
KONRAD FIAŁKOWSKI

**AUTOKODY
I PROGRAMOWANIE
MASZYN
CYFROWYCH**



mach tych przewidziana jest możliwość definiowania makroinstrukcji. Również niektóre realizacje asemblerów dopuszczają użycie podprogramów.

Do bardziej znanych asemblerów należą: PACT opracowany dla maszyny IBM-701, SAP (Share Assembly Program) opracowany dla maszyny IBM-704. W Polsce opracowany został SAS (System Adresów Symbolicznych) dla maszyny ZAM-2 [11].

4.3. SYSTEM ADRESÓW SYMBOLICZNYCH — SAS

System Adresów Symbolicznych (SAS) jest językiem zewnętrznym maszyny ZAM-2. Program napisany w tym języku jest rozumiany przez maszynę po uprzednim umieszczeniu w jej pamięci programu zwanego *translatorem SAS*.

Program napisany w języku SAS składa się z części zwanych rozdziałami. Są to odcinki programu złożone z rozkazów, liczb oraz dyrektyw.

Rozkazy w języku SAS składają się z dwuliterowego skrótu części operacyjnej (tabl. 4.1), oraz z części adresowej. Część adresowa może zawierać adres bezwzględny, adres względny lub adres symboliczny, a ponadto dodatkowe znaki charakteryzujące rozkaz. Ich postać i znaczenie podaje literatura [11].

Adresy bezwzględne podają adresy komórek pamięci operacyjnej, na przykład:

PA 913

DO 2

Adresy względne wskazują położenie komórki, do której zawartości odnosi się rozkaz względem samego rozkazu, na przykład:

UA — 2

jest rozkazem powodującym umieszczenie w akumulatorze zawartości komórki o adresie o dwa mniejszym niż adres komórki, w której umieszczony jest sam rozkaz UA — 2.

Tak więc rozkaz DO + 0 spowoduje dodanie do akumulatora słowa przedstawiającego rozkaz DO + 0, traktując to słowo jako liczbę. W odróż-

nieniu od adresów bezwzględnych adresy względne zapisywane są jako liczby opatrzone znakiem.

Adresy symboliczne określają argument rozkazu przez podanie symbolu bloku (zawierającego liczby lub rozkazy) oraz indeksu wskazującego względne położenie argumentu rozkazu wewnątrz bloku. W szczególnym przypadku blok może być jednoelementowy (np. pojedyncza liczba) i wtedy symbol stanowi adres symboliczny tego pojedynczego elementu (liczby), np.

DO X

Adres X wskazuje zerowy element bloku X (jedyny element, jeśli blok jest jednoelementowy). Natomiast

DO X(11)

wskazuje jedenasty element bloku X.

Liczby zapisywane są w języku SAS w zapisie dziesiętnym, przy czym kropka oddziela część całkowitą od ułamkowej, np.

+1721.123

Oczywiście przy zapisie liczb występują problemy związane ze skalą obliczeń, lecz rozważania dotyczące skali przekraczają zakres tej pracy.

W języku SAS istnieje również pojęcie numeru symbolicznego. Numery symboliczne są to symbole poszczególnych partii programu. Pisze się je z lewej strony rozkazu lub liczby stanowiącej początek odpowiedniego bloku, na przykład:

.....
1K) DO 1002
.....

W dalszym ciągu programu można się odwołać do numeru symbolicznego, na przykład

.....
SK 1K)
.....

lub do rozkazu należącego do bloku (np. trzeciego licząc od jedynki w bloku)

.....
SK 1K + 2)
.....

W pierwszym przypadku zrealizowany zostanie skok do pierwszego rozkazu bloku oznaczonego symbolem 1K, w drugim zaś do третьiego rozkazu tego bloku.

Interesujący jest sposób w jaki można w języku SAS dołączać do programu podprogramy umieszczone uprzednio w pamięci pomocniczej maszyny. W tym celu w programie używa się dyrektywy FUN. Niech np pożądane będzie dołączenie podprogramu sinusa wprowadzonego uprzednio do pamięci pomocniczej i oznaczonego numerem identyfikacyjnym np. 15. W tym celu w miejscu programu, gdzie powinien zostać dołączony podprogram, należy napisać

.....

x1) FUN 15

.....

Numer symboliczny dopisany z lewej strony dyrektywy FUN pozwala odwoływać się do podprogramu z dowolnego miejsca rozdziału programu

Koniec translacji programu napisanego w języku SAS sygnalizowany jest translatorowi dyrektywą STA n/N , gdzie N jest numerem rozdziału, n natomiast adresem rozkazu w tym rozdziale. Dyrektywa ta powoduje zakończenie translacji, wprowadzenie z pamięci pomocniczej do pamięci operacyjnej rozdziału N i rozpoczęcie wykonywania rozkazu — start od adresu n .

Podany wyżej zarys języka SAS ma na celu naszkicowanie najistotniejszych cech charakterystycznych tego języka. Oczywiście, nie może być on traktowany jako podstawa nauki programowania w tym języku; wszystkim zainteresowanym poleca się literaturę stanowiącą kompendium programowania w języku SAS [11].,

Język SAS jest również językiem pośredniczącym przy tłumaczeniu programów napisanych w języku SAKO (System Automatycznego Kodowania).

Język SAS powstał jako rezultat seminarium prowadzonego w Zakładzie Programowania Instytutu Maszyn Matematycznych PAN. Pierwszy translator SAS dla maszyny XYZ zrealizowany został w 1960 r. a jego rozszerzoną i ulepszoną wersję dla maszyny ZAM-2 uruchomiono pod koniec 1961 roku.

