N,A,S);
17     'VALUE' N;
18     'INTEGER' N;
19     'ARRAY' A;
20     'STRING' S;
21     'EXTERNAL';
21     'PROCEDURE' REWIND(N);
23     'VALUE' N;
24     'INTEGER' N;
25     'EXTERNAL';
25     'PROCEDURE' FREEMT(N);
27     'VALUE' N;
28     'INTEGER' N;
29     'EXTERNAL';
29     'PROCEDURE' SE(N,P,R);
31     'VALUE' N;
32     'INTEGER' N,P,R;
33     'BEGIN'
33     P:=N/'10;
35     R:=N-10*P
35     'END' SE;
35     'INTEGER' 'PROCEDURE' ND(N);
37     'VALUE' N;
38     'INTEGER' N;
39     'BEGIN'
39     'INTEGER' G,H;
39     H:=0;
41     G:=1;
42     EA:H:=H+1;
43     G:=10+G;
44     'IF' N/'G#0 'AND' H<8 'THEN' 'GO TO' EA;
45     ND:=H
45     'END' ND;
45     'PROCEDURE' PN(S,N,ND);
47     'VALUE' N,ND;

48     'INTEGER' N,ND;
49     'STRING' S;
50     'BEGIN'
50     'INTEGER' I,K;
50     WRITE TEXT(S);
52     WRITE TEXT('('||I||')');
53     'FOR' I:=ND 'STEP' -1 'UNTIL' 1 'DO'
54     'BEGIN'
54     K:=10+(I-1);
56     K:=N/'K;
57     K:=K-K/'10*10;

```

```

58 'END' I;
59 'IF' CON 'THEN' WRITE TEXT('('I%')') 'ELSE' WRITE TEXT('('I'=')');
60 'END' PN;
60 'PROCEDURE' NL(H,P,R);
62 'VALUE' H;
63 'INTEGER' H,P,R;
64 'BEGIN'
64 H:=H+('IF' P>0'AND'R>0 'THEN' P+R+4 'ELSE' 'IF' P>0'AND'R=0 'THEN' P+3
65 'ELSE' 'IF' P=0'AND'R>0 'THEN' R+9 'ELSE' 0)+8;
66 SU:=SU+H;
67 'IF' SU>125 'THEN'
67 'BEGIN'
67 NEWLINE(1);
69 SU:=H
69 'END'
69 'END' NL;
69 'PROCEDURE' PPH(T,N);
71 'VALUE' T,N;
72 'INTEGER' T,N;
73 'BEGIN'
73 'IF' T=1 'THEN' WRITE TEXT('('GAS')') 'ELSE'
74 'IF' T=2 'THEN' WRITE TEXT('('LIQUID')') 'ELSE' WRITE TEXT('('SOLID')');
75 WRITE TEXT('(' % PHASE ')');
76 PRINT(N,ND(N),0)
76 'END' PPH;
76 'PROCEDURE' PJL(NV,ET);
78 'VALUE' NV;
79 'INTEGER' NV;
80 'LABEL' ET;
81 'BEGIN'
81 'IF' NV>0 'THEN'
82 'BEGIN'
82 'IF' IIVLN,M#NV 'THEN' 'GO TO' ET;
84 'IF' M-2<IIVLN,2 'THEN' M:=M+1
84 'END';
85 P:=P+1;
86 NEWLINE(1);
87 PN('('SI')',P,ND(P))
87 'END' PJL;
87 'PROCEDURE' PST(X,I,N);
89 'VALUE' I,N;
90 'INTEGER' I,N;
91 'ARRAY' X;
92 'BEGIN'
92 'INTEGER' J,K,L,P,R,U;
92 WRITE TEXT('('('SC')' STREAM ')');
94 PRINT(SN[N],ND(SN[N]),0);
95 SE(KEY[45],P,R);

96 WRITE TEXT('('('2C')' T= ')');
97 PRINT(XLI,1],P,R);
98 SE(KEY[46],P,R);
99 WRITE TEXT('('('SS')' P= ')');
100 PRINT(XLI,2],P,R);
101 'FOR' J:=1 'STEPA' 1 'UNTIL' ISTLN,4] 'DO'
102 'BEGIN'
102 NEWLINE(2);
104 PPH(ISTLN,J+3],J);
105 WRITE TEXT('('('C')' N= ')');
106 SE(KEY[47],P,R);
107 SU:=0;
108 NL(-1,P,R);
109 PRINT(XLI,3+(J-1)*(IST[N,5]+1)],P,R);
110 SPACE(5);
111 SE(KEY[48],P,R);

```



```

113      'BEGIN'
113      L:=IST[N,ISTLN,4J+K+5];
115      U:=ND(L);
116      NL(U+1,P,R);
117      PN('('X')',L,U);
118      PRINT(XLI,(J-1)*ISTLN,5J+K+J+2],P,R);
119      'IF' SU<121 'THEN' SPACE(5)
119      'END' K
119      'END' J
119      'END' PST;
119      'PROCEDURE' PPB(X,I,LC,UC,TP,Y,J,S,SL);
121      'VALUE' I,LC,UC,TP,J,SL;
122      'INTEGER' I,LC,UC,TP,J,SL;
123      'INTEGER' 'ARKAY' Y;
124      'ARRAY' X;
125      'STRING' S;
126      'BEGIN'
126      'INTEGER' M,N,P,R;
126      NEWLINE(2);
128      SU:=0;
129      'IF' TP>0 'THEN' SE(TP,P,R);
130      'FOR' N:=LC 'STEP' 1 'UNTIL' UC 'DO'
131      'BEGIN'
131      'IF' TP=-1 'THEN'
132      'BEGIN'
132      'IF' Y[J,N]=0 'THEN' 'GO TO' EB;
134      SE(Y[J,N],P,R)
134      'END';
135      M:=ND(N);
136      NL(SL+M,P,R);
137      PN(S,N,M);
138      PRINT(XLI,N],P,R);
139      'IF' SU<121 'THEN' SPACE(5);
140      EB: 'END' N
140      'END' PPB;
140      'PROCEDURE' PSM(X,I,NC,TP,Y,S);
142      'VALUE' I,NC,TP;
143      'INTEGER' I,NC,TP;
144      'INTEGER' 'ARKAY' Y;
145      'ARRAY' X;
146      'STRING' S;
147      'BEGIN'
147      'INTEGER' N,N,P,R;

147      NEWLINE(2);
149      SU:=0;
150      'IF' TP>0 'THEN' SE(TP,P,R);
151      'FOR' N:=1 'STEP' 1 'UNTIL' NC 'DO'
152      'BEGIN'
152      'IF' TP=-1 'THEN'
153      'BEGIN'
153      'IF' Y[N]=0 'THEN' 'GO TO' EC;
155      SE(Y[N],P,R)
155      'END';
156      M:=ND(N);
157      NL(M+2,P,R);
158      PN(S,N,M);
159      PRINT(XLI,N],P,R);
160      'IF' SU<121 'THEN' SPACE(5);
161      EC: 'END' N
161      'END' PMS;
161      ON(9);
163      CON:='FALSE';
164      SELECT OUTPUT(2);
165      'FOR' I:=8,9,10 'DO'

```

```

166      'BEGIN'
166      REWIND(12+1);
168      FREEMT(12+1)
168      'END';
169      'IF' KEY[49] < 0 'THEN'
169      'BEGIN'
169      WRITE TEXT('('('('10C')' INCORECT % NUMERATION % IN % THE % ARRAY ')');
171      PRINT(-KEY[49],2,0);
172      'GO TO' E17
172      'END';
173      'IF' KEY[6] > 0 'THEN'
173      'BEGIN'
173      WRITE TEXT('('('('10C')' IT IS % UNKNOWN % WHERE % TO % GO % FROM % THE % BLOCK ')');
175      PRINT(KEY[6],3,0);
176      'GO TO' EPL
176      'END';
177      'IF' KEY[7] > 0 'THEN'
177      'BEGIN'
177      WRITE TEXT('('('('10C')' ERROR ')');
179      PRINT(KEY[7],ND(KEY[7]),0);
180      WRITE TEXT('('('('2C')' BLOCK ')');
181      K:=BN[KEY[1]];
182      PRINT(K,ND(K),0);
183      'GO TO' EPL
183      'END';
184      'IF' KEY[4] < 0 'THEN'
184      'BEGIN'
184      WRITE TEXT('('('('10C')' NEGATIVE % NUMBER % IN % KEY[4] % AFTER %
185      CALCULATION % OF % BLOCK ')');
186      K:=BN[KEY[1]];
187      PRINT(K,ND(K),0);
188      'GO TO' EPL
188      'END';
189      'IF' KEY[49] = 0 'THEN' 'GO TO' E1;
190      WRITE TEXT('('('('10C')' ERROR % TYPE ')');
191      PRINT(KEY[49],ND(KEY[49]),0);
192      WRITE TEXT('('('('2C')' BLOCK ')');
193      K:=BN[KEY[1]];

194      PRINT(K,ND(K),0);
195      'IF' KEY[4] > 0 'THEN'
195      'BEGIN'
195      WRITE TEXT('('('('2C')' ECONOMICAL % CALCULATION % CONTROLLED % BY %
196      BLOCK ')');
197      K:=BN[KEY[4]];
198      PRINT(K,ND(K),0);
199      'END';
200      'IF' KEY[2] > 0 'THEN'
200      'BEGIN'
200      WRITE TEXT('('('('10C')' INLET % STREAMS % OF % THE % BLOCK ')');
202      PRINT(K,ND(K),0);
203      'FOR' I:=1 'STEP' 1 'UNTIL' KEY[2] 'DO' PST(1S,I,PMLKEY[1],2*(I+1));
205      'END';
206      'IF' KEY[3] > 0 'THEN'
206      'BEGIN'
206      WRITE TEXT('('('('10C')' OUTLET % STREAMS % OF % THE % BLOCK ')');
208      PRINT(K,ND(K),0);
209      'FOR' I:=1 'STEP' 1 'UNTIL' KEY[3] 'DO'
210      PST(1S,I,PMLKEY[1],2*(KEY[2]+I+1));
211      'END';
212      EPL: 'IF' KEY[26] = 0 'THEN'
212      'BEGIN'
212      WRITE TEXT('('('('10C')' ARRAY % S1 ')');
214      'FOR' I:=1 'STEP' 1 'UNTIL' KEY[24] 'DO'
215      'BEGIN'

```

```

217 PRINT(I,ND(I),0);
218 PSM(SI,1,KEYL25],9,FSI,('SI'))
218 'END' I
218 'END';
219 'IF' KEYL42]#0 'THEN'
219 'BEGIN'
219 WRITE TEXT('('('C10C'))' ARRAY % ME '))';
221 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL40] 'DO'
222 'BEGIN'
222 WRITE TEXT('('('C5C'))' ROW '))';
224 PRINT(I,ND(I),0);
225 PSM(ME,1,KEYL41],KEYL42],FME,('ME'))
225 'END' I
225 'END';
226 E1:WRITE TEXT('('('C10C'))' SYSTEM % STREAMS '('5C')) T=[K] '('C'))'
226 P=[N/M*2] '('C'))' N=[KMOL/S] '('C'))' X=[KMOL/KMOL] '))';
227 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL12] 'DO' PST(ST,I,I);
229 'IF' KEYL32]=0'AND'KEYL35]=0'AND'KEYL37]=0 'THEN' 'GO TO' E6;
230 'IF' KEYL37]>0 'THEN' SE(KEYL37],P,R);
231 SP:=CODE('(' % '))');
232 K:=KEYL27]+KEYL28];
233 WRITE TEXT('('('C10C'))' SYSTEM % BLOCKS '))';
234 'FOR' I:=1 'STEP' 1 'UNTIL' K 'DO'
235 'BEGIN'
235 'FOR' J:=1 'STEP' 1 'UNTIL' K 'DO' 'IF' I=CS[J] 'THEN' 'GO TO' E2;
238 'GO TO' E5;
239 E2:NEWLINE(5);
240 'IF' KEYL32]#-1 'THEN' WRITE TEXT('('BLOCK'))' 'ELSE'
240 'FOR' L:=-4 'STEP' 1 'UNTIL' 0 'DO'
241 'IF' FPB[PMI,3],L]#SP 'THEN' PRINTCH(FPB[PMI,3],L));
242 PRINT(BNLI],ND(BNLI]),0);
243 'IF' KEYL32]#0 'THEN'
243 PPB(pB,1,1,KEYL31],KEYL32],FPB,PMI,3],('pB'))',2);

