

A 4947 I / 1

Prace Naukowe Centrum Obliczeniowego

Politechniki Wrocławskiej

Seria: Studia i Materiały

1

1

Zagadnienia oprogramowania systemów komputerowego wspomagania dydaktyki

Wrocław 1980



Problems of software of systems for computer-aided teaching process

Contents

Artur KLAJN, Zbigniew SZPUNAR, A concept of program module for use in conversational procedures	3
Franciszk KL'USKA, Base des données sur les étudiants	17
Dariusz KRAJEWSKI, Compiler efficiency testing	23
Ludwik KUŻNIARZ, A project of conversational Fortran compiler	37
Andrzej PILAWSKI, On the problem of the difficulty degree of exercises used in CAI systems	45
Jerzy PISARSKI, Lech TUZINKIEWICZ, Conversational minisystem of arithmetical expression PT78	61
Zbigniew SZPUNAR, Document generation method for use in the process of data extraction from data bases	75

Spis rzeczy

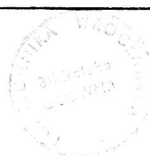
Artur KLAJN, Zbigniew SZPUNAR, Koncepcja modułu obsługi dialogu dla procedur konwersacyjnych	3
Franciszek KL'USKA, Baza danych zawierająca informacje o studentach . .	17
Dariusz KRAJEWSKI, Badanie efektywności kompilatorów	23
Ludwik KUŻNIARZ, Projekt konwersacyjnego kompilatora języka Fortran	37
Andrzej PILAWSKI, O zagadnieniu stopnia trudności zadań kontrolnych w systemach CAI	45
Jerzy PISARSKI, Lech TUZINKIEWICZ, Konwersacyjny minisystem wy- rażeń arytmetycznych PT78	61
Zbigniew SZPUNAR, Metoda generacji dokumentów w procesie wyprowa- dzania informacji z baz danych systemów informatycznych	75

Prace Naukowe Centrum Obliczeniowego
Politechniki Wrocławskiej

1

Seria:
Studia i Materiały

1



Zagadnienia oprogramowania systemów komputerowego wspomagania dydaktyki



Wydawnictwo Politechniki Wrocławskiej · Wrocław 1980

Redaktor naczelny
Marian KLOZA

Redaktor naukowy
Antoni MAZURKIEWICZ

Opracowanie redakcyjne
Maria KOPEĆ

Korekta
Helena ŚNIEŻYK

WYDAWNICTWO POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

Nakład 200 + 75 egz. Ark. wyd. 5,75. Ark. druk. 5³/₈. Papier offset. kl. V, 80 g. B1.
Przekazano do druku w lipcu 1980 r. Druk ukończono we wrześniu 1980 r.
Zakład Graficzny Politechniki Wrocławskiej. Zam. nr 8385/80 – B-10 – Cena zł20,-

*Procedura dialogowa,
moduł wejściowy*

Artur KLAJN^{*)}, Zbigniew SZPUNAR^{*)}

Koncepcja modułu obsługi dialogu dla procedur konwersacyjnych

Przedstawiono koncepcję budowy modułu obsługi dialogu dla procedur konwersacyjnych. Podano opis formalny struktury modułu, opartej na grafie opisującym funkcje realizowane przez procedurę, oraz opisano algorytm działania modułu wejściowego. Model został wykorzystany do realizacji programowych procedur Informatycznego Systemu Obsługi Procesu Nauczania DIALOS/MLK, zbudowanego w Centrum Obliczeniowym Politechniki Wrocławskiej.

1. WSTĘP

Wraz z dynamicznym rozwojem systemów konwersacyjnych daje się zauważyć wzrost popularności dialogowych procedur programowych, realizujących wiele funkcji systemów informatycznych, pracujących w czasie rzeczywistym. Specyfika działania tych procedur (realizacja i obsługa dialogu operator-maszyna cyfrowa, wykonywanie poleceń w trybie natychmiastowym), rzutuje na metodykę ich projektowania, rodzi wiele istotnych i nowych problemów. Na przykład od założonego sposobu komunikacji operatora z procedurą zależy zarówno łatwość wykonania procedury, jak też efektywność jej działania.

Istnieje wiele metod prowadzenia dialogu pomiędzy człowiekiem a maszyną cyfrową [1]. W większości przypadków należą one do jednego z następujących typów:

^{*)} Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

1. Dialog inicjowany przez operatora - większość par czynności instrukcja-odpowiedź zapoczątkowuje człowiek (np. metoda prostych pytań, wprowadzenie informacji w języku naturalnym).

2. Dialog inicjowany przez maszynę - większość par czynności podanych wyżej zapoczątkowuje maszyna (np. metoda prostych instrukcji dla operatora, wybór z repertuaru).