4.4. KOMPILACJA I INTERPRETACJA

Jeżeli pracę maszyny, poczynając od przekazania przez programistę programu urządzeniom wejściowym, a kończąc na uzyskaniu przez maszynę wyników, traktować jako jeden proces, to może on być realizowany dwoma różnymi sposobami.

Pierwszy omawiany dotychczas sposób polega na wprowadzeniu do maszyny całego programu napisanego w języku zewnętrznym, przekształceniu programu przez program tłumaczący w program wynikowy, a następnie wykonaniu programu wynikowego. Przy takim wykonaniu programu wyróżnić można dwie fazy: fazę przygotowania programu wynikowego i fazę wykonania programu wynikowego. Automatyczne kodowanie, przy którym program realizowany jest w ten sposób, nazywane jest *kompilacją*.

Możliwy jest również inny sposób wykonywania programu. Program napisany w języku zewnętrznym może być analizowany w maszynie fragmentami, przy czym dla każdego pojedynczego fragmentu następuje tłumaczenie, a następnie uzyskany w języku wynikowym fragment programu jest natychmiast wykonywany. Po nim tłumaczony jest i wykonywany następny fragment itd. dopóty, dopóki nie zostanie zrealizowany cały program.

Analizowanymi fragmentami programu są elementy języka zewnętrznego, którym odpowiadają skończone fragmenty programu w języku wynikowym (rozkazy, grupy rozkazów, podprogramy). Automatyczne kodowanie, przy którym program realizowany jest w ten sposób, nazywa się *interpretacją*. Cechą charakterystyczną interpretacji jest niezachowywanie w pamięci fragmentów programu wynikowego, które zostały wykonane. Tak więc przy interpretacji, w odróżnieniu od komplikacji, program wynikowy nie znajduje się w całości w pamięci maszyny.

Mögliwe są formy automatycznego kodowania pośrednie między komplikacją a interpretacją.

Każdy z dwu opisanych wyżej sposobów realizacji automatycznego kodowania ma zarówno swoje zalety jak i wady.

Najpoważniejszą wadą interpretowania jest konieczność każdorazowego dokonywania tłumaczenia, nawet wtedy, kiedy dany fragment informacji realizowany jest wielokrotnie (np. w pętli) w danym programie. Przewodzi to do przedłużenia czasu wykonywania programu przez maszynę. Ponadto program interpretujący musi być umieszczony w pamięci maszyny przez cały czas wykonywania programu napisanego w języku zewnętrznym. Przy komplikacji konieczność taka nie istnieje i gdy program wynikowy zostanie już ułożony, obszar pamięci zajęty podczas komplikacji przez program tłumiaczący może być dowolnie wykorzystany.

Z drugiej strony interpretacja jest metodą bardziej elastyczną i nie wiążą się z nią problemy podziału pamięci, które muszą być uwzględniane przez program komplikujący.

Praktyka wykazała, że dla problemów obliczeniowych stosowanie metody komplikacji jest dogodniejsze, dlatego też większość współcześnie istniejących programów automatycznego kodowania należy do grupy programów komplikujących (compiling routines). Pojęcie programu komplikującego używane jest zazwyczaj w znaczeniu węższym niżby to wynikało z wyżej podanego omówienia. Jest ono niekiedy używane przeciwnie do pojęcia assemblera. Zarówno kryteria podziału jak i terminologia nie są w tym przypadku zbyt ścisłe, jednak na ogół przyjmuje się, że program komplikujący w drugim podanym wyżej znaczeniu dopuszcza w języku zewnętrznym wyłącznie rozkazy nie mające bezpośrednich odpowiedników w rozkazach maszyny. Tak więc programowanie w językach programów komplikujących nie jest programowaniem w języku maszyny. Ponadto, program komplikujący może analizować rozkazy języka zewnętrznego kompleksowo, wykorzystując przy tworzeniu danego odcinka programu wynikowego informacje zawarte w różnych rozkazach języka zewnętrznego.

4.5. JĘZYKI AUTOMATYCZNEGO KODOWANIA

Zagadnienie języka używanego przez programistę przy sformułowaniu zadania jest, jak wspomniano, zagadnieniem pierwszoplanowym. Język ten powinien być jak najbardziej dogodny dla człowieka. Z tego też wywodzi się idea wykorzystania symboliki algebry w językach wewnętrznych.

Pierwsze eksperymenty w tym kierunku datują się z lat 1951—52 i były prowadzone niezależnie, w Związku Radzieckim i w Stanach Zjednoczonych. W efekcie tych prac powstały: język dla maszyny Striela-2 (1955), FORTRAN (FORmula TRANslator, 1956), dla maszyny IBM-704, język dla maszyny BESM (1956), UNICODE dla maszyny Univac Scientific 1103A (1957), IT (Internal Translator) dla maszyn: IBM-650, Datatron-205, Univac Scientific 1103A, IBM-701. Języki te są językami stworzonymi najwcześniej. Później na przełomie lat pięćdziesiątych — sześćdziesiątych powstały następne języki. W tej pracy zajmiemy się językiem SAKO (System Automatycznego Kodowania, dla maszyny ZAM-2), językami ALGOL (ALGOrithmic Language), FORTRAN i LISP.

Zależnie od budowy translatora tłumaczenie programu napisanego w języku zewnętrznym w program wynikowy może przebiegać w jednym lub wielu etapach. Przy dwuetapowym tłumaczeniu programu językiem pośrednim, w którym program zapisywany jest po pierwszym etapie tłumaczenia, może być język zewnętrzny takich programów pierwotnych jak SAP, SOAP (Symbolic Optimal Assambly Program), czy SAS (ściślej mówiąc WSJ — Wewnętrzny Język Symboliczny — zbliżony do języka SAS).