244 'IF' KEYL35]#0 'THEN'
244 'BEGIN'
244 L:=0;
246 'FOR' J:=1 'STEP' 1 'UNTIL' KEYL33] 'DO'
247 'IF' APB[J,1]=I 'THEN'
247 'BEGIN'
247 L:=J;
249 'IF' KEYL35]>0 'THEN' 'GO TO' E3 'ELSE'
249 'FOR' M:=1 'STEP' 1 'UNTIL' KEYL36] 'DO'
250 'IF' FAP[M,1]=PMI,3] 'THEN'
250 'BEGIN'
250 N:=M;
252 'GO TO' E3
252 'END' M
252 'END' J;
253 'GO TO' E4;
254 E3:PPB(APB,L,2,KEYL34],KEYL35],FAP,N,('APB'))',3)
254 'END';
255 E4:'IF' KEYL37]=0'OR'I>KEYL27] 'THEN' 'GO TO' E5;
256 WRITE TEXT('('('C2C'))' OPERATING % COSTS % = '))';
257 PRINT(EPLI,2],P,R);
258 WRITE TEXT('('('C5S'))' CAPITAL % COSTS % = '))';
259 PRINT(EPLI,3],P,R);
260 E5:'END' I;
261 E6:'IF' KEYL46]=0'OR'KEYL51]=0 'THEN' 'GO TO' E13;
262 K:=0;
263 L:=KEYL25]+5;
264 REWIND(23);
265 'FOR' I:=1 'STEP' 1 'UNTIL' KEYL51] 'DO'
266 'BEGIN'
266

```



```

268 READ BINARY(43,SI,('SI')));
269 'FOR' J:=1 'STEP' 1 'UNTIL' L 'DO' DS[J]:=DSI[J];
271 'IF' K=DS[1] 'THEN' 'GO TO' E12;
272 K:=DS[1];
273 WRITE TEXT(('('('100'))' ARRAY % SI % OF % THE % BLOCK '));
274 PRINT(BN[K],ND(BN[K]),0);
275 'FOR' J:=6 'STEP' 1 'UNTIL' L 'DO'
276 'IF' DSLJ]>0 'THEN'
276 'BEGIN'
276 NEWLINE(5);
278 'GO TO' E6A
278 'END' J;
279 'GO TO' E12;
280 E6A:P:=0;
281 'FOR' J:=6 'STEP' 1 'UNTIL' L 'DO'
282 'IF' DSLJ]>0 'THEN'
282 'BEGIN'
282 N:=0;
284 'IF' KEYL17]>0 'THEN'
284 'FOR' M:=1 'STEP' 1 'UNTIL' KEYL18] 'DO'
285 'IF' DSLJ]=IIV[M,1] 'THEN'
285 'BEGIN'
285 N:=M;
287 'GO TO' E7
287 'END';
288 E7:M:=3;
289 V:=SN[DS[J]];
290 W:=ND(V);

291 PJL(1,E8);
292 WRITE TEXT(('T % OF % THE % STREAM')));
293 PRINT(V,W,0);
294 FSILP]:=KEYL45];
295 E8:PJL(2,E9);
296 WRITE TEXT(('P % OF % THE % STREAM')));
297 PRINT(V,W,0);
298 FSILP]:=KEYL46];
299 E9:'FOR' R:=1 'STEP' 1 'UNTIL' IST[DS[J],4] 'DO'
300 'BEGIN'
300 PJL((R-1)*(IST[DS[J],5]+1)+3,E10);
302 WRITE TEXT(('N % OF % THE % '));
303 T:=IST[DS[J],R+5];
304 PPH(T,R);
305 WRITE TEXT(('IN % THE % STREAM')));
306 PRINT(V,W,0);
307 FSILP]:=KEYL47];
308 E10:'FOR' U:=1 'STEP' 1 'UNTIL' IST[DS[J],5] 'DO'
309 'BEGIN'
309 PJL((R-1)*(IST[DS[J],5]+1)+U+3,E11);
311 H:=IST[DS[J],IST[DSLJ],4]+U+1];
312 CON:='TRUE';
313 PH(('X'),H,ND(H));
314 CON:='FALSE';
315 WRITE TEXT(('OF % THE % '));
316 PPH(T,R);
317 WRITE TEXT(('IN % THE % STREAM')));
318 PRINT(V,W,0);
319 FSILP]:=KEYL48];
320 E11:'END' U
320 'END' R
320 'END' J;
321 E12:T:='IF' DS[5]>0 'THEN' DS[5] 'ELSE' 1;
322 'FOR' J:=1 'STEP' T 'UNTIL' DS[2] 'DO'
323 'BEGIN'
323 WRITE TEXT(('('('SC'))' ITERATION')));

```

```

326 PRINT(H,ND(H),0);
327 PSM(SI,J,DSL3],-1,FSI,('SI'))
327 'END' J
327 'END' I;
328 E13:'IF' KEYL42J=0'OR'KEY[52]=0 'THEN' 'GO TO' E16;
329 K:=0;
330 L:=KEY[41]+5;
331 REWIND(23);
332 'FOR' I:=1 'STEP' 1 'UNTIL' KEY[52] 'DO'
333 'BEGIN'
333 READ BINARY(23,DME,('DME'));
335 READ BINARY(23,ME,('ME'));
336 'FOR' J:=1 'STEP' 1 'UNTIL' L 'DO' DM[J]:=DME[J];
338 'IF' K=DM[1] 'THEN' 'GO TO' E15;
339 K:=DM[1];
340 WRITE TEXT('('('100'))' ARRAY % ME % OF % THE % BLOCK '));
341 PRINT(BN[K],ND(BN[K]),0);
342 'FOR' J:=6 'STEP' 1 'UNTIL' L 'DO'
343 'IF' DM[LJ]>0 'THEN'
343 'BEGIN'
343 NEWLINE(5);
345 'GO TO' E14
345 'END' J;

346 'GO TO' E15;
347 E14:'FOR' J:=6 'STEP' 1 'UNTIL' L 'DO'
348 'IF' DM[LJ]>0 'THEN'
348 'BEGIN'
348 H:=DM[J]/'100000;
350 N:=(DM[J]-H*100000)/'100;
351 P:=DM[J]-100*(H*1000+N);
352 NEWLINE(1);
353 PN('ME'),J-5,ND(J-5));
354 WRITE TEXT('(%'FEATURE'))';
355 PRINT(P,ND(P),0);
356 'IF' H=6 'THEN' WRITE TEXT('('OF % THE % STREAM'))';
356 'ELSE' 'IF' H=18'OR'H=20 'THEN' WRITE TEXT('('OF % THE % BLOCK'))';
357 PRINT(N,ND(N),0);
358 'IF' H=18'OR'H=20 'THEN' WRITE TEXT('('IN % THE % ARRAY % '));
359 'IF' H=18 'THEN' WRITE TEXT('('PB'))' 'ELSE'
359 'IF' H=20 'THEN' WRITE TEXT('('APB'))'
359 'END' J;
360 E15:T:='IF' DM[5J]>0 'THEN' DM[5] 'ELSE' 1;
361 'IF' DM[4J]=-1 'THEN' P:=DM[4J]:=1 'ELSE' P:=0;
362 'FOR' J:=1 'STEP' T 'UNTIL' DM[2] 'DO'
363 'BEGIN'
363 NEWLINE(5);
365 'IF' P=0 'THEN' WRITE TEXT('('ITERATION'))' 'ELSE' WRITE TEXT('('ROW'))';
366 H:=DM[4J]+J-1;
367 PRINT(H,ND(H),0);
368 PSM(ME,J,DM[3],KEY[42],FME,('ME'));
368 'END' J
368 'END' I;
369 E16:'IF' KEYL26J=1'OR'KEY[42]=-1 'THEN'
369 'BEGIN'
369 REWIND(23);
371 FREEMT(23)
371 'END';
372 E17:OFF(9)
372 'END' PRINT DATA;
SEGMENT PRINTDATA LENGTH 5355

12/16/01 09/09/78 COMPILED BY XALM MK. 15C
STATEMENT
0 'FINISH'

```

ZAKŁADNIK 3

08/15/28

14/07/79

COMPILED BY XALT MK. 5A

STATEMENT

```

0  'PROGRAM' (ANALYZER)
0  'PRIORITY' 80
0  'EXTENDED'
0  'BEGIN'
1  'INTEGER' TA, TB, TC, NA, NCPM, NS, P, F, G, H, I, J, K, L, M, N, RER, NL, KR, SP, SR, NI, LP,
1      LS, SU, HNF, S, MLO, LDS, LDO, LVS, LVO, NXD, NXV, LXD, LXV, LXUP, NZ, LZ1,
1      LZ2, MLP, LD, LG, LK, LK1, LL, LM, LMR, N2, LLP;
1  'BOOLEAN' KS, K7, KP;
2  SELECT INPUT(3);
4  TA:=READ;
5  TB:=READ;
6  TC:='IF' TB<3 'THEN' READ 'ELSE' 0;
7  NA:=READ;
8  NCPM:=2+READ;
9  NS:=READ;
10 'BEGIN'
10 'INTEGER' 'ARRAY' BN[1:NA], PN[1:NA, 1:NCPM], Y, Z[1:(NCPM-3)/2, 1:2],
10      SN[1:NS], SL[1:NS, 1:3], X[1:(NCPM-3)/2];
10 'BOOLEAN' 'ARRAY' A[1:'IF' TB=3 'THEN' 1 'ELSE' NA, 1:'IF' TB=3 'THEN' 1 'ELSE' NA];
11 'PROCEDURE' ABV(V, N);
13 'VALUE' N;
14 'INTEGER' N;
15 'INTEGER' 'ARRAY' V;
16 'BEGIN'
16 'INTEGER' I, M, P, X;
16 'FOR' M:=N, N-1 'WHILE' P>0 'DO'
18 'BEGIN'
18 P:=0;
20 'FOR' I:=2 'STEP' 1 'UNTIL' N 'DO'
21 'IF' V[I-1]>V[I] 'THEN'
21 'BEGIN'
21 X:=V[I-1];
23 V[I-1]:=V[I];
24 V[I]:=X;
25 P:=1;
25 'END'
25 'END' M
25 'END' ABV;
25 'PROCEDURE' ABVNI(V, M, NR, LC, UC, T);
27 'VALUE' NR, LC, UC, T;
28 'INTEGER' NR, LC, UC, T;
29 'INTEGER' 'ARRAY' V, M;
30 'BEGIN'
30 'INTEGER' I, N, P, R;
30 'IF' T=1 'THEN' 'FOR' N:=1 'STEP' 1 'UNTIL' NR 'DO' V[N]:=M[N, 1];
33 'FOR' N:=NR, N-1 'WHILE' P>0 'DO'
34 'BEGIN'
34 P:=0;
36 'FOR' I:=2 'STEP' 1 'UNTIL' N 'DO'
37 'IF' V[I-1]>V[I] 'THEN'
37 'BEGIN'
37 P:=V[I-1];
39 V[I-1]:=V[I];
40 V[I]:=P;
41 'FOR' R:=LC 'STEP' 1 'UNTIL' UC 'DO'
42 'BEGIN'
42 P:=M[I-1, R];
44 M[I-1, R]:=M[I, R];
45 M[I, R]:=P
45 'END' R;
46 P:=1;
46 'END'
46 'END' N
46 'END' ABVNI;
46 'PROCEDURE' INMI(X, N, M, NR, AN, ER);
48 'VALUE' N, NR, AN;
49 'INTEGER' N, NR, AN;
50 'INTEGER' 'ARRAY' X, N;
51 'LABEL' ER;
52 'BEGIN'
52 'INTEGER' I, J;
52 J:=0;
54 'FOR' I:=1 'STEP' 1 'UNTIL' NR 'DO'
55 'BEGIN'
55 ET:J:=J+1;
57 'IF' M[I, 1]>X[J] 'THEN'
57 'BEGIN'
57 'IF' J<N 'THEN' 'GO TO' ET;
59 EU:BER:=AN;
60 'GO TO' ER
60 'END'
60 'ELSE'
60 'IF' M[I, 1]=X[J] 'THEN'

```

```

60  N[I,1]:=J;
61  J:=J-1;
62  'END';
62  'ELSE';
62  'GO TO' EU;
62  'END' I;
62  'END' INMI;
62  'INTEGER' 'PROCEDURE' TRC;
63  'BEGIN';
63  'INTEGER' C;
63  EC:C:=NEXTCH;
63  'IF' C=SP 'OR' C=NL 'THEN';
63  'BEGIN';
63  SKIPCH;
67  'GO TO' EC;
67  'END';
68  TRC:=C;
68  'END' TRC;
68  'PROCEDURE' TRS;
69  'BEGIN';
69  ED:'IF' SR#NEXTCH 'THEN';
70  'BEGIN';
70  SKIPCH;
72  'GO TO' ED;
72  'END';
73  SKIPCH;
73  'END' TRS;
73  'PROCEDURE' RRI(M,I,LC,UC);
73  'VALUE' I,LC,UC;
73  'INTEGER' I,LC,UC;
77  'INTEGER' 'ARRAY' M;
78  'BEGIN';
78  'INTEGER' J,L;