Istnieją jednak dialogi, w których zasada ta nie obowiązuje i żadna ze stron biorących udział w dialogu nie jest przez cały czas stroną inicjującą. Dialog taki zwany jest dialogiem hybrydowym [1] i wydaje się najbardziej uniwersalny. W skład procedury wykorzystującej ideę dialogu hybrydowego musi wchodzić wygodny język komunikacji człowiek-maszyna cyfrowa, moduł wykonawczy realizujący polecenia użytkownika oraz pewien moduł wejściowy spełniający rolę pośrednika pomiędzy operatorem a modułem wykonawczym procedury. Od struktury i sposobu działania modułu wejściowego zależy w znacznej mierze efektywność procedury oraz komfort prowadzenia dialogu.

W niniejszej pracy przedstawiono koncepcję budowy modułu wejściowego procedury dialogowej. Podano opis formalny struktury modułu, zaprezentowano metodologiczne podstawy jego projektowania oraz opisano algorytm działania.

2. SFORMUŁOWANIE PROBLEMU

Dany jest zbiór informacji (baza danych) oraz zbiór wymagań użytkownika dotyczący obsługi bazy danych. Wymagania użytkownika dają się przedstawić w postaci grup żądań różnych rodzajów (np. zapis informacji, wyprowadzanie informacji), w ramach których można wyodrębnić pewne potrzeby elementarne (np. zapis pojedynczego rekordu, zapis grupy rekordów). Potrzeby występujące w ramach jednej grupy można przedstawić w postaci wariantów operacji jednego rodzaju.

Należy zbudować moduł wejściowy dialogowej procedury obsługi bazy danych, realizujących żądane funkcje, wyposażony w wygodny język komunikacji pomiędzy użytkownikiem a procedurą i zapewniający możliwość prowadzenia dialogu hybrydowego.

3. KONCEPCJA MODUŁU WEJŚCIOWEGO PROCEDURY DIALOGOWEJ

3.1. Określenia podstawowe

Niech wymagania użytkownika będą określone zbiorem

$$F = \{F^1, \dots, F^i, \dots, F^n\}, \quad (1)$$

gdzie

$$F^i = \{f_1^i, \dots, f_j^i, \dots, f_{k_1}^i\} \quad (2)$$

jest podzbiorem operacji, których wykonanie sprowadza się do uaktywnienia tego samego fragmentu modułu wykonawczego. Operację f_j^i nazwiemy funkcją elementarną procedury, podzbiór operacji F^i - makrofunkcją.

Każdą z makrofunkcji (2) można przedstawić w postaci drzewa

$$D^i = \langle E^i, U^i \rangle, \quad (3)$$

gdzie

$$E^i = \{e_0^i, e_1^i, \dots, e_j^i, \dots, e_{m_1}^i\} \quad (4)$$

jest zbiorem wierzchołków drzewa, natomiast

$$U^i = \{u_1^i, \dots, u_j^i, \dots, u_{n_1}^i\} \quad (5)$$

jest zbiorem gałęzi, $u_j^i = (e_{j_1}^i, e_{j_2}^i)$.

Zbiór wierzchołków można przedstawić w postaci sumy rozłącznych podzbiorów

$$E^i = \{e_0^i\} \cup E_s^i \cup E_k^i, \quad (6)$$

gdzie

$$e_0^i = \langle \text{sym}_0^i, \text{int}_0^i \rangle \quad (7)$$

jest wierzchołkiem początkowym drzewa (sym - symbol wierzchołka, int - jego interpretacja),

E_k^i - zbiór wierzchołków końcowych, tzn. wierzchołków drzewa incyden-nych tylko do jednego innego wierzchołka i nie będących wierzchołkiem początkowym,

E_s^i - zbiór pozostałych wierzchołków drzewa.

Każdy z wierzchołków drzewa, z wyjątkiem wierzchołka początkowego, jest opisany uporządkowaną czwórką

$$e_j^i = \langle \text{sym}_j^i, \text{int}_j^i, \text{zbow}_j^i, \text{typ}_j^i \rangle, \quad (8)$$

gdzie sym - symbol wierzchołka,

int - interpretacja wierzchołka,

zbw -- zbiór wartości przypisanych do wierzchołka,

typ -- typ wierzchołka.