Niewątpliwą zaletą dwuetapowego tłumaczenia jest umożliwienie programistie wglądu i dokonania ewentualnych zmian w programie napisanym w języku programu pierwotnego uzyskanym po pierwszym etapie tłumaczenia.

Języki automatycznego kodowania są w mniejszym lub w większym stopniu zbliżone do tradycyjnego języka matematyki. Dzięki temu przyzwojenie umiejętności programowania w tych językach jest sprawą stosunkowo łatwą.

Na początku tego rozdziału wspomniano, że proces programowania składa się z dwóch faz, z których drugą jest kodowanie. Automatyczne kodowanie ma na celu zredukowanie do minimum prac wykonywanych przez programistę w drugiej fazie. Cel ten zostaje osiągnięty, jeżeli dla danej maszyny problem może być przedstawiony w języku łatwym z punktu widzenia programisty, a jednocześnie język ten dostępny będzie dla maszyny, jest ona bowiem zaopatrzona w odpowiedni translator. Przy takim programowaniu faza kodowania wykonywana przez programistę zostaje poważnie zmniejszona, prace bowiem wykonywane poprzednio przez niego realizuje maszyna posługując się tłumaczem. Języki dostępne dla maszyn i wyko-

rzystywane dla celów automatycznego kodowania nazywane są językami automatycznego kodowania. Translatory tłumaczące programy zapisane w tych językach w programy wynikowe zawierają zazwyczaj wiele tysięcy rozkazów. Tak duża złożoność języków staje się zrozumiałą, jeżeli uwzględnimy sobie, że translatorzy wykonują pracę wprawnych znających programowanie w języku maszyny programistów.

Stworzenie języków automatycznego kodowania — języków algorytmicznych — ma jednak szersze znaczenie wykraczające poza ramy maszyn cyfrowych. Języki te mogą stać się ogólnie przyjętym i ogólnie zrozumiałym środkiem zapisu procesów algorytmicznych. Zadanie to wykonuje już częściowo język ALGOL-60 opisany w dalszej części tej pracy.

5. SYSTEM AUTOMATYCZNEGO KODOWANIA — SAKO

System Automatycznego Kodowania — język SAKO przeznaczony jest do wyrażania i zapisywania problemów obliczeniowych i logicznych. Język SAKO tworzony był w taki sposób, by można w nim było stosunkowo prosto wyrazić dowolny problem obliczeniowy lub logiczny, a ponadto by program pisany w tym języku pozwalał wykorzystać wszelkie potencjalne możliwości maszyny cyfrowej. Problem jest tym bardziej skomplikowany, że język powinien być możliwie jak najprostszy i niezależny od charakterystycznych cech konkretnej maszyny cyfrowej. W pewnym sensie założenia te są przeciwwstawne. Ostatecznie język SAKO jest kompromisem między tymi tendencjami, a tym, że całkowite wykorzystanie możliwości maszyny i umożliwienie zapisu dowolnego problemu obliczeniowego lub logicznego uznano za pierwszoplanowe. Niemniej jednak język SAKO nie jest trudniejszy w nauce i w użyciu niż inne języki tej klasy.

Program napisany w języku SAKO przekształcony jest przez translator w program wynikowy. Program wynikowy jest optymalizowany zarówno ze względu na objętość pamięci, jaką zajmuje, jak i ze względu na czas potrzebny maszynie do jego wykonania. Tak więc translator nie tylko dokonuje tłumaczenia, ale również układają program wynikowy w ten sposób, by był on jak najbardziej efektywny. Te właśnie cechy translatora uzasadniają podane w poprzednim rozdziale stwierdzenie, że zastępuje on doświadczonego i wprawnego programistę.

System Automatycznego Kodowania stworzony został na podstawie opracowań teoretycznych Instytutu Maszyn Matematycznych PAN. Projektodawcami są prof. dr Leon Łukaszewicz i dr Antoni Mazurkiewicz oraz Zespół Zakładu Programowania IMM PAN.

5.1. STRUKTURA JĘZYKA SAKO

Program napisany w języku SAKO składa się ze zdań. Język SAKO podobnie jak inne języki automatycznego kodowania (autokody) jest językiem sformalizowanym. Tak więc znaczenie poszczególnych zdań jest ściśle określone, a budowa każdego z nich opisywana właściwym językowi systemem reguł.

W języku SAKO wyróżnia się następujące kategorie zdań

1. *Deklaracje* informujące translator o strukturze wprowadzanego programu. Po wykorzystaniu deklaracji przez translator są one pomijalne i nie wchodzą w skład programu wynikowego.

2. *Rozkazy* (makrorozkazy) tłumaczone przez translator na program wynikowy.

3. *Komentarze* pomijane przez translator i nie mające żadnego wpływu na program wynikowy, a służące jedynie programiście do zorientowania się w znaczeniu poszczególnych części programu.

W tablicy 5.1 podano wszelkie deklaracje i rozkazy używane w języku SAKO.

Tak jak w programach pisanych w języku maszyny (np. w języku PMC) kolejność występowania rozkazów w programie i ich wykonywania przez maszynę jest na ogół różna.

W języku SAKO rozkazy wykonywane są w następującej kolejności. Pierwszym rozkazem wykonywanym przez maszynę jest zawsze pierwszy rozkaz napisany w programie. Po wykonaniu rozkazu nie będącego rozkazem sterującym maszyna przystępuje do realizacji rozkazu następującego po nim w programie. Zmiana kolejności wykonywania rozkazów może być zrealizowana po rozkazie sterującym bezwarunkowym lub po rozkazie sterującym warunkowym, jeżeli warunek został spełniony. Ostatnim rozkazem wykonywanym jest rozkaz STOP.