78  L:=1;
80  'FOR' J:=LC 'STEP' 1 'UNTIL' UC 'DO';
81  'BEGIN';
81  'IF' L=1 'THEN';
82  'BEGIN';
82  'IF' TRC=SR 'THEN' L:=0;
83  'END';
84  M[I,J]:='IF' L=1 'THEN' READ 'ELSE' 0;
84  'END' J;
85  TRS;
85  'END' RRI;
85  'INTEGER' 'PROCEDURE' ND(N);
87  'VALUE' N;
88  'INTEGER' N;
89  'BEGIN';
89  'INTEGER' G,H;
89  H:=0;
91  G:=1;
92  EA:H:=H+1;
93  G:=10*G;
94  'IF' H/'G#0' AND 'H<8' 'THEN' 'GO TO' EA;
95  ND:=H;
95  'END' ND;
95  'PROCEDURE' BS(P,Q);
97  'VALUE' P,Q;
98  'INTEGER' P,Q;
99  'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,P] 'DO';
100  'BEGIN';
100  'FOR' K:=1 'STEP' 1 'UNTIL' NS 'DO';
102  'IF' PM[I,2*(Q+J+1)]=SN[K] 'THEN';
102  'BEGIN';
102  S[K,2]/'P1:=I;
104  'GO TO' EA;
104  'END';
105  EA:'END' BS;
105  'PROCEDURE' RF(A,P,Q);
107  'VALUE' P,Q;

```



```

109 'INTEGER' 'ARRAY' A;
110 'FOR' K:=1 'STEP' 1 'UNTIL' PM[J,P1] 'DO'
111 'BEGIN'
111 N:=READ;
113 'FOR' L:=1 'STEP' 1 'UNTIL' PM[J,P1] 'DO'
114 'IF' N=PM[J,2*(Q+L+1)+1] 'THEN'
114 'BEGIN'
114 N:=PM[J,2*(Q+L+1)];
116 'GO TO' ED;
116 'END';
117 EB: 'FOR' L:=1 'STEP' 1 'UNTIL' NS 'DO'
118 'IF' N=SN[L] 'THEN' 'GO TO' EC;
119 EC: RRI(A,L,1,MNFS)
119 'END' RF;
119 'PROCEDURE' PATHS(N,A);
121 'VALUE' N;
122 'INTEGER' N;
123 'BOOLEAN' 'ARRAY' A;
124 'BEGIN'
124 'INTEGER' H,I,I1,K;
124 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
126 'BEGIN'

126 I1:=I-1;
128 'FOR' N:=1 'STEP' 1 'UNTIL' I1,I+1 'STEP' 1 'UNTIL' N 'DO'
129 'IF' A[H,I] 'THEN'
129 'FOR' K:=1 'STEP' 1 'UNTIL' N 'DO'
130 'IF' A[I,K] 'THEN' A[H,K]:='TRUE'
130 'END' I
130 'END' PATHS;
130 'PROCEDURE' NLI(P,S);
132 'VALUE' P,S;
133 'INTEGER' P,S;
134 'BEGIN'
134 SU:=SU+P+3;
136 'IF' SU>120 'THEN'
136 'BEGIN'
136 NEWLINE(1);
138 SPACE(S);
139 SU:=S+P+3
139 'END'
139 'END' NLI;
139 'PROCEDURE' ZMB(A,M,N);
141 'VALUE' M,N;
142 'INTEGER' M,N;
143 'BOOLEAN' 'ARRAY' A;
144 'BEGIN'
144 'INTEGER' I,J;
144 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
146 'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
147 A[I,J]:='FALSE'
147 'END' ZMB;
147 'INTEGER' 'PROCEDURE' LKR(A,V,N,SYI);
149 'VALUE' N;
150 'INTEGER' N;
151 'BOOLEAN' SYI;
152 'INTEGER' 'ARRAY' V;
153 'BOOLEAN' 'ARRAY' A;
154 'BEGIN'
154 'INTEGER' I,J,K,L,LK;
154 'IF' SYI 'THEN'
155 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
156 'FOR' J:=I+1 'STEP' 1 'UNTIL' N 'DO'
157 A[I,J]:=A[J,I]:=A[I,J] AND A[J,I];
158 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
159 'BEGIN'
159 A[I,I]:='TRUE';
161 V[I]:=0;
162 'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
163 'IF' A[I,J] 'THEN' V[I]:=V[I]+1
163 'END' I;

```

```

165 LK:=0;
166 I:=1;
167 EP7:L:=1;
168 K:=V[I];
169 EP8:'IF' I#N 'THEN'
169 'BEGIN'
169 I:=I+1;
171 'IF' K=V[I] 'THEN'
171 'BEGIN'
171 L:=L+1;
173 'GO TO' EP8

173 'END'
173 'END';
174 LK:=LK+L/'K';
175 'IF' I<N 'THEN' 'GO TO' EP7;
176 LKR:=LK
176 'END' LKR;
176 'PROCEDURE' WCN(A,V,N,VB,VE,L);
178 'VALUE' N,L;
179 'INTEGER' N,L;
180 'INTEGER' 'ARRAY' V,VB,VE;
181 'BOOLEAN' 'ARRAY' A;
182 'BEGIN'
182 'INTEGER' I,J,K;
182 K:=VB[1]:=0;
184 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
185 'BEGIN'
185 'IF' K>0 'THEN'
186 'FOR' J:=1 'STEP' 1 'UNTIL' VE[K] 'DO'
187 'IF' I=V[J] 'THEN' 'GO TO' EP9;
188 K:=K+1;
189 VE[K]:='IF' K=1 'THEN' 0 'ELSE' VE[K-1];
190 VB[K]:=VE[K]+1;
191 'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
192 'IF' A[I,J] 'THEN'
192 'BEGIN'
192 VE[K]:=VE[K]+1;
194 V[VE[K]]:=J
194 'END';
195 'IF' K=L 'THEN' 'GO TO' EP10;
196 EP9:'END';
197 EP10:'END' WCN;
197 'PROCEDURE' CDV(A,N);
199 'VALUE' N;
200 'INTEGER' N;
201 'INTEGER' 'ARRAY' A;
202 'BEGIN'
202 'INTEGER' I,J;
202 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
204 'FOR' J:=1,2 'DO' A[I,J]:=READ
205 'END' CDV;
205 'PROCEDURE' PNK(N,ND);
207 'VALUE' N,ND;
208 'INTEGER' N,ND;
209 'BEGIN'
209 'INTEGER' I,K;
209 SPACE(1);
211 'FOR' I:=ND 'STEP' -1 'UNTIL' 1 'DO'
212 'BEGIN'
212 K:=10*(I-1);
214 K:=N/'K';
215 K:=K-K/'10*10';
216 OUT BASIC(2,K)
216 'END' I;
217 PRINTCH(K);
218 SPACE(1)
218 'END' PNK;
218 'PROCEDURE' DRS(E);
220 'VALUE' E;
221 'INTEGER' E;

```

```

222 WRITE TEXT('('('C5S')) STREAM '));
224 N2:=Nd(SN[E]);
225 PRINT(SN[E],N2,0);
226 WRITE TEXT('(' : %% '));
227 SU:=H,=N2+17
227 'END' DRS;
227 'PROCEDURE' DRF(F);
229 'VALUE' F;
230 'INTEGER' F;
231 'BEGIN'
231 N2:=Nd(F);
233 NLI(N2,H);
234 PRINT(F,N2,0)
234 'END' DRF;
234 SP:=CODE('('%')));
236 NL:=CODE('(' EL')));
237 SR:=CODE('(' *')));
238 KR:=CODE('(' . '));
239 'FOR' I:=1 'STEP' 1 'UNTIL' NA 'DO'
240 'BEGIN'
240 BN[I]:=READ;
242 'IF' BN[I] LE 0 'THEN'
242 'BEGIN'
242 BER:=16;
244 'GO TO' E45
244 'END';
245 RRI(PM,I,4,NCPM);
246 'END' I;
247 ABVMI(BN,PM,NA,4,NCPM,0);
248 'FOR' I:=1 'STEP' 1 'UNTIL' NA 'DO'
249 'BEGIN'
249 PM[I,1]:=PM[I,2]:=PM[I,3]:=0;
251 J:=4;
252 E1: 'IF' PM[I,J]>0 'THEN'
252 'BEGIN'
252 PM[I,1]:=PM[I,1]+1;
254 Y[PM[I,1],2]:=PM[I,J];
255 Y[PM[I,1],1]:=PM[I,J+1]
255 'END'
255 'ELSE'
255 'IF' PM[I,J]<0 'THEN'
255 'BEGIN'
255 PM[I,2]:=PM[I,2]+1;
257 Z[PM[I,2],2]:=-PM[I,J];
258 Z[PM[I,2],1]:=PM[I,J+1]
258 'END';
259 J:=J+2;
260 'IF' J<NCPM 'THEN' 'GO TO' E1;
261 ABVMI(X,Y,PM[I,1],1,2,1);
262 ABVMI(X,Z,PM[I,2],1,2,1);
263 'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,1] 'DO'
264 'BEGIN'
264 PM[I,2*(J+1)]:=Y[J,2];
266 PM[I,2*J+3]:=Y[J,1]
266 'END' J;
267 L:=2*(PM[I,1]+1);
268 'FOR' J:=1 'STEP' 1 'UNTIL' PM[I,2] 'DO'
269 'BEGIN'
269 PM[I,2*J+L]:=Z[J,2];
271 PM[I,2*J+L+1]:=Z[J,1]

271 'END' J
271 'END' I;
272 'FOR' I:=1 'STEP' 1 'UNTIL' NS 'DO'
273 'BEGIN'
273 SN[I]:=READ;
275 SI[I,1]:=SI[I,2]:=0;
276 SI[I,3]:=READ
276 'END';
277 ABVMI(SN,S,NS,1,3,0);

```



```

279 BEGIN
279 BS(1,0);
281 BS(2,PM[1,1]);
281 END;
282 MNFS:=1;
283 FOR I:=1 STEP 1 UNTIL NS DO
284 IF MNFS<S[I,3] THEN MNFS:=S[I,3];
285 BEGIN
285 INTEGER ARRAY FB,FE[1:IF TA=0 THEN NS ELSE 1,1:IF TA=0 THEN MNFS
285 ELSE 1];
285 IF TA=1 THEN GO TO E3;
287 FOR I:=1 STEP 1 UNTIL NS DO
288 FOR J:=1 STEP 1 UNTIL MNFS DO
289 FB[I,J]:=FE[I,J]:=-1;
290 FOR I:=1 STEP 1 UNTIL NA DO
291 BEGIN
291 M:=READ;
293 FOR J:=1 STEP 1 UNTIL NA DO
294 IF RN[J]=M THEN GO TO E2;
295 E2:RF(FE,1,0);
296 RF(FB,2,PM[J,1]);
296 END;
297 IF TB#2 THEN FREE INPUT;
298 KS:='FALSE';
299 FOR I:=1 STEP 1 UNTIL NS DO
300 FOR J:=1 STEP 1 UNTIL S[I,3] DO
301 IF FB[I,J]=0 AND FE[I,J]=0 THEN
301 BEGIN
301 IF NOT KS THEN
302 BEGIN
302 SELECT OUTPUT(2);
304 WRITE TEXT('('('TOC') THE % DIRECTIONS % OF % INFORMATION % FLOW %
304 ARE % IN % CONTRADICTION % FOR % : '('('C')')');
305 KS:='TRUE';
305 END;
306 WRITE TEXT('('('CSS') THE % FEATURE ')');
307 PRINT(J,ND(J),0);
308 WRITE TEXT('(' OF % THE % STREAM ')');
309 PRINT(SN[I],ND(SN[I]),0)
309 END;
310 IF KS THEN GO TO E46;
311 E3:IF TB=3 THEN GO TO E43;
312 ZMB(A,NA,NA);
313 IF TA=1 THEN
313 BEGIN
313 FOR I:=1 STEP 1 UNTIL NS DO
315 IF S[I,1]>0 AND S[I,2]>0 THEN A[S[I,1],S[I,2]]:='TRUE'
315 END
315 ELSE
315 FOR I:=1 STEP 1 UNTIL NS DO
316 BEGIN
316 KZ:=KP:='FALSE';
318 FOR J:=1 STEP 1 UNTIL S[I,3] DO
319 BEGIN
319 IF KZ THEN GO TO E4;
321 IF FB[I,J]=0 AND FE[I,J]=1 THEN A[S[I,1],S[I,2]]:=KZ:='TRUE';
322 E4:IF KP THEN GO TO E5;
323 IF FB[I,J]=1 AND FE[I,J]=0 THEN A[S[I,2],S[I,1]]:=KP:='TRUE';
324 E5:IF KZ AND KP THEN GO TO E6;
325 END J;
326 E6:END I;
327 NI:=0;
328 FOR I:=1 STEP 1 UNTIL NA DO
329 FOR J:=1 STEP 1 UNTIL NA DO
330 IF A[I,J] THEN NI:=NI+1;
331 BEGIN
331 INTEGER ARRAY T[1:NI,1:3],B[1:NA];
331 PROCEDURE BA(SYL,DO);
333 BOOLEAN SYL,DO;
334 BEGIN

```