Program napisany w języku SAMO zakończony jest zawsze deklaracją KONIEC, informującą translator o zakończeniu wprowadzania programu.

TABLICA 5.1

LISTA ZADAŃ W JĘZYKU — SAKO

1. DEKLARACJE

1. ROZDZIAŁ: nazwa
2. PODPROGRAM: (lista wyników) = nazwa (lista argumentów)
3. BLOK: (n, m, ..., k): lista
4. STRUKTURA (I, J, ..., K): lista
5. TABLICA (n, m, ..., k): nazwa
6. TABLICA OKTALNA (n, m, ..., k): nazwa
7. CAŁKOWITE: lista
8. SKALA DZIESIETNA PARAMETROW a
9. SKALA BINARNA PARAMETROW n
10. JĘZYK SAS
11. JĘZYK SAKO
12. KONIEC

2. ROZKAZY STERUJĄCE

1. SKOCZ DO
2. SKOCZ WEDŁUG I: α, β, γ, δ, ..., ω
3. POWTORZ OD α: I=J(K)L
4. IDZ DO ROZDZIAŁU: nazwa
5. WROC
6. GDY A>B: α, INACZEJ β
7. GDY A+B: α, INACZEJ β
8. GDY BYŁ NADMIAR: α, INACZEJ β
9. GDY KLUCZ I: α, INACZEJ β
10. STOP α

3. ROZKAZY ARYTMETYCZNE I BOOLOWSKIE

1. A=B
2. A≡B
3. (lista) = nazwa podprogramu (lista)
4. PODSTAW: nazwa podprogramu (lista)
5. USTAW SKALE DZIESIETNE: I
6. USTAW SKALE BINARNE: I
7. ZWIEKSZ SKALE DZIESIETNE O I: lista
8. ZWIEKSZ SKALE BINARNE O I: lista

TABL. 5.1 cd.

- 4. ROZKAZY WEJSCIA I WYJSCIA**
1. CZYTAJ: lista
 2. CZYTAJ OKTALNIE: lista
 3. CZYTAJ WIERSZ: nazwa bloku
 4. DRUKUJ (I, J): lista zmiennych
 5. DRUKUJ WIERSZ: nazwa bloku
 6. DRUKUJ SLOWO: lista zmiennych
 7. TEKST
 8. TEKST WIERSZY n:
 9. SPACJA n
 10. LINIA n
- 5. ROZKAZY WSPOLPRACY Z BEBNEM**
1. CZYTAJ Z BEBNA OD I: lista
 2. PISZ NA BEBEN OD I: lista

Zdania SAKO budowane są z następujących znaków:

1. Dwudziestu pięciu liter alfabetu łacińskiego: A, B, ..., Z
2. Cyfr: 0, 1, ..., 9
3. Symboli relacji i operacji: + - × / = * >
4. Separatorów . , : ;
5. Spacji (odstępów między znakami)

Pewne określone ciągi znaków (nie licząc spacji) nazywane są *zwrotami* języka SAKO. Zwrotami są na przykład

CZYTAJ

PISZ

Pewne ciągi znaków o określonej budowie nazywane są *wyrażeniami*. Przykładami wyrażeń są

liczby

zmienne i funkcje

wyrażenia arytmetyczne i boolowskie

Inne ciągi znaków konstruowane dowolnie, lecz mające ścisłe określony początek i koniec nazywane są *tekstami*. Tekstami są komentarze, na przykład

OBJETOSC V=

Zdania języka SAKO składają się ze zwrotów, wyrażeń i tekstów. Zwroty, wyrażenia i teksty, które mają tworzyć zdanie, muszą być dobrane w ten sposób, by utworzone zdanie stanowiło jedno z dopuszczalnych w języku SAKO zdań.

Jakkolwiek dopuszczalne zdania SAKO mają różnorodną konstrukcję, większość z nich budowana jest według następujących reguł:

zwrot — część operacyjna rozkazu lub deklaracji

wyrażenie — parametry zdaniowe

zwrot — dwukropek

na przykład

CZYTAJ : A

DRUKUJ (4, 5) : X1

To, że większość zdań SAKO ma podobną budowę, stanowi poważne ułatwienie dla użytkownika uczącego się programowania w tym języku.

5.2. POJĘCIA PODSTAWOWE W JĘZYKU SAKO

Niżej zostanie podany opis, niektórych pojęć podstawowych języka SAKO. Opis ten nie zawiera wszystkich pojęć i dlatego zainteresowanych odsyła się do literatury.

Liczبę ułamkową zapisuje się w sposób ogólnie przyjęty w matematyce, przy czym część całkowita oddzielona jest od części ułamkowej kropką, na przykład

— 38.1234

Oprócz liczby ułamkowej występuje w SAKO pojęcie liczby całkowitej, na przykład

+157

Zarówno liczby całkowite jak i liczby ułamkowe mogą być grupowane w bloki. Blok liczbowy jest to skończona i uporządkowana grupa liczb (całkowitych lub ułamkowych) oznaczona wspólnym symbolem. Każdemu

elementowi bloku odpowiada układ indeksów wyznaczających jego miejsce w bloku (indeksy liczne są od zera). Ilość indeksów nazywa się *wymiarem* (np. dwa indeksy — blok prostokątny).