```

334 ZMB(A,NA,NA);
336 'FOR' I:=1 'STEP' 1 'UNTIL' NI 'DO'
337 A[T[I,1],T[I,2]]:=TRUE;
338 'IF' SYL 'THEN'
338 'FOR' I:=1 'STEP' 1 'UNTIL' NI 'DO'
339 A[T[I,2],T[I,1]]:=TRUE;
340 'IF' DO 'THEN'
340 'FOR' I:=1 'STEP' 1 'UNTIL' NA 'DO'
341 A[I,I]:=TRUE;
341 'END' BA;
341 K:=0;
343 'FOR' I:=1 'STEP' 1 'UNTIL' NA 'DO'
344 'FOR' J:=1 'STEP' 1 'UNTIL' NA 'DO'
345 'IF' A[I,J] 'THEN'
345 'BEGIN'
345 K:=K+1;
347 T[K,1]:=I;
348 T[K,2]:=J;
349 T[K,3]:=0
349 'END';
350 PATHS(NA,A);
351 LP:=LKR(A,B,NA,'TRUE');
352 MLO:=B[NA];
353 'BEGIN'
353 'INTEGER' 'ARRAY' P,R[1:LP];
353 'PROCEDURE' KONTURY(R,W);
353 'BOOLEAN' W;
356 'PROCEDURE' R;
357 'BEGIN'
357 'INTEGER' D,I,J,K,K1,K2,K3,K4,N1,N2,SJ;
357 'INTEGER' 'ARRAY' P,S[1:N];
358 N1:=-N-1;
360 N2:=N1-1;
361 'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
362 S[I]:=0;
363 K2:=1;
364 SCAN:'FOR' K3:=S[K2] 'WHILE' (K3=N2'OR'K3=N1)'AND'K2<N 'DO'
365 'BEGIN'
365 'IF' K3=N1 'THEN' S[K2]:=N2;

367 K2:=K2+1
367 'END';
368 K3:=S[K2];
369 'FOR' K:=K2+1 'STEP' 1 'UNTIL' N 'DO'
370 'IF' S[K]=N1 'THEN' S[K]:=N2;
371 'IF' K3=0 'THEN'
371 'BEGIN'
371 I:=1;
373 K1:=P[1]:=K2;
374 CD:I:=I+1;
375 CD1:'FOR' J:=ABS(S[K2])+1 'STEP' 1 'UNTIL' N 'DO'
376 'BEGIN'
376 SJ:=S[J];
378 'IF' SJ#N2 'THEN'
378 'BEGIN'
378 'IF' A[B[G+J],B[G+K2]]'AND'(K3=0'OR'SJ=N1'OR'SJ>0) 'THEN'
379 'BEGIN'
379 S[K2]:='IF' K3=0 'THEN' J 'ELSE' -J;
381 'IF' SJ=N1 'THEN'
381 'BEGIN'
381 'IF' K3=0 'THEN' K3:=K2;
383 S[J]:=0
383 'END';
384 'IF' S[J]>0 'THEN'
384 'BEGIN'
384 D:=K4:=0;
386 K:=I;
387 'FOR' K:=K-1 'WHILE' K4#J 'DO'
388 'BEGIN'
388 K4:=P[K];
390 D:=D+1

```

```

391 'IF' D 'THEN' R(K4,D) 'ELSE'
391 'BEGIN'
391 H:=K4:=0;
393 K:=1;
394 'IF' D=1 'THEN' M:=M+1 'ELSE' L:=L+1;
395 R(J,D);
396 'IF' D>1 'THEN' 'FOR' K:=K-1 'WHILE' K4#J 'DO'
397 'BEGIN'
397 K4:=P[K];
399 R(K4,D)
399 'END'
399 'END';
400 'GO TO' CD1
400 'END'
400 'ELSE'
400 'BEGIN'
400 K2:=P[I]:=J;
402 'GO TO' CD
402 'END'
402 'END'
402 'END'
402 'END';
403 S[K2]:=N1;
404 'IF' K2#K1 'THEN'
404 'BEGIN'
404 I:=I-1;
406 K2:=P[I-1];
407 'IF' K3=K2 'THEN' K3:=0;

408 'GO TO' CD1
408 'END';
409 'GO TO' SCAN
409 'END'
409 'END' KONTURY;
409 'PROCEDURE' GT(K4,D);
411 'INTEGER' K4,D;
412 'IF' D=1 'THEN' LK1:=LK1+1 'ELSE' LK:=LK+1;
412 MCN(A,B,NA,P,R,LP);
414 SELECT OUTPUT(2);
415 WRITE TEXT('('('10C')' THE % LIST % OF % SUBSYSTEMS ')');
416 'FOR' I:=1 'STEP' 1 'UNTIL' LP 'DO'
417 'BEGIN'
417 WRITE TEXT('('('5C')' SUBSYSTEM ')');
419 PRINT(I,ND(I),0);
420 WRITE TEXT('('('2C5S')' BLOCK ')');
421 'IF' R[I]>P[I] 'THEN' WRITE TEXT('(' S')');
422 WRITE TEXT('(' % : % ')');
423 H:=SU:=15;
424 'FOR' J:=P[I] 'STEP' 1 'UNTIL' R[I] 'DO'
425 DRF(BN[B[J]])
425 'END' I;
426 'IF' TB=1 'THEN'
426 'BEGIN'
426 LDS:=LDO:=LVS:=LVO:=NXD:=NXV:=N2:=1;
428 'GO TO' E9
428 'END';
429 LDS:=READ;
430 LDO:=READ;
431 LVS:=READ;
432 LVO:=READ;
433 E9,N2:='IF' LP<LDS 'THEN' LDS 'ELSE' LP;
434 N2:='IF' N2<LDO 'THEN' LDO 'ELSE' N2;
435 N2:='IF' N2<LVS 'THEN' LVS 'ELSE' N2;
436 N2:='IF' N2<LVO 'THEN' LVO 'ELSE' N2;
437 'BEGIN'
437 'INTEGER' 'ARRAY' DS[1:'IF' LDS>0 'THEN' LDS 'ELSE' 1,1:2],C[1:LP],
437 DO[1:'IF' LDO>0 'THEN' LDO 'ELSE' 1,1:2],FIX[1:N2],
437 VS[1:'IF' LVS>0 'THEN' LVS 'ELSE' 1,1:2],
437 VO[1:'IF' LVO>0 'THEN' LVO 'ELSE' 1,1:2];
437 'BOOLEAN' 'ARRAY' DI[1:LP,1:LP];
438 'PROCEDURE' BD:

```

```

439 'INTEGER' I,J;
439 ZMB(D,LP,LP);
441 'FOR' I:=1 'STEP' 1 'UNTIL' LP 'DO'
442 'FOR' J:=1 'STEP' 1 'UNTIL' LP 'DO'
443 'IF' A[B[P[I]],B[P[J]]] 'THEN' D[I,J]:='TRUE'
443 'END' BD;
443 'PROCEDURE' NLKS;
444 'IF' 'NOT' KS 'THEN'
444 'BEGIN'
444 NEWLINE(8);
446 KS:='TRUE';
446 'END' NLKS;
446 'PROCEDURE' DRT(NSO,NRC,TSO);
448 'VALUE' NSO,NRC,TSO;
449 'INTEGER' NSO,NRC,TSO;
450 'BEGIN'
450 WRITE TEXT('('('('('C')' THE % FEATURE '))');

452 PRINT(NRC,ND(NRC),0);
453 'IF' TSO=1 'THEN' WRITE TEXT('('(' OF % THE % STREAM '))')
453 'ELSE' WRITE TEXT('('(' OF % THE % OBJECT '))');
454 PRINT(NSO,ND(NSO),0)
454 'END' DRT;
454 'PROCEDURE' DDV(TDV);
456 'INTEGER' TDV;
457 'IF' TDV=1 'THEN' WRITE TEXT('('(' IS % AN % APPARENT % DECISION %
457 VARIABLE % BECAUSE % DOES % NOT % INFLUENCE % THE % GOAL %
457 FUNCTION % VALUE '))')
457 'ELSE' WRITE TEXT('('(' EMPLOYED % FOR % CALCULATION % OF %
457 GOAL % FUNCTION % VALUE % IS % INDEPENDEND % OF % DECISION %
457 VARIABLES '))');
457 'PROCEDURE' KDV(M,LDV,NT);
459 'VALUE' LDV,NT;
460 'INTEGER' LDV,NT;
461 'INTEGER' 'ARRAY' M;
462 'BEGIN'
462 'INTEGER' I,K,L;
462 'BOOLEAN' WA,KB;
463 'PROCEDURE' DRB(TY);
465 'INTEGER' TY;
466 'BEGIN'
466 NLKS;
468 WRITE TEXT('('('('('C')' THE % FEATURE '))');
469 PRINT(L,ND(L),0);
470 'IF' TY=3 'THEN' WRITE TEXT('('(' IS % NOT % INVOLVED % IN '))')
470 'ELSE' WRITE TEXT('('(' OF '))');
471 WRITE TEXT('('(' % THE % STREAM '))');
472 PRINT(SN[K],ND(SN[K]),0);
473 'IF' TY=1 'THEN' WRITE TEXT('('(' IS % NOT % ALLOWED % TO % BE % A %
473 DECISION % VARIABLE '))') 'ELSE'
473 'IF' TY=2 'THEN' WRITE TEXT('('(' IS % NOT % ALLOWED % TO % BE % USED %
473 FOR % CALCULATION % OF % GOAL % FUNCTION % VALUE '))')
473 'END' DRB;
473 'FOR' I:=1 'STEP' 1 'UNTIL' LDV 'DO'
475 'BEGIN'
475 K:=MLI,1];
477 L:=MLI,2];
478 WA:='IF' TA=1 'THEN' S[K,1]=0 'AND' S[K,2]>0 'ELSE'
478 FB[K,L]=-1 'AND' FF[K,L]=1 'OR' FB[K,L]=1 'AND' FE[K,L]=-1 'OR'
478 FB[K,L]=FE[K,L];
479 'IF' S[K,3]<L 'THEN' DRB(3) 'ELSE'
479 'IF' 'IF' NT=1 'THEN' 'NOT' WA 'ELSE' WA 'THEN' DRB(NT)
479 'END'
479 'END' KDV;
479 'INTEGER' 'PROCEDURE' SP(N);
481 'VALUE' N;
482 'INTEGER' N;
483 'BEGIN'
483 'INTEGER' I,J;
483 'FOR' I:=1 'STEP' 1 'UNTIL' LP 'DO'
485 'FOR' J:=P[I] 'STEP' 1 'UNTIL' R[I] 'DO'

```



```

487 ERA:SP:=I
487 'END' SP;
487 'PROCEDURE' SDV(DV,LDV,TSO);
489 'VALUE' LDV,TSO;
490 'INTEGER' LDV,TSO;
491 'INTEGER' 'ARRAY' DV;

492 'BEGIN'
492 'INTEGER' K,L,H;
492 'FOR' K:=1 'STEP' 1 'UNTIL' LDV 'DO'
494 'BEGIN'
494 'FOR' L:=E 'STEP' 1 'UNTIL' N 'DO'
496 'FOR' H:=P[C[L]] 'STEP' 1 'UNTIL' R[C[L]] 'DO'
497 'IF' 'IF' TSO=1 'THEN' S[DV[K,1],1]=B[H] 'OR' S[DV[K,1],2]=B[H] 'ELSE'
497 DV[K,1]=B[H] 'THEN'
497 'BEGIN'
497 DRT('IF' TSO=1 'THEN' SN[DV[K,1]] 'ELSE' BN[DV[K,1],DV[K,2],TSO);
499 'GO TO' ES
499 'END';
500 ES: 'END'
500 'END' SDV;
500 'IF' TB=1 'THEN' 'GO TO' E10A;
502 CDV(DS,LDS);
503 CDV(DO,LDO);
504 CDV(VS,LVS);
505 CDV(VO,LVO);
506 FREE INPUT;
507 ABVMI(FIX,DS,LDS,1,2,1);
508 ABVMI(FIX,DO,LDO,1,2,1);
509 ABVMI(FIX,VS,LVS,1,2,1);
510 ABVMI(FIX,VO,LVO,1,2,1);
511 INMI(SN,NS,DS,LDS,1,E45);
512 INMI(BN,NA,DO,LDO,2,E45);
513 INMI(SN,NS,VS,LVS,3,E45);
514 INMI(BN,NA,VO,LVO,4,E45);
515 KS:='FALSE';
516 'IF' LDS=0 'AND' LDO=0 'THEN'
516 'BEGIN'
516 WRITE TEXT('('('10C') 'THE % DECISION % VARIABLES % WERE % NOT %
517 DEFINED ')');
518 KS:='TRUE';
518 'END';
519 'IF' LVS=0 'AND' LVO=0 'THEN'
519 'BEGIN'
519 NLKS;
521 WRITE TEXT('('('2C') 'THERE % WERE % NOT % DEFINED % THE % FEATURES %
521 WHICH % ARE % USED % FOR % CALCULATION % OF % GOAL %
521 FUNCTION % VALUE ')');
521 'END';
522 'IF' KS 'THEN' 'GO TO' E46;
523 KDV(DS,LDS,1);
524 KDV(VS,LVS,2);
525 'IF' KS 'THEN' 'GO TO' E46;
526 NZ:=LVV:=LXD:=LXUP:=LZ1:=LZ2:=0;
527 'IF' TA=0 'THEN'
527 'FOR' I:=1 'STEP' 1 'UNTIL' LDS 'DO'
528 'BEGIN'
528 K:=DS[I,1];
530 L:=DS[I,2];
531 'IF' FB[K,L]=FE[K,L] 'THEN' NZ:=NZ+1
531 'END';
532 NXD:=LDS+LDO+NZ;
533 NXV:=LVS+LVO;
534 E10A: 'BEGIN'
534 'INTEGER' 'ARRAY' XD[1:NXD],XV[1:NXV],XUP[1:'IF' TB=2 'THEN' LP 'ELSE' 1],
534 Z1,Z2[1:'IF' NZ>0 'THEN' NZ 'ELSE' 1];
534 'IF' TB=1 'THEN' 'GO TO' E26;