Innym pojęciem podstawowym występującym w języku SAKO jest *zmienna prosta*. Jest to symbol przyjmujący w toku obliczeń różne wartości. Zmienną zapisuje się jako ciąg liter, cyfr i spacji rozpoczynający się od litery, na przykład

X112

DELTA

MEL34

W odróżnieniu od zmiennej prostej *zmienna indeksowa* jest elementem bloku określonym wartościami indeksów. Symbol zmiennej indeksowanej jest równocześnie symbolem bloku, którego elementem jest zmienna, na przykład

ALFA (0, 15)

COMP (1, 12, 34, 3)

W16 (0, ALFA, MIS (4, 0))

Indeksy mogą przyjmować jedynie wartości całkowite nieujemne.

Tak jak w języku SAS w języku SAKO występuje pojęcie numeru. Numery służą do oznaczania rozkazów. Numerem jest ciąg liter, cyfr i spacji rozpoczynający się od cyfry. Numer wypisuje się z lewej strony rozkazu oddzielając go od rozkazu nawiasem zamykającym, np. rozkaz **CZYTAJ : A** można oznaczyć numerem **1WAM** i zapisać ten rozkaz jako:

1WAM) CZYTAJ : A

Do rozkazów oznaczonych numerami można się odwoływać z różnych miejsc tego samego rozdziału programu.

Funkcje występujące w języku SAKO dzielą się na *funkcje języka* i *funkcje definiowane*. Funkcjami języka są najczęściej używane funkcje wchodzące na stałe do systemu SAKO. Funkcje te są podane w tabl. 5.2.

Funkcje definiowane wprowadza się rozkazem **PODPROGRAM** nazywając je zgodnie z zasadami obowiązującymi przy tworzeniu zmiennych prostych. Argumentem funkcji może być wyrażenie arytmetyczne.

TABLICA 5.2

FUNKCJE JĘZYKA SAKO

Skrót	Nazwa	Znaczenie
SIN (X)	Sinus	
COS (X)	Cosinus	
TNG (X)	Tangens	
ASN (X)	Arcus sinus	
ACS (X)	Arcus cosinus	
ATG (X)	Arcus tangens	Oznacza kąt, leżący w óciartce kątowej wyznaczonej przez znak X i znak Y, którego tangens jest równy Y/X
ARC (X, Y)	Arcus	
PWK (X)	Pierwiastek kwadratowy	
PWS (X)	Pierwiastek sześcienny	
LN (X)	Logarytm naturalny	
EXP (X)	Eksponent	
MAX (X, Y, ..., Z)	Maksimum	Oznacza największą z liczb X, Y, ..., Z
MIN (X, Y, ..., Z)	Minimum	Oznacza najmniejszą z liczb X, Y, ..., Z
MOD (X, Y)	Moduł	Oznacza resztę powstałą z dzielenia X przez Y (tylko dla całkowitych A i B)
SGN (X, Y)	Sigmaum	Oznacza liczbę $X \cdot \operatorname{sgn} Y$
ABS (X)	Wartość bezwzględna (absolutna)	Oznacza wartość absolutną X
ENT (X)	Część całkowita	Oznacza część całkowitą liczby X

Wyrażenie arytmetyczne ma sens ogólnie przyjęty w matematyce. Tak więc wyrażeniem arytmetycznym jest

$$\text{ALFA} \times Y * 2 + B / 3 = \text{PWK} (Y + B)$$

gdzie znaki $+$, $-$, \times , $*$, $/$ mają sens ogólnie przyjęty w matematyce, $*$ zaś jest symbolem potęgowania.

W SAKO przyjmuje się następujący priorytet wykonywania działań: Na wstępie obliczane są wartości funkcji i zmiennych indeksowanych. Następnie wykonywane jest potęgowanie, mnożenie, dzielenie, odejmowanie, dodawanie. Jak w konwencjonalnym zapisie matematycznym kolejność wykonywania działań w obrębie wyrażenia można ustalać używając nawiasów, przy czym działania zamknięte w większej ilości nawiasów wykonywane są wcześniej. Tak więc wyrażenie określające wartość pierwiastka równania kwadratowego zapisane w języku SAKO ma postać następującą

$$(-B + \sqrt{B^2 - 4 \times A \times C}) / (2 \times A)$$

5.3. NIEKTÓRE DEKLARACJE I ROZKAZY JĘZYKA SAKO

Tak jak w języku SAS i w języku SAKO używana jest deklaracja ROZDZIAŁ. Sygnalizuje ona translatorowi, że fragment programu znajdujący się z tą deklaracją, aż do najbliższej deklaracji ROZDZIAŁ lub KONIEC, ma niezależną symbolikę i jest jako całość umieszczany w pamięci wewnętrznej. Równocześnie w deklaracji podaje się nazwę rozdziału, na przykład.

ROZDZIAŁ

Wszystkie rozdziały danego programu umieszczone są w pamięci pomocniczej. Programista może zażądać przepisania danego rozdziału do pamięci wewnętrznej wykorzystując rozkaz: IDZ DO ROZDZIAŁU, na przykład

IDZ DO ROZDZIAŁU

Część rozdziału może zostać wyróżniona jako podprogram. Wyróżnienia tego dokonuje się deklaracją PODPROGRAM. W deklaracji tej podaje się listę wyników, nazwę podprogramu i listę argumentów, na przykład

PODPROGRAM (X, Y)=SUB (A, B, GAMMA)

Część programu występująca po deklaracji **PODPROGRAM** do najbliższej deklaracji **PODPROGRAM ROZDZIAŁ** lub **KONIEC** traktowana będzie jako całość. Podprogram ten można wykorzystywać w dalszej części programu. Przed wywołaniem podprogramu wartości jego argumentów można określić rozkazem **PODSTAW**, na przykład:

PODSTAW : SUB (+3.14, W, Z4)

W wyniku zrealizowania tego rozkazu odpowiednie argumenty podprogramu **SUB** przyjmą podane w rozkazie **PODSTAW** wartości. Konieczność uprzedniego podstawienia wartości argumentów przed wywołaniem programu nie istnieje. Wartości argumentów można podstawać równocześnie z wywoływaniem podprogramu w celu jego wykonania. W tym przypadku używa się *formuły operacyjnej* lub też formuły arytmetycznej, jeżeli podprogram określa funkcję (ma tylko jeden wynik liczbowy).