536 'FOR' I:=1 'STEP' 1 'UNTIL' LVO 'DO'
537 'BEGIN'
537 'FOR' J:=1 'STEP' 1 'UNTIL' LVV 'DO'

```

```

540 LXV:=LXV+1;
541 XVELXV:=VO[I,1];
542 E11:'END';
543 'FOR' I:=1 'STEP' 1 'UNTIL' LVS 'DO'
544 'BEGIN'
545 K:=VS[I,1];
546 L:='IF' TA=1 'THEN' 1 'ELSE' 'IF' FB[K,VS[I,2]]=0 'THEN' 1 'ELSE' 2;
547 'FOR' J:=1 'STEP' 1 'UNTIL' LXV 'DO'
548 'IF' S[K,L]=XV[J] 'THEN' 'GO TO' E12;
549 LXV:=LXV+1;
550 XVELXV:=S[K,L];
551 E12:'END';
552 'FOR' I:=LXV+1 'STEP' 1 'UNTIL' NXV 'DO'
553 XV[I]=0;
554 ABV(XV,LXV);
555 BA('FALSE','TRUE');
556 PATHS(NA,A);
557 'FOR' I:=1 'STEP' 1 'UNTIL' LDS 'DO'
558 'BEGIN'
559 K:=DS[I,1];
560 K2:='IF' TA=1 'THEN' 'FALSE' 'ELSE' FB[K,DS[I,2]]=FE[K,DS[I,2]];
561 L:='IF' TA=1 'THEN' 2 'ELSE' 'IF' FB[K,DS[I,2]]=1 'THEN' 1 'ELSE' 2;
562 E13:'FOR' J:=1 'STEP' 1 'UNTIL' LXD 'DO'
563 'IF' S[K,L]=XD[J] 'THEN' 'GO TO' E16;
564 'FOR' J:=1 'STEP' 1 'UNTIL' LXV 'DO'
565 'IF' A[S[K,L],XV[J]] 'THEN' 'GO TO' E14;
566 'IF' K2 'THEN' 'GO TO' E15;
567 NLKS;
568 DRT(SN[K],DS[I,2],1);
569 DDV(1);
570 'GO TO' E16;
571 E14:LXD:=LXD+1;
572 XD[LXD]:=S[K,L];
573 'IF' K2 'THEN'
574 E15:'BEGIN'
575 K2:='FALSE';
576 L:=2;
577 'GO TO' E13
578 'END';
579 E16:'END';
580 'FOR' I:=1 'STEP' 1 'UNTIL' LDO 'DO'
581 'BEGIN'
582 K:=DO[I,1];
583 'FOR' J:=1 'STEP' 1 'UNTIL' LXD 'DO'
584 'IF' K=XD[J] 'THEN' 'GO TO' E18;
585 'FOR' J:=1 'STEP' 1 'UNTIL' LXV 'DO'
586 'IF' A[K,XV[J]] 'THEN' 'GO TO' E17;
587 NLKS;
588 DRT(BN[K],DO[I,2],2);
589 DDV(1);
590 'GO TO' E18;
591 E17:LXD:=LXD+1;
592 XD[LXD]:=K;
593 E18:'END';
594 'FOR' I:=LXD+1 'STEP' 1 'UNTIL' NXD 'DO'
595 XD[I]=0;

596 ABV(XD,LXD);
597 'FOR' I:=1 'STEP' 1 'UNTIL' LVS 'DO'
598 'BEGIN'
599 K:=VS[I,1];
600 L:='IF' TA=1 'THEN' 1 'ELSE' 'IF' FB[K,VS[I,2]]=0 'THEN' 1 'ELSE' 2;
601 'FOR' J:=1 'STEP' 1 'UNTIL' LXD 'DO'
602 'IF' A[XD[J],S[K,L]] 'THEN' 'GO TO' E19;
603 NLKS;
604 DRT(SN[K],VS[I,2],1);
605 DDV(2);
606 E19:'END';
607 'FOR' I:=1 'STEP' 1 'UNTIL' LVO 'DO'
608 'BEGIN'
609 'FOR' J:=1 'STEP' 1 'UNTIL' LXD 'DO'

```

```

609 NLKS;
610 DRT(BN[VO[I,1]],VO[I,2],2);
611 DDV(2);
612 E20:'END';
613 'IF' KS 'THEN' 'GO TO' E46;
614 'FOR' I:=1 'STEP' 1 'UNTIL' LP 'DO'
615 'BEGIN'
616 L:=B[LP[I]];
617 'FOR' J:=1 'STEP' 1 'UNTIL' LX0 'DO'
618 'IF' A[XD[J]],L1 'THEN'
619 'FOR' K:=1 'STEP' 1 'UNTIL' LXV 'DO'
620 'IF' A[L,XV[K]] 'THEN'
621 'BEGIN'
622 LXUP:=LXUP+1;
623 XUP[LXUP]:=I;
624 'GO TO' E21
625 'END';
626 E21:'END';
627 'FOR' I:=LXUP+1 'STEP' 1 'UNTIL' LP 'DO'
628 XUP[I]:=0;
629 ZMB(D,LP,LP);
630 'IF' TA=0 'THEN'
631 'FOR' I:=1 'STEP' 1 'UNTIL' LDS 'DO'
632 'BEGIN'
633 K:=DS[I,1];
634 L:=DS[I,2];
635 'IF' FB[K,L]#FF[K,L] 'THEN' 'GO TO' E23;
636 'FOR' M:=1 'STEP' 1 'UNTIL' LX0 'DO'
637 'IF' S[K,1]=XD[M] 'THEN'
638 'FOR' N:=1 'STEP' 1 'UNTIL' LX0 'DO'
639 'IF' S[K,2]=XD[N] 'THEN' 'GO TO' E22;
640 LZ1:=LZ1+1;
641 Z1[LZ1]:=K;
642 'GO TO' E23;
643 E22:LZ2:=LZ2+1;
644 Z2[LZ2]:=K;
645 M:=SP(S[K,1]);
646 N:=SP(S[K,2]);
647 D[M,N1:=D[N,M]]:='TRUE';
648 E23:'END';
649 'IF' LXUP=1 'THEN' 'GO TO' E26;
650 'FOR' I:=1 'STEP' 1 'UNTIL' LXUP 'DO'
651 'FOR' J:=1 'STEP' 1 'UNTIL' LXUP 'DO'
652 'IF' A[B[P[XUP[I]]],B[P[XUP[J]]]] 'THEN'
653 D[XUP[I],XUP[J]]:=D[XUP[J],XUP[I]]:='TRUE';
654
655 PATHS(LP,D);
656 N:=LKR(D,C,LP,'TRUE');
657 'IF' LP-LXUP+1=N 'THEN' 'GO TO' E26;
658 M:=N-LP+LXUP;
659 WRITE TEXT('('('10C') THE % OPTIMIZATION % TASK % CONSIDERED %
660 CONSISTS % IN % FACT % OF ')');
661 PRINT(M,ND(M),0);
662 WRITE TEXT('(' MUTUALLY % INDEPENDENT % OPTIMIZATION % TASKS ')');
663 N:=0;
664 'FOR' I:=1 'STEP' 1 'UNTIL' M 'DO'
665 'BEGIN'
666 WRITE TEXT('('('5C') THE % OPTIMIZATION % TASK ')');
667 PRINT(I,ND(I),0);
668 WRITE TEXT('('('3C') THE % DECISION % VARIABLES % : '('C')')');
669 'FOR' J:=1 'STEP' 1 'UNTIL' LXUP 'DO'
670 'BEGIN'
671 'FOR' K:=1 'STEP' 1 'UNTIL' N 'DO'
672 'IF' XUP[J]=C[K] 'THEN' 'GO TO' E24;
673 'GO TO' E25;
674 E24:'END';
675 E25:E:=N+1;
676 'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
677 'IF' D[XUP[J],XUP[K]] 'THEN'
678 'BEGIN'
679 N:=N+1;

```



```

672      'END';
673      G:=0;
674      SDV(DS,LDS,1);
675      SDV(DO,LDO,2);
676      WRITE TEXT('(','(','(3C)'); THE % FEATURES % EMPLOYED % FOR % CALCULATION %
677      OF % GOAL % FUNCTION % VALUE % : '(','(','(C)')')');
678      G:=0;
679      SDV(VS,LVS,1);
680      SDV(VO,LVO,2);
681      'END';
682      'GO TO' E46;
683      E26:BA('TRUE','TRUE');
684      'IF' TB=1 'THEN' 'GO TO' E26B;
685      'FOR' I:=1 'STEP' 1 'UNTIL' LZ1 'DO'
686      A[S[Z1[I],1],S[Z1[I],2]]:=A[S[Z1[I],1],S[Z1[I],1]]='TRUE';
687      'FOR' I:=1 'STEP' 1 'UNTIL' LZ2 'DO'
688      A[S[Z2[I],1],S[Z2[I],2]]:=A[S[Z2[I],1],S[Z2[I],1]]='TRUE';
689      E26B:PATHS(NA,A);
690      BD;
691      LS:=LKR(D,C,LP,'FALSE');
692      MLP:=C[LP];
693      'BEGIN'
694      'INTEGER' 'ARRAY' U,V[1:LS],UP[1:MLP,1:2];
695      WCN(D,C,LP,U,V,LS);
696      BA('FALSE','TRUE');
697      'IF' TB=2 'THEN'
698      'FOR' I:=1 'STEP' 1 'UNTIL' LZ1 'DO'
699      'BEGIN'
700      'FOR' J:=1 'STEP' 1 'UNTIL' LX0 'DO'
701      'IF' S[Z1[I],1]=XD[J] 'THEN'
702      'BEGIN'
703      A[S[Z1[I],1],S[Z1[I],2]]:=A[S[Z1[I],1],S[Z1[I],1]]='TRUE';
704      'GO TO' E27
705      'END';
706      A[S[Z1[I],2],S[Z1[I],1]]:=A[S[Z1[I],2],S[Z1[I],2]]='TRUE';
707      E27:'END';
708      PATHS(NA,A);
709      BD;
710      'IF' TB=2 'THEN'
711      'FOR' I:=1 'STEP' 1 'UNTIL' LP 'DO'
712      'FOR' J:=1 'STEP' 1 'UNTIL' LP 'DO'
713      'IF' 'NOT' (D[I,J] 'OR' D[J,I]) 'THEN'
714      'BEGIN'
715      KP:=KS:='FALSE';
716      'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
717      'IF' I=XUP[K] 'THEN'
718      'BEGIN'
719      KP:='TRUE';
720      'GO TO' E28
721      'END';
722      E28:'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
723      'IF' J=XUP[K] 'THEN'
724      'BEGIN'
725      KS:='TRUE';
726      'GO TO' E29
727      'END';
728      E29:'IF' 'NOT' KP 'AND' KS 'THEN'
729      'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
730      'IF' D[I,XUP[K]] 'THEN'
731      'BEGIN'
732      D[I,J]:='TRUE';
733      'GO TO' E30
734      'END';
735      'IF' KP 'AND' 'NOT' KS 'THEN'
736      'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
737      'IF' D[XUP[K],J] 'THEN'
738      'BEGIN'
739      D[I,J]:='TRUE';
740      'GO TO' E30
741      'END';

```

```

724 'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
725 'IF' D[I,XUP[K]] 'THEN'
725 'FOR' L:=1 'STEP' 1 'UNTIL' LXUP 'DO'
726 'IF' D[XUP[L],J] 'THEN'
726 'BEGIN'
726 D[I,J]:='TRUE';
728 'GO TO' E30
728 'END';
729 E30:'END';
730 PATHS(LP,D);
731 BA('FALSE','FALSE');
732 WRITE TEXT('('('10C')) THE % SEQUENCE % OF % SUBSYSTEM % CALCULATION
732 % : '));
733 LD:=LG:=0;
734 'FOR' I:=1 'STEP' 1 'UNTIL' LS 'DO'
735 'BEGIN'
735 H:=0;
737 L:=V[I]-U[I]+1;
738 'FOR' J:=1 'STEP' 1 'UNTIL' L 'DO'
739 'BEGIN'
739 UP[J,2]:=C[U[I]+J-1];
741 UP[J,1]:=0;
742 'FOR' K:=1 'STEP' 1 'UNTIL' LP 'DO'

743 'IF' D[K,UP[J,2]] 'THEN' UP[J,1]:=UP[J,1]+1;
744 'IF' TB=2 'THEN'
744 'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
745 'IF' UP[J,2]=XUP[K] 'AND' UP[J,1]>H 'THEN' H:=UP[J,1]
745 'END';
746 'IF' L=1 'THEN' 'GO TO' E33;
747 'IF' H>0 'THEN'
747 'FOR' J:=1 'STEP' 1 'UNTIL' L 'DO'
748 'BEGIN'
748 'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
750 'IF' UP[J,2]=XUP[K] 'THEN'
750 'BEGIN'
750 UP[J,1]:=UP[J,1]+H;
752 'GO TO' E32
752 'END';
753 'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
754 'IF' D[XUP[K],UP[J,2]] 'THEN'
754 'BEGIN'
754 UP[J,1]:=UP[J,1]+H+H;
756 'GO TO' E32
756 'END';
757 E32:'END';
758 ABVMI(FIX,UP,L,1,2,1);
759 E33:'IF' H>0 'THEN'
759 'FOR' J:=1 'STEP' 1 'UNTIL' L 'DO'
760 'FOR' K:=1 'STEP' 1 'UNTIL' LXUP 'DO'
761 'BEGIN'
761 'IF' LD=0 'AND' UP[J,2]=XUP[K] 'THEN' LD:=J;
763 'IF' UP[J,2]=XUP[K] 'THEN' LG:=J
763 'END';
764 'IF' LS>1 'THEN'
764 'BEGIN'
764 WRITE TEXT('('('5C')) THE % GROUP '));
766 PRINT(I,ND(I),0)
766 'END';
767 NEWLINE(1);
768 K:=0;
769 'FOR' J:=1 'STEP' 1 'UNTIL' L 'DO'
770 'BEGIN'
770 E34:K:=K+1;
772 NEWLINE(1);
773 'IF' H=0 'THEN' SPACE(1) 'ELSE'
773 'IF' LD=LE[K] 'AND' K=LE[LG+2] 'THEN' PRINTCH(SR) 'ELSE' SPACE(1);
774 PNK(K,ND(K));
775 'IF' H=0 'OR' (K#LG+1 'AND' K#LG+2) 'THEN'
775 'BEGIN'
775 WRITE TEXT('('('5C')) THE % SUBSYSTEM % CALCULATION

```

```

778 PRINT(M,ND(M),0);
779 N:=B[P[M]]1;
780 'IF' R[M]>P[M] OR A[N,N] THEN WRITE TEXT('(' WITH % THE % BLOCK % OF %
780 SOLVING % OF % NONLINEAR % EQUATION % SYSTEM ')');
781 'IF' H>0 AND K=LG THEN 'GO TO' E34
781 'END'
781 'ELSE'
781 'IF' K=LG+1 THEN
781 'BEGIN'
781 WRITE TEXT('(' THE % GOAL % FUNCTION % BLOCK % ( % IF % IT % IS %
782 NECESSARY % ) ')');
783 'GO TO' E34

783 'END'
783 'ELSE'
783 WRITE TEXT('(' THE % OPTIMIZATION % BLOCK ')');
784 'END'
784 'END';
785 'IF' LS>1 THEN WRITE TEXT('('('15C') THE % SEQUENCE % OF % GROUP %
785 CALCULATION % IS % OPTIONAL ('C') EACH % GROUP % CAN % BE %
785 CALCULATED % INDEPENDENT % OF % REMAINING % ONES % THAT % IS %
785 SEPARATELY ')');
786 'IF' TB=2 THEN
786 'BEGIN'
786 'IF' LS=1 THEN NEWLINE(3);
786 WRITE TEXT('('('12C') THE % FAMILY % OF % SUBSYSTEMS %AND %BLOCK %WHICH
786 % ARE % DESIGNATED % BY % * % IS % CALCULATED % ITERATIVELY %
786 DURING % OPTIMIZATION ')');
786 'END';
789 'IF' TC=0 THEN 'GO TO' E40
789 'END'
789 'END'
789 'END';
790 'IF' TA=0 THEN
790 'BEGIN'
790 'FOR' I:=1 'STEP' 1 'UNTIL' NI 'DO'
792 'BEGIN'
792 'FOR' J:=1 'STEP' 1 'UNTIL' NS 'DO'
794 'IF' T[I,1]=S[J,1] AND T[I,2]=S[J,2] THEN
794 'FOR' K:=1 'STEP' 1 'UNTIL' S[J,3] 'DO'
795 'IF' FB[J,K]=0 AND FE[J,K]=1 THEN T[I,3]:=T[I,3]+1;
796 'FOR' J:=1 'STEP' 1 'UNTIL' NS 'DO'
797 'IF' T[I,1]=S[J,2] AND T[I,2]=S[J,1] THEN
797 'FOR' K:=1 'STEP' 1 'UNTIL' S[J,3] 'DO'
798 'IF' FB[J,K]=1 AND FE[J,K]=0 THEN T[I,3]:=T[I,3]+1;
799 'END'
799 'END'
799 'ELSE'
799 'FOR' I:=1 'STEP' 1 'UNTIL' NI 'DO'
800 'FOR' J:=1 'STEP' 1 'UNTIL' NS 'DO'
801 'IF' T[I,1]=S[J,1] AND T[I,2]=S[J,2] THEN T[I,3]:=T[I,3]+S[J,3];
802 KS:='TRUE';
803 'FOR' I:=1 'STEP' 1 'UNTIL' LP 'DO'
804 'BEGIN'
804 'IF' P[I]=R[I] AND NOT A[B[P[I]],R[P[I]]] THEN 'GO TO' E42;
806 'IF' KS THEN
806 'BEGIN'
806 KS:='FALSE';
806 WRITE TEXT('('('110C') THE % OPTIMAL % TEARING % SETS % AND % THE %
806 SEQUENCE % OF % BLOCK % CALCULATION % ADEQUATE % FOR % THEM
806 ')');
808 'END';
809 G:=P[I]-1;
810 N:=R[I]-G;
811 'IF' P[I]=R[I] THEN
811 'BEGIN'
811 LK:=LK1:=LL:=LMR:=LM:=1;
813 'GO TO' E36A
813 'END';
814 LK:=LK1:=LL:=L:=M:=0;
815 'FOR' J:=P[I] 'STEP' 1 'UNTIL' R[I] 'DO'

```

```

817 'IF' A[B[J],B[K]] 'THEN' LL:=L1+1;

818 KONTURY(GT,'TRUE');
819 LL:=LL-LK1;
820 E36A:'BEGIN'
820 'INTEGER' 'ARRAY' SK[1:LL],PE[1:'IF' LK1>0 'THEN' LK1'ELSE'1],Q[1:LK],
820 PV,SV[1:N];
820 'BOOLEAN' 'ARRAY' KT[1:LK,1:LL];
821 'PROCEDURE' BK(K4,D);
823 'INTEGER' K4,D;
824 'BEGIN'
824 'INTEGER' L1,L2;
824 'INTEGER' 'PROCEDURE' TS(B1,B2);
826 'INTEGER' B1,B2;
827 'BEGIN'
827 'INTEGER' J;
827 'FOR' J:=1 'STEP' 1 'UNTIL' NI 'DO'
829 'IF' B[G+B1]=T[J,1] 'AND' B[G+B2]=T[J,2] 'THEN' 'GO TO' ETS;
830 ETS:TS:=J
830 'END' TS;
830 'IF' D>1 'THEN'
831 'BEGIN'
831 'IF' H>0 'THEN'
832 'BEGIN'
832 L1:=TS(H,K4);
834
834 'FOR' L2:=1 'STEP' 1 'UNTIL' LL 'DO'
835 'IF' L1=SK[L2] 'THEN' 'GO TO' EBK;
836 EBK:KT[L,L2]:='TRUE'
836 'END';
837 H:=K4
837 'END'
837 'ELSE'
837 PE[H]:=K4
837 'END' BK;
837 'PROCEDURE' OTS(KT,LK,Q,MV,MX,LMR,LM,NO,ET);
839 'INTEGER' LK,LMR,LM;
840 'BOOLEAN' NO;
841 'INTEGER' 'ARRAY' Q;
842 'BOOLEAN' 'ARRAY' KT,MV,MX;
843 'LABEL' ET;
844 'BEGIN'
844 'INTEGER' I,J,K,L,N,E,D,LKT,LMRT,LMT,NX,NXD,NT,LMV;
844 'BOOLEAN' PU,W,ZE;
845 'BOOLEAN' 'ARRAY' X[1:LL],EX,EXD[1:LK];
846 'INTEGER' 'PROCEDURE' LKTX(E1,E2);
848 'BOOLEAN' 'ARRAY' E1,E2;
849 'BEGIN'
849 'INTEGER' I,L;
849 L:=0;
851 'FOR' I:=1 'STEP' 1 'UNTIL' LK 'DO'
852 'IF' E1[I] 'AND' 'NOT' E2[I] 'THEN' L:=L+1;
853 LKTX:=L
853 'END' LKTX;
853 'PROCEDURE' BTX(TX,E1,E2,QT);
855 'BOOLEAN' 'ARRAY' TX,E1,E2;
856 'INTEGER' 'ARRAY' QT;
857 'BEGIN'
857 'INTEGER' I,J,L;
857 L:=0;
859 'FOR' I:=1 'STEP' 1 'UNTIL' LK 'DO'
860 'IF' E1[I] 'AND' 'NOT' E2[I] 'THEN'

860 'BEGIN'
860 L:=L+1;
862 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
863 TX[L,J]:=KT[I,J];
864 QT[L]:=Q[I]
864 'END'
864 'END' BTX;
864 ZMR(MV,LM,D,LL);

```



```

867 LMV:=0;
868 EA:I:=1;
869 LM:=0;
870 ZMR(MX,LMR,LL):
871 EAB:E:=1;
872 PU:='TRUE';
873 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
874 'IF' MV[I,J]'AND'KT[K,J] 'THEN'
874 'BEGIN'
874 PU:='FALSE';
876 'GO TO' EB
876 'END';
877 EB:L:=0;
878 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
879 X[J]:=MV[I,J];
880 'IF' PU 'THEN'
880 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
881 'IF' KT[K,J] 'THEN'
881 'BEGIN'
881 L:=L+1;
883 'IF' E=L 'THEN'
883 'BEGIN'
883 'IF' X[J] 'THEN'
884 'BEGIN'
884 'IF' E<Q[K] 'THEN' E:=E+1 'ELSE' 'GO TO' EJ;
886 'GO TO' ERA
886 'END';
887 X[J]:='TRUE';
888 'GO TO' EC
888 'END';
889 ERA:'END';
890 EC:'IF' LM=0 'THEN' 'GO TO' EH;
891 'FOR' J:=1 'STEP' 1 'UNTIL' LM 'DO'
892 'BEGIN'
892 'FOR' L:=1 'STEP' 1 'UNTIL' LL 'DO'
894 'IF' 'NOT'(X[L]'EQUIV'MX[J,L]) 'THEN' 'GO TO' ECA;
895 'GO TO' 'IF' PU 'THEN' EI 'ELSE' EJ;
896 ECA:'END';
897 'FOR' J:=1 'STEP' 1 'UNTIL' LK 'DO'
898 'BEGIN'
898 EX[J]:='FALSE';
900 'FOR' L:=1 'STEP' 1 'UNTIL' LL 'DO'
901 'IF' X[L]'AND'KT[J,L] 'THEN'
901 'BEGIN'
901 EX[J]:='TRUE';
903 'GO TO' ED
903 'END';
904 ED:'END';
905 NX:=0;
906 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
907 'IF' X[J] 'THEN' NX:=NX+T[SK[J],3];