Jeżeli na trzech argumentach (+3.14, W, Z4) chcemy wykonać operacje oznaczone wspólną nazwą **SUB**; o których wiadomo, że w ich wyniku powstają dwie wartości będące rezultatami operacji **SUB** i rezultaty te chcemy nazwać **POL** i **S11**, to piszemy formułę operacyjną

(POL, S11)=SUB (+3.14, W, Z4)

Formuła operacyjna jest uogólnieniem *formuły arytmetycznej*. Formuła arytmetyczna jest pojęciem dobrze znany z codziennej praktyki i umożliwia obliczenie wartości funkcji (pojedynczej wartości), na przykład:

W=A+SIN (A+Z)

lub

X1=(-B+PWK (B * 2 — 4×A×C))/(2×A)

Ponieważ programista może wykorzystać zarówno rozkaz **PODSTAW** jak i formuły, może dowolnie ustalać wartości argumentów podprogramu bez wywoływania podprogramu (**PODSTAW**) lub podstawać równocześnie z wywołaniem (formuły). Niżej podano przykład deklarowania podprogramu i odwoływania się do podprogramu.

PODPROGRAM : (X, Y)=SUB (A, B, GAMMA)

V=PWK (A)

U = PWK (B)

X = V * 3 + U + GAMMA

Y = U * 3 + V - GAMMA

WROC

PODSTAW : SUB (+3.14, , Z4)

(POL, S11) = SUB (., W, .)

Na przykładzie tym pokazano, w jaki sposób można stosować równocześnie rozkaz PODSTAW i formułę operacyjną. Ponadto można spostrzec, że ostatnim rozkazem podprogramu jest rozkaz WROC powodujący powrót po wykonaniu podprogramu do programu głównego (w przykładzie do rozkazu następnego po formule operacyjnej). Ścisłe omówienie budowy i wykorzystania podprogramów w języku SAKO zawierają prace [29] i [31].

W języku SAKO zachodzi konieczność deklarowania wartości, które przyjmują wyłącznie wartości całkowite (np. indeksy). Jest to realizowane deklaracją CALKOWITE, na przykład:

CALKOWITE : A, B

W programie pisany w języku SAKO istnieje możliwość pisania określonych fragmentów programu w języku SAS. W tym celu w toku pisania programu należy użyć deklaracji: JEZYK SAS i pisać w języku SAS aż do deklaracji JEZYK SAKO.

Oprócz omówionych już rozkazów sterujących: IDZ DO ROZDZIAŁU, WROC opisane zostaną rozkazy SKOCZ DO, SKOCZ WEDLUG, oraz POWTORZ będące bezwarunkowymi rozkazami sterującymi. Rozkaz SKOCZ DO jest rozkazem skoku bezwarunkowego. W rozkazie tym podawany jest numer rozkazu, do którego należy zrealizować skok, na przykład

— — —
IWAM) CZYTAJ: A
 — — —
 — — —

SKOCZ DO IWAM
 — — —

Rozkaz SKOCZ WEDLUG I:IWAM, 4, 9JK powoduje skok zależnie od wartości I. Wartości I są całkowite nieujemne. Tak więc gdy wartość I wynosi 0, wykonany jest skok do rozkazu o numerze 1WAM, gdy I=1, do rozkazu o numerze 4, gdy I=2, do rozkazu o numerze 9JK itd.

Rozkaz POWTORZ OD 7SI:A=1(1)20 oznacza, że odcinek programu między rozkazem o numerze 7SI a samym rozkazem POWTORZ będzie powtarzany dopóty, dopóki wartość A, będąca przy pierwszym powtórzeniu jednością, zwiększa się przy każdym powtórzeniu o jedność, nie osiągnie wartości 20. Tak więc rozkaz POWTORZ realizuje pętle programu, na przykład:

— — —
*** 7SI) X=A+B**
 — — —
 — — —

POWTORZ OD 7SI:A=1(1)20

W języku SAKO realizowane są również rozkazy warunkowe, umożliwiające maszynie podjęcie prostych decyzji co do dalszej drogi realizowania programu zależnie od uzyskanych rezultatów. Rozkazami tymi są: GDY A>B:4SAO, INACZEJ 3CARO, lub GDY A=B:SAO, INACZEJ 3CARO. W rozkazach tych, gdy podany warunek jest spełniony, następującym rozkazem wykonywanym jest rozkaz o numerze 4SAO, w przeciwnym razie rozkaz o numerze 3CARO.

Wczytanie danych z urządzenia wejścia do pamięci maszyny umożliwia rozkaz CZYTAJ, na przykład:

CZYTAJ: A, B, C

Rozkaz ten powoduje podstawienie wczytanych trzech liczb jako wartości A, B, C.

Drukowanie wyników umożliwia rozkaz DRUKUJ, na przykład
DRUKUJ (5,5): OMEGA

powoduje wydrukowanie wartości zmiennej OMEGA, przy czym przewidziany został druk pięciu cyfr przed i pięciu cyfr za przecinkiem.

Powyższe omówienie rozkazów i deklaracji języka SAKO pozwoli na podanie prostych przykładów programów napisanych w tym języku.