908 W:='TRUE';
909 EDA:D:=1;
910 EE:'FOR' J:=1 'STEP' 1 'UNTIL' LK 'DO'
911 'BEGIN'
911 EXD[J]:='FALSE';
913 'FOR' L:=1 'STEP' 1 'UNTIL' LL 'DO'
914 'IF' MX[D,L]'AND'KT[J,L] 'THEN'
914 'BEGIN'
914 EXD[J]:='TRUE';
916 'GO TO' EF
916 'END';
917 EF:'END';
918 NXD:=0;
919 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
920 'IF' MX[D,J] 'THEN' NXD:=NXD+T[SK[J],3];
921 LKT:='IF' W 'THEN' LKTX(EX,EXD) 'ELSE' LKTX(EXD,EX);
922 ZE:=LKT=0;
923 'IF' ZE 'THEN' LKT:=1;
924 'BEGIN'

```

```

024 'BOOLEAN' 'ARRAY' TX[1:LKT,1:LL];
025 LMRT:=LL;
027 'IF' ZE 'THEN' 'GO TO' EG;
028 'IF' U 'THEN' BTX(TX,EX,EXD,QT) 'ELSE' BTX(TX,EXD,FX,QT);
029 EG:'BEGIN'
029 'BOOLEAN' 'ARRAY' MVT,MXT[1:LMRT,1:LL];
029 NT:=0;
031 'IF' ZE 'THEN' 'GO TO' EGA;
032 OTS(TX,LKT,QT,MVT,MXT,LMRT,LMT,'FALSE',EG);
033 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
034 'IF' MXT[1,J] 'THEN' NT:=NT+T[SK[J],3];
035 EGA:'IF' U 'THEN'
035 'BEGIN'
035 'IF' 'IF' NO 'THEN' NT+NXD<NX 'ELSE' NT+NXD'LE'NX 'THEN'
036 'GO TO' 'IF' PU 'THEN' EI 'ELSE' EJ
036 'END'
036 'ELSE'
036 'IF' 'IF' NO 'THEN' NT+NX<NXD 'ELSE' NT+NX'LE'NXD 'THEN'
036 'BEGIN'
036 'FOR' J:=D+1 'STEP' 1 'UNTIL' LM 'DO'
038 'FOR' L:=1 'STEP' 1 'UNTIL' LL 'DO'
039 MX[J-1,L]:=MX[J,L];
040 LM:=LM-1;
041 'IF' D<LM 'THEN' 'GO TO' EE
041 'END'
041 'END'
041 'END';
042 D:=D+1;
043 'IF' D'LE'LM'AND'W'OR'D<LM'AND'NOT'W 'THEN' 'GO TO' EE;
044 'IF' NOT'W 'THEN' 'GO TO' EHA;
045 EH:LM:=LM+1;
046 'IF' LM>LMR 'THEN'
046 'BEGIN'
046 LMR:=LMR+10;
048 'GO TO' ET
048 'END';
049 'FOR' J:=1 'STEP' 1 'UNTIL' LL 'DO'
050 MX[LM,J]:=X[J];
051 W:='FALSE';
052 'IF' LM>1 'THEN' 'GO TO' EDA;

053 EHA:'IF' NOT'PU 'THEN' 'GO TO' EJ;
054 EI:'IF' E<O[K] 'THEN'
054 'BEGIN'
054 E:=E+1;
056 'GO TO' EB
056 'END';
057 EJ:'IF' I<LMV 'THEN'
057 'BEGIN'
057 I:=I+1;
059 'GO TO' EAB
059 'END';
060 'IF' K<LK 'THEN'
060 'BEGIN'
060 'FOR' J:=1 'STEP' 1 'UNTIL' LM 'DO'
062 'FOR' L:=1 'STEP' 1 'UNTIL' LL 'DO'
063 MV[J,L]:=MX[J,L];
064 LMV:=LM;
065 K:=K+1;
066 'GO TO' EA
066 'END'
066 'END' OTS;
066 'IF' P[I]=R[I] 'THEN' 'GO TO' #39;
068 L:=0;
069 'FOR' J:=P[I] 'STEP' 1 'UNTIL' R[I] 'DO'
070 'FOR' K:=P[I] 'STEP' 1 'UNTIL' R[I] 'DO'
071 'IF' J#K'AND'A[B[J],B[K]] 'THEN'
071 'BEGIN'
071 L:=L+1;
073 'FOR' M:=1 'STEP' 1 'UNTIL' NI 'DO'
074 'IF' B[J]-T[M,1]AND'B[K]-T[M,2] 'THEN' 'GO TO' #42;

```



```

975 'END';
976 ABV(SK,LL);
977 ZMB(KT,LK,LL);
978 L:=M:=0;
979 KONTURY(BK,'FALSE');
980 J:=1;
981 E37: 'FOR' K:=1 'STEP' 1 'UNTIL' LL 'DO'
982 'BEGIN'
982 'FOR' L:=1 'STEP' 1 'UNTIL' LK 'DO'
984 'IF' KT[L,J] 'AND' 'NOT' KT[L,K] 'THEN' 'GO TO' E38;
985 'IF' T[SK[J],3] 'LE' T[SK[K],3] 'THEN' 'GO TO' E38;
986 'FOR' M:=J+1 'STEP' 1 'UNTIL' LL 'DO'
987 'BEGIN'
987 'FOR' L:=1 'STEP' 1 'UNTIL' LK 'DO'
989 KT[L,M-1]:=KT[L,M];
990 SK[M-1]:=SK[M]
990 'END';
991 LL:=LL-1;
992 'GO TO' 'IF' J'LE'LL 'THEN' E37 'ELSE' E38A;
993 E38: 'END';
994 J:=J+1;
995 'IF' J'LE'LL 'THEN' 'GO TO' E37;
996 E38A: 'FOR' J:=1 'STEP' 1 'UNTIL' LK 'DO'
997 'BEGIN'
997 Q[J]:=0;
999 'FOR' K:=1 'STEP' 1 'UNTIL' LL 'DO'
1000 'IF' KT[J,K] 'THEN' Q[J]:=Q[J]+1
1000 'END';
1001 LLP:=LL;

1002 J:=1;
1003 E38B: 'IF' Q[J]=1 'THEN'
1003 'BEGIN'
1003 'FOR' K:=1 'STEP' 1 'UNTIL' LL 'DO'
1005 'IF' KT[J,K] 'THEN' 'GO TO' E38C;
1006 E38C: L:=1;
1007 E38D: 'IF' KT[L,K] 'THEN'
1007 'BEGIN'
1007 'FOR' H:=L+1 'STEP' 1 'UNTIL' LK 'DO'
1009 'BEGIN'
1009 'FOR' M:=1 'STEP' 1 'UNTIL' LL 'DO'
1011 KT[H-1,M]:=KT[H,M];
1012 Q[H-1]:=Q[H]
1012 'END';
1013 LK:=LK-1;
1014 'GO TO' 'IF' L'LE'LK 'THEN' E38D 'ELSE' E38E
1014 'END';
1015 L:=L+1;
1016 'IF' L'LE'LK 'THEN' 'GO TO' E38D;
1017 E38E: L:=SK[K];
1018 'FOR' M:=K+1 'STEP' 1 'UNTIL' LL 'DO'
1019 'BEGIN'
1019 'FOR' H:=1 'STEP' 1 'UNTIL' LK 'DO'
1021 KT[H,M-1]:=KT[H,M];
1022 SK[M-1]:=SK[M]
1022 'END';
1023 SK[LL]:=L;
1024 LL:=LL-1;
1025 'GO TO' 'IF' J'LE'LK 'THEN' E38B 'ELSE' E38F
1025 'END';
1026 J:=J+1;
1027 'IF' J'LE'LK 'THEN' 'GO TO' E38B;
1028 E38F: 'IF' LK=0 'OR' LL=0 'THEN' 'GO TO' E38I;
1029 J:=1;
1030 E38G: 'FOR' H:=1 'STEP' 1 'UNTIL' J-1,J+1 'STEP' 1 'UNTIL' LK 'DO'
1031 'BEGIN'
1031 'FOR' M:=1 'STEP' 1 'UNTIL' LL 'DO'
1033 'IF' 'NOT' KT[J,M] 'AND' KT[H,M] 'THEN' 'GO TO' E38H;
1034 'FOR' K:=J+1 'STEP' 1 'UNTIL' LK 'DO'
1035 'BEGIN'
1035 'FOR' M:=1 'STEP' 1 'UNTIL' LL 'DO'

```

```

1038 Q[K-1]:=Q[K]
1038 'END';
1039 LK:=LK-1;
1040 'GO TO' 'IF' J'LE' LK 'THEN' E38G 'ELSE' E38I;
1041 E38H:'END';
1042 J:=J+1;
1043 'IF' J'LE' LK 'THEN' 'GO TO' E38G;
1044 E38I:LMR:='IF' LL>0 'THEN' LL 'ELSE' 1;
1045 'IF' LK=0'AND'LL>0 'THEN'
1045 'BEGIN'
1045 'FOR' J:=LL+1 'STEP' 1 'UNTIL' LLP 'DO'
1047 SK[J-LL]:=SK[J];
1048 LLP:=LLP-LL
1048 'END';
1049 E39:'BEGIN'
1049 'BOOLEAN' 'ARRAY' MV,MX[1:LMR,1;'IF' LL>0 'THEN' LL 'ELSE' 1];
1049 'IF' P[I]=R[I] 'THEN'
1050 'BEGIN'

```

```

1050 LL:=0;
1052 PE[1]:=1;
1053 PV[1]:=1;
1054 'GO TO' E40
1054 'END';
1055 'IF' LK=0'OR'LL=0 'THEN'
1055 'BEGIN'
1055 LM:=1;
1057 'GO TO' E39A
1057 'END';
1058 OTS(KT,LK,0,MV,MX,LMR,LM,'TRUE',E39);
1059 E39A:'FOR' J:=1 'STEP' 1 'UNTIL' LK1 'DO'
1060 A[B[G+PE[J]],B[G+PE[J]]]:='FALSE';
1061 E40:'FOR' L:=1 'STEP' 1 'UNTIL' LM 'DO'
1062 'BEGIN'
1062 'IF' P[I]=R[I] 'THEN' 'GO TO' #41;
1064 'FOR' J:=1 'STEP' 1 'UNTIL' LLP 'DO'
1065 'IF' 'IF' J'LE' LL 'THEN' MX[L,J] 'ELSE' 'TRUE' 'THEN'
1065 A[T[SK[J],1],T[SK[J],2]]:='FALSE';
1066 N2:=-N-2;
1067 'FOR' H:=1 'STEP' 1 'UNTIL' N 'DO'
1068 SV[H]:=0;
1069 'FOR' K:=1,K+1 'WHILE' K'LE' N 'DO'
1070 'BEGIN'
1070 'FOR' H:=1 'STEP' 1 'UNTIL' N 'DO'
1072 'IF' SV[H]=0 'THEN'
1072 'BEGIN'
1072 'FOR' J:=1 'STEP' 1 'UNTIL' N 'DO'
1074 'IF' SV[J]=0 'THEN'
1074 'BEGIN'
1074 'IF' A[B[G+J],B[G+H]] 'THEN' 'GO TO' NEXTI
1075 'END';
1076 PV[K]:=H;
1077 SV[H]:=N2;
1078 'GO TO' NEXTK;
1079 NEXTI:'END';
1080 NEXTK:'END';
1081 'FOR' J:=1 'STEP' 1 'UNTIL' LLP 'DO'
1082 'IF' 'IF' J'LE' LL 'THEN' MX[L,J] 'ELSE' 'TRUE' 'THEN'
1082 A[T[SK[J],1],T[SK[J],2]]:='TRUE';
1083 E41:WRITE TEXT('('('5C')' THE % SUBSYSTEM ')');
1084 PRINT(I,ND(I),0);
1085 NEWLINE(3);
1086 'IF' TA=1 'THEN'
1086 'BEGIN'
1086 WRITE TEXT('(' THE % TORN % STREAMS % : ')');
1088 H:=SU:=18;
1089 'FOR' J:=1 'STEP' 1 'UNTIL' LLP 'DO'
1090 'IF' 'IF' J'LE' LL 'THEN' MX[L,J] 'ELSE' 'TRUE' 'THEN'
1090 'BEGIN'
1090 K:=T[SK[J],1];
1092 M:=T[SK[J],2];

```

```

1094 'IF' K=S[E,1]'AND'M=S[E,2] 'THEN' DRF(SN[E])
1094 'END';
1095 'FOR' J:=1 'STEP' 1 'UNTIL' LK1 'DO'
1096 'BEGIN'
1096 K:=B[G+PE[J]];
1098 'FOR' E:=1 'STEP' 1 'UNTIL' NS 'DO'
1099 'IF' K=S[E,1]'AND'K=S[E,2] 'THEN' DRF(SN[E])