5.4. PRZYKŁADY PROSTYCH PROGRAMÓW W JĘZYKU SAKO

1. Należy obliczyć sumę określona wzorem

$$A = \sum_{X=0}^{99} \left(BX^2 + \frac{CX}{X+1} \right)$$

- 1 K) OBLICZANIE SUMY STU SKŁADNIKOW
- 2 CALKOWITE: X
- 3 CZYTAJ: B, C
- 4 A=0
- 5 * 99S) A=A+B×X * 2+C×X/(X+1)
- 6 POWTORZ OD 99S: X=0(1)99
- 7 LINIA
- 8 DRUKUJ (4.2): A
- 9 STOP NASTEPNY
- 10 KONIEC

Pierwszy wiersz jest komentarzem i zostanie pominięty przy wczytywaniu programu. Następnie wartość X jest deklarowana jako całkowita.

Pozostałe wartości występujące w programie są rzeczywiste. Rozkaz w wierszu 3 powoduje wczytanie wartości B i C. Formuła zawarta w wierszu 4 ustala wartość początkową A jako zerową. Formuła zawarta w wierszu 5 powtórzona sto razy (rozkazem zawartym w 6 wierszu) dla wartości X=0,1,2,...99 wyznacza żądaną wartość sumy. Rozkaz LINIA stanowi przygotowanie do drukowania z nowego wiersza. Rozkaz DRUKUJ po-

woduje wydrukowanie wartości A. Program kończy rozkaz STOP. Deklaracja KONIEC informuje translator o zakończeniu wprowadzenia programu.

Uwaga. Wyżej podany program oraz programy następne nie zawierają deklaracji i rozkazów dotyczących skali. Pojęcie skali nie jest wprowadzane w tej pracy.

2. Należy obliczyć pierwiastki równania kwadratowego

$$Ax^2 + Bx + C = 0$$

Jeśli pierwiastki rzeczywiste istnieją (tzn. gdy $B^2 - 4AC \geq 0$), należy obliczone pierwiastki wydrukować. Jeżeli pierwiastków rzeczywistych nie ma, należy zaprzestać obliczeń (zrealizować rozkaz STOP).

1 K) PIERWIASTKI ROWNANIA KWADRATOWEGO

2 1WAK) CZYTAJ: A, B, C

3 GDY $4 \times A \times C > B^2$ 2 : 1B, INACZEJ 1P

4 1P) DEL = PWK ($B^2 - 4 \times A \times C$)

5 $X_1 = (-B - \sqrt{DEL}) / 2 \times A$

6 $X_2 = (-B + \sqrt{DEL}) / 2 \times A$

7 LINIA

8 DRUKUJ (3.5) : X1

9 LINIA

10 DRUKUJ (3.5) : X2

11 1B) STOP 1WAK

12 KONIEC

Pierwszy wiersz zawiera komentarz. Rozkaz w wierszu drugim powoduje wprowadzenie współczynników równania. Wiersz trzeci zawiera rozkaz warunkowy. Maszyna sprawdza, czy wyróżnik jest nieujemny. W przypadku ujemnego wyróżnika pierwiastki rzeczywiste nie istnieją i maszyna zatrzyma się na rozkazie o numerze 1B (wiersz 11). Jeżeli pierwiastki rzeczywiste istnieją, rozpoczyna ich obliczanie obliczając pierwiastek kwadratowy z wyróżnika (wiersz 4). Następne rozkazy (wiersz 5, 6) powodują obliczenie wartości tych pierwiastków, a następne rozkazy zawarte w wierszach 7—10 — drukowanie wartości tych pierwiastków (każda wartość od nowego wiersza). Po wydrukowaniu maszyna staje na rozkaz STOP. W przypadku naciśnięcia klucza „Start” maszyna rozpocznie

powtórne wykonywanie programu poczynając od rozkazu 1WAK, rozkaz stopu bowiem wskazuje ten właśnie numer (STOP 1WAK).

3. Należy obliczyć okres drgań wahadła fizycznego według przybliżonego wzoru

$$T = 2\pi \sqrt{\frac{I}{mgr}} \left(1 + \frac{A^2}{16}\right)$$

w którym:

T — okres drgań wahadła,

m — masa wahadła,

I — moment bezwładności wahadła względem punktu zawieszenia,

r — odległość środka ciężkości od punktu zawieszenia,

g — przyspieszenie grawitacyjne,

A — amplituda kątowa drgań wahadła.

Dla ustalonych wartości parametrów m , g , I należy obliczyć wartość T zależnie od wartości r i A , przy czym r zmienia się co 1 w zakresie od 1 do 10 (w ustalonych jednostkach), a amplituda A co pi szesnastych od pi szesnastych do pi pół.

Uzyskanych 80 wartości należy wydrukować.

1 K) TABLICOWANIE OKRESÓW DRGAN WAHADŁA FIZYCZNEGO

2 1) CZYTAJ: I, M, G, PI

3** 2) $T = 2 \times \text{PI} \times \text{PWK} (I/(M \times G \times R)) \times (1 + A * 2/16)$

4 LINIA

5 DRUKUJ (4.4) : T

6 PIPOŁ = PI/2

7 PISZESNASTE = PI/16

8 POWTORZ OD 2 : A = PISZESNASTE (PISZESNASTE)
PIPOL

9 POWTORZ GD 2 : R = 1.0(1.0)10.0

10 STOP 1

11 KONIEC

Pierwszy wiersz jest komentarzem. Rozkaz CZYTAJ powoduje wczytanie wartości I, M, G, PI. Wiersz trzeci zawiera formułę arytmetyczną określającą okres drgań wahadła. Wiersze 4 i 5 zawierają rozkazy przygotowania drukowania i drukowanie wartości okresu drgań. Wiersze 6 i 7 za-

wierają przygotowania dla rozkazu POWTORZ (wyznaczają odpowiednie ułamki liczby PI). Wiersz 8 zawiera rozkaz POWTORZ realizujący ośmio-krotne powtórzenie obliczenia wartości T dla kolejnych wartości A (przy ustalonym R). Przy każdym powtórzeniu wartości T są drukowane od nowego wiersza. Wiersz 9 zawiera rozkaz POWTORZ realizujący dziesięciokrotne powtórzenie tablicowania wartości T, przy czym każde powtórzenie obejmuje tablicowanie wartości T dla wszystkich A przy ustalonym R. Tak więc każdemu powtórzeniu odnoszącemu się do R odpowiada 8 powtórzeń odnoszących się do A. W sumie maszyna wydrukuje 80 wartości, co jest zgodne z warunkami zadania. Po zakończeniu obliczeń i drukowań maszyna wykona rozkaz STOP 1.

4. Należy znaleźć pierwiastek równania przestępnego w przedziale $(0, 1)$

$$f(x) = 1 - \sum_{i=1}^{50} g_i e^{-ix} = 0$$

przy czym wiadomo, że

$$0 \leq g_i \leq 1$$

oraz

$$\sum_{i=1}^{50} g_i > 1$$

Przy tych założeniach wartość $f(x)$ dla $x=0$ jest ujemna, natomiast dla $x=1$ dodatnia. Ponadto funkcja jest monotoniczna, a więc istnieje dokładnie jeden pierwiastek w przedziale. Pierwiastek ten będzie poszukiwany metodą dzielenia przedziału na połowy.

Jeżeli w przedziale (a, b) znajduje się pierwiastek, to

$$f(a) \times f(b) < 0 \quad (1)$$

Należy wyznaczyć

$$f\left(\frac{a+b}{2}\right)$$

i jeśli

$$f(a) \times f\left(\frac{a+b}{2}\right) < 0 \quad (2)$$

to pierwiastek znajduje się w przedziale $\left(a, \frac{a+b}{2}\right)$, jeżeli natomiast

$$f\left(\frac{a+b}{2}\right) \times f(b) < 0 \quad (3)$$

to pierwiastek znajduje się w przedziale $\left(\frac{a+b}{2}, b\right)$.

Postępowania takie można powtarzać aż do osiągnięcia dowolnie małego przedziału.

Jeżeli ani nierówność (2), ani nierówność (3) nie jest spełniona, wtedy pierwiastek równania wynosi dokładnie $\frac{a+b}{2}$, żadna bowiem z dwóch nierówności nie jest spełniona tylko wówczas, gdy

$$f\left(\frac{a+b}{2}\right) = 0 \quad (4)$$

Obliczenie pierwiastka będzie kontynuowane dopóty, dopóki nie zostanie warunek

$$b-a < \varepsilon \quad (5)$$

przy czym ε przyjmuje się jako dokładność obliczania pierwiastka.

W przypadku spełnienia warunku (5) jako wartość pierwiastka przyjmuje się

$$X = \frac{a+b}{2} \quad (6)$$

Ponieważ w zadanym przedziale funkcja $f(x)$ jest monotonicznie rosnąca, przeto wystarczy badać, czy wartość funkcji w punkcie $\frac{a+b}{2}$ jest ujemna, dodatnia lub równa zeru i odpowiednio postępować tak, jakby były spełnione nierówności (2), (3) lub warunek (4).

K) WYZNACZANIE PIERWIASTKA ROWNANIA PRZESTEPNEGO

CALKOWITE : I

BLOK (0,49) : G

CZYTAJ: E,* G

A=0

B=1

1T) H=(A+B)/2

GDY E > B-A: 1FIN, INACZEJ NASTEPNY

F=1

*2) F=F-G(I-1)×EXP (I×(-H))

POWTORZ OD 2: I=1(1)50

GDY F=0: 1FIN, INACZEJ NASTEPNY

GDY F>0: 1P, INACZEJ NASTEPNY

A=H

SKOCZ DO 1T

1P) B=H

SKOCZ DO 1T

1FIN) X=H

DRUKUJ (1.9) : X

STOP 1T

KONIEC

5.5. UWAGI O TWORZENIU PROGRAMU WYNIKOWEGO W JĘZYKU SAKO

Tworzenie programu wynikowego dla programu napisanego w języku SAKO przebiega w dwu etapach. Pierwszym etapem jest tłumaczenie języka SAKO na język WJS (Wewnętrzny Język Symboliczny). Język WJS będący wewnętrzną postacią języka SAS, posiada dyrektywy tak jak język SAS. Deklaracje języka SAKO zastępowane są na ogół dyrektywami języka WJS. Niektóre z deklaracji oraz pozostałe zdania języka SAKO zastępowane są w procesie tłumaczenia sekwencjami rozkazów napisanych w języku WJS. W procesie tym każdemu zwrotowi lub działaniu odpowiada szkielet sekwencji zawierający tylko części operacyjne rozkazów. Symbole użyte w zdaniu lub argumenty działań wpisane są do tego szkieletu jako adresy symboliczne (patrz opis języka SAS).

Drugim etapem tłumaczenia jest przejście od języka WJS do języka wewnętrznego maszyny. Zadanie to wykonuje program zwany *Adresatorem*. Program tłumaczony jest rozdziałami, które są traktowane jako osobne całości. Dokładny opis procesu przekształcania programu napisanego w języku SAKO w program wynikowy podaje literatura [7].

Cena zł 27.-



T04154030