1099 'END'
1099 'END'
1099 'ELSE'
1099 'BEGIN'
1099 WRITE TEXT('(' THE % TORN % FEATURES % : ')');
1101 'FOR' J:=1 'STEP' 1 'UNTIL' LLP 'DO'
1102 'IF' 'IF' J'LE'LL 'THEN' MX[L,J] 'ELSE' 'TRUE' 'THEN'
1102 'BEGIN'
1102 K:=T[SK[J],1];
1104 M:=T[SK[J],2];
1105 'FOR' E:=1 'STEP' 1 'UNTIL' NS 'DO'
1106 'BEGIN'
1106 KZ:=K=S[E,1]'AND'M=S[E,2];
1108 'IF' KZ'OR'K=S[E,2]'AND'M=S[E,1] 'THEN'
1108 'BEGIN'
1108 DRS(E);
1110 'FOR' F:=1 'STEP' 1 'UNTIL' MNFS 'DO'
1111 'IF' 'IF' KZ 'THEN' FB[E,F]=0'AND'FE[E,F]=1
1111 'ELSE' FB[E,F]=1'AND'FE[E,F]=0 'THEN' DRF(F)
1111 'END'
1111 'END'
1111 'END';
1112 'FOR' J:=1 'STEP' 1 'UNTIL' LK1 'DO'
1113 'BEGIN'
1113 K:=B[G+PE[J]];
1115 'FOR' E:=1 'STEP' 1 'UNTIL' NS 'DO'
1116 'IF' K=S[E,1]'AND'K=S[E,2] 'THEN'
1116 'BEGIN'
1116 DRS(E);
1118 'FOR' F:=1 'STEP' 1 'UNTIL' MNFS 'DO'
1119 'IF' FB[E,F]=0'AND'FE[E,F]=1'OR'FB[E,F]=1'AND'FE[E,F]=0 'THEN' DRF(F)
1119 'END'
1119 'END'
1119 'END';
1120 WRITE TEXT('('('('3C')' THE % SEQUENCE % OF % BLOCK % CALCULATION
1120 '('C')')');
1121 H:=N+1;
1122 'FOR' J:=1 'STEP' 1 'UNTIL' H 'DO'
1123 'BEGIN'
1123 NEWLINE(1);
1125 SPACE(1);
1126 PNK(J,ND(J));
1127 WRITE TEXT('(' THE % BLOCK ')');
1128 'IF' J=H 'THEN' WRITE TEXT('(' % OF % SOLVING % OF % NONLINEAR %
1128 EQUATION % SYSTEM ')')
1128 'ELSE'
1128 'BEGIN'
1128 K:=BN[B[G+PV[J]]];
1130 PRINT(K,ND(K),0)
1130 'END'
1130 'END'
1130 'END'
1130 'END';
1131 E42:'END'
1131 'END'
1131 'END';
1132 'GO TO' E46;
1133 E43:SELECT OUTPUT(2);
1134 WRITE TEXT('('('('10C')' THE % DIVISION % OF % THE % SYSTEM % STREAM %
1134 FEATURES % WITH % REGARD % ON % THE % PART % THEY % MAY %
1134 PLAY % DURING % OPTIMIZATION '('5C')' THE % DECISION %

```

```

1135 'IF' TA=1 'THEN' WRITE TEXT('(' THE % FEATURES % OF % INLET % STREAMS %
1135 OF % THE % SYSTEM '))');
1135 'ELSE'
1135 'BEGIN'
1135 WRITE TEXT('(' THE % FOLLOWING % FEATURES % : '('C')')');
1137 'FOR' I:=1 'STEP' 1 'UNTIL' NS 'DO'
1138 'BEGIN'
1138 'IF' S[I,1]=0 'THEN'
1139 'FOR' J:=1 'STEP' 1 'UNTIL' S[I,3] 'DO'
1140 'IF' FE[I,J]=1 'THEN'
1140 'BEGIN'
1140 DRS(I);
1142 'FOR' K:=J 'STEP' 1 'UNTIL' S[I,3] 'DO'
1143 'IF' FE[I,K]=1 'THEN' DRF(K);
1144 'GO TO' E44
1144 'END';
1145 'IF' S[I,2]=0 'THEN'
1145 'FOR' J:=1 'STEP' 1 'UNTIL' S[I,3] 'DO'
1146 'IF' FB[I,J]=1 'THEN'
1146 'BEGIN'
1146 DRS(I);
1148 'FOR' K:=J 'STEP' 1 'UNTIL' S[I,3] 'DO'
1149 'IF' FB[I,K]=1 'THEN' DRF(K);
1150 'GO TO' E44
1150 'END';
1151 'FOR' J:=1 'STEP' 1 'UNTIL' S[I,3] 'DO'
1152 'IF' FB[I,J]=FE[I,J] 'THEN'
1152 'BEGIN'
1152 DRS(I);
1154 'FOR' K:=J 'STEP' 1 'UNTIL' S[I,3] 'DO'
1155 'IF' FB[I,K]=FE[I,K] 'THEN' DRF(K);
1156 'GO TO' E44
1156 'END';
1157 E44:'END';
1157 'END';
1158 WRITE TEXT('('('SC') THE % FEATURES % USED % FOR % CALCULATION % OF %
1158 GOAL % FUNCTION % VALUE % MAY % BE % THE % REMAINING % STREAM
1158 % FEATURES :('SC') IN % THE % SIMILAR % WAY % THE % OBJECT %
1158 FEATURES % OF % THE % SYSTEM % CAN % BE % EASILY % DIVIDED
1158 '))');
1159 'GO TO' E46
1159 'END';
1160 E45:SELECT OUTPUT(2);
1161 WRITE TEXT('('('100') INCORRECT % NUMERATION % IN % THE % ARRAY % '))');
1162 'IF' BER=1 'THEN' WRITE TEXT('(' DS '))' 'ELSE'
1162 'IF' BER=2 'THEN' WRITE TEXT('(' DO '))' 'ELSE'
1162 'IF' BER=3 'THEN' WRITE TEXT('(' VS '))' 'ELSE'
1162 'IF' BER=4 'THEN' WRITE TEXT('(' VO '))' 'ELSE'
1162 WRITE TEXT('(' *BNPM '))');
1163 E46:'END';
1163 'END'

```

```

SEGMENT ANALYZER      LENGTH 14035
NO OF BUCKETS USED    275
COMPILED #ANALYZER    EC

```

CONSOLIDATED BY XPCK 42H DATE 14/07/79 TIME 08/22/35

AAAA00070642(1) RENAMED ICLA-DEFAULT(1)

```

PROGRAM ANAL
EXTENDED DATA (22AM)
EXTENDED PROGRAM (EBM)
CORE            22336

```

```

CLV      ALGOL60
SEG      WRITETEXT
SEG      PRINTCH

```


ZALACZNIK 4

18/47/33

06/09778

COMPILED BY XALT MK. 5A

STATEMENT

```

0      PROGRAM (ARCHIVIST)
0      PRIORITY 180
0      BEGIN
1      INTEGER (OFF, I, J, K, L, M, N, NL, SP, SR,
1      BOOLEAN T;
2      INTEGER ARRAY V(1:5);
3      PROCEDURE CREATE(N, S);
5      VALUE IN;
6      INTEGER IN;
7      STRING S;
8      EXTERNAL;
8      PROCEDURE USE(N, S);
10     VALUE IN;
11     INTEGER IN;
12     STRING S;
13     EXTERNAL;
13     PROCEDURE DATA SKIP(N);
15     VALUE IN;
16     INTEGER IN;
17     EXTERNAL;
17     PROCEDURE WRITE BINARY(N, A, S);
19     VALUE IN;
20     INTEGER IN;
21     ARRAY A;
22     STRING S;
23     EXTERNAL;
23     PROCEDURE REWIND(N);
25     VALUE IN;
26     INTEGER IN;
27     EXTERNAL;
27     PROCEDURE FREEMT(N);
29     VALUE IN;
30     INTEGER IN;
31     EXTERNAL;
31     INTEGER, PROCEDURE, TRC;
32     BEGIN
32     INTEGER IC;
32     DO:CH=NEXTCH;
34     IF IC=SP OR IC=NL THEN
34     BEGIN
34     SKIPCH;
36     GO TO EC
36     END;
37     IC=IC
37     END TRC;
37     PROCEDURE STG(W, A, ND);
39     VALUE IN;
40     INTEGER IN;
41     ARRAY W, A;
42     BEGIN
42     FOR I=1 STEP 1 UNTIL ND DO W[I]=V[I];
45     WRITE BINARY(N, W, IC(13));
46     WRITE TEXT('IC(13C) THE % DIMENSION % RECORD ');
47     PRINT D(2, 0);
48     WRITE TEXT('IS % STORED % ON % THE % MAGNETIC % TAPE ');

```

```

50 WRITE TEXT('('('C') THE % DATA % RECORD ')');
51 PRINT(D/2/0);
52 WRITE TEXT('(' IS % STORED % ON % THE % MAGNETIC % TAPE ')');
53 GO TO EX
53 END STG;
53 N:=20;
55 SF:=CODE('(' % ')');
56 NL:=CODE('(' EL ');');
57 SR:=CODE('(' * ')');
58 SELECT INPUT(3);
59 TR:=READ BOCLEAN;
60 IF TR THEN
60 BEGIN
60 USE (N/('3')));
62 DATA SKIP(N)
62 END
62 ELSE CREATE(N,('3'));
63 F:=READ;
64 FOR D:=1 STEP 1 UNTIL F 'DO'
65 BEGIN
65 L:=1;
67 FOR J:=1 STEP 1 UNTIL 5 'DO'
68 BEGIN
68 IF L=1 THEN
69 BEGIN
69 IF TRC=SR THEN L:=0
70 END;
71 V[J]:=IF L=1 THEN READ ELSE 0
71 END J;
72 BD:=IF SR=NEXTCH THEN
72 BEGIN
72 SKIPCH;
74 GO TO ED
74 END;
75 SKIPCH;
76 SELECT OUTPUT(2);
77 NEWLINE(5);
78 IF V[5]>0 THEN
78 BEGIN
78 ARRAY W[1:5], A[1:V[1],1:V[2],1:V[3],1:V[4],1:V[5]];
78 FOR I:=1 STEP 1 UNTIL V[1] 'DO'
80 FOR J:=1 STEP 1 UNTIL V[2] 'DO'
81 FOR K:=1 STEP 1 UNTIL V[3] 'DO'
82 FOR L:=1 STEP 1 UNTIL V[4] 'DO'
83 FOR M:=1 STEP 1 UNTIL V[5] 'DO'
84 A[I,J,K,L,M]:=READ;
85 STG(W7A75)
85 END;
86 IF V[4]>0 THEN
86 BEGIN
86 ARRAY W[1:4], A[1:V[1],1:V[2],1:V[3],1:V[4]];
86 FOR I:=1 STEP 1 UNTIL V[1] 'DO'
88 FOR J:=1 STEP 1 UNTIL V[2] 'DO'
89 FOR K:=1 STEP 1 UNTIL V[3] 'DO'
90 FOR L:=1 STEP 1 UNTIL V[4] 'DO'
91 A[I,J,K,L]:=READ;
92 STG(W7A74)
92 END;
93 IF V[3]>0 THEN
93 BEGIN
93 ARRAY W[1:3], A[1:V[1],1:V[2],1:V[3]];
93 FOR I:=1 STEP 1 UNTIL V[1] 'DO'
95 FOR J:=1 STEP 1 UNTIL V[2] 'DO'

```



```

97      A(I,J7K) := READ;
98      STG(W7A,3)
98      FEND;
99      IF V[2] > 0 THEN
99      BEGIN
99      ARRAY I[1:2], A[1:V[1]71:V[2]];
99      FOR I:=1 STEP 1 UNTIL V[1] DO,
101     FOR J:=1 STEP 1 UNTIL V[2] DO,
102     A[I,J] := READ;
103     STG(W7A,2)
103     FEND;
104     IF V[1] > 0 THEN
104     BEGIN
104     ARRAY I[1:1], A[1:V[1]];
104     FOR I:=1 STEP 1 UNTIL V[1] DO
106     A[I] := READ;
107     STG(W7A,1)
107     FEND;
108     EXTEND D;
109     REWIND(N);
110     FREEMT(N)
110     FEND;

```

```

:SEGMENT ARCHIVIST      LENGTH 1179
:INFO OF BUCKETS USED 27
:COMPILED #ARCHIVIST    EC

```

:CONSOLIDATED BY XPCX 12J DATE 06/09/78 TIME 18/48/21

:ICLALQCL (4) RENAMED ICLAL-DEFAULT(4)

```

:PROGRAM ARCH
:EXTENDED DATA (22AM)
:COMPACT PROGRAM (DMP)
:CORE 9344

```

```

:SEG :WRITERINARY
:SEG :CREATE
:SEG :DATASKIP
:SEG :REWIND
:SEG :USE
:SEG :FREEMT
:CUV :ALQCL60
:SEG :WRITETEXT
:SEG :SKIRCH
:SEG :PRINT
:SEG :INBLINE
:SEG :INEXTCH
:SEG :READPOCLEAN
:SEG :SELECTINPUT
:SEG :CODE
:SEG :READ
:SEG :READCH

```