Zbiór wierzchołków (4) można przedstawić również w postaci

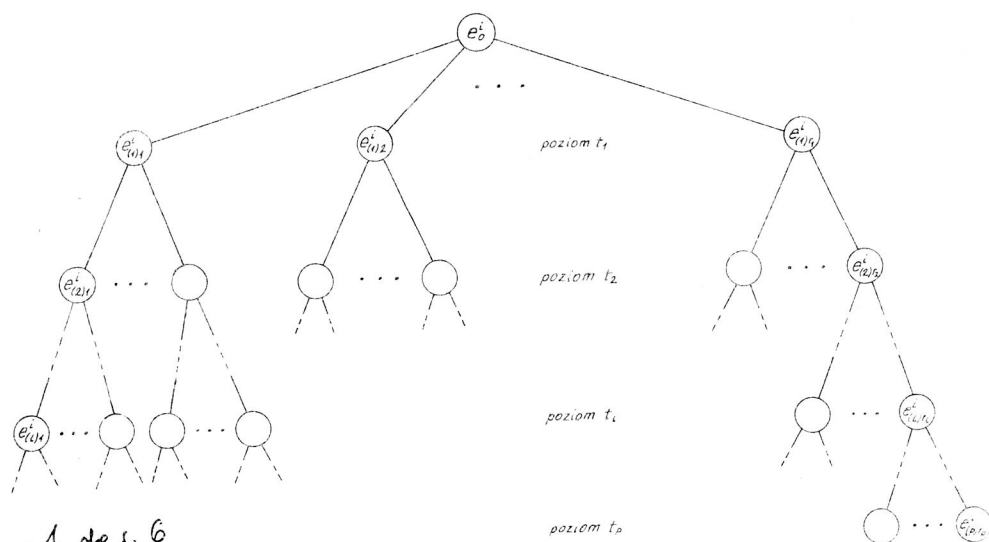
$$E^i = \bigcup_{l=1}^p E_l^i \quad (9)$$

gdzie p -- liczba poziomów w drzewie.

Elementami podzbiorów E_l^i , $l=1,2,\dots,p$, zbioru E^i są wierzchołki jednego typu w drzewie D^i . Każdy taki podzbiór E_l^i wyznacza poziom typu t_l wierzchołków w drzewie. Aby podkreślić przynależność poszczególnych wierzchołków e_j^i , $j=1,2,\dots,m_l$, drzewa D^i do podzbioru E_l^i , zamiast e_j^i będziemy pisać $e_{(l)j}^i$. Zatem

$$E_l^i = \{e_{(l)1}^i, e_{(l)2}^i, \dots, e_{(l)j}^i, \dots, e_{(l)r_l}^i\}. \quad (10)$$

Interpretację graficzną powyższego opisu zamieszczono na rysunku 1. Wierzchołek początkowy e_0^i w drzewie określa nazwę danej makrofunkcji, pozostałe zaś wraz z łączącymi je gałęziami definiują poszczególne jej funkcje elementarne.



Rys.1. Interpretacja graficzna drzewa D^i makrofunkcji F^i
Fig.1. Graphic interpretation of tree D^i of macrofunction F^i

Każdej z funkcji $f_j^i \in F^i$, odpowiada uporządkowany ciąg wierzchołków w drzewie makrofunkcji

$$f_j^i \leftrightarrow \{e_0^i, e_{(1)j_1}^i, e_{(2)j_2}^i, \dots, e_{(k)j_k}^i, \dots, e_{(q)j_q}^i\}, \quad q \leq p, \quad (11)$$

rozpoczynający się od wierzchołka początkowego i wyznaczający przejście do jednego z wierzchołków końcowych, przy czym kolejne dwa wierzchołki należą do różnych poziomów i są połączone gałęzią.

3.2. Odwzorowanie struktury modułu w graf

Dokonajmy przekształcenia zbioru drzew makrofunkcji w graf

$$G = \langle W, V \rangle, \quad (12)$$

gdzie W - zbiór wierzchołków grafu,

V - zbiór krawędzi.

Graf ten, zwany dalej grafem struktury modułu, powstaje w wyniku syntezy drzew wszystkich makrofunkcji według następujących zasad:

1. Zbiór W_0 wierzchołków początkowych grafu zawiera wszystkie wierzchołki początkowe poszczególnych drzew: $e_0^i \in E^i$, $i = 1, 2, \dots, n$, którym przyporządkowuje się symbole $w_{(0)i}$

$$W_0 = \{w_{(0)1}, w_{(0)2}, \dots, w_{(0)i}, \dots, w_{(0)n}\}. \quad (13)$$

2. Zbiór W_1 wierzchołków 1-tego poziomu grafu powstaje w wyniku operacji na wierzchołkach typu t_1 poszczególnych drzew przez złożenie wierzchołków mających różne symbole i (lub) interpretację oraz nałożenie wierzchołków mających identyczne symbole i interpretację z zachowaniem związków incydencji nakładanych wierzchołków z pozostałymi wierzchołkami w postaci krawędzi grafu

$$W_1 = \{w_{(1)1}, w_{(1)2}, \dots, w_{(1)j}, \dots, w_{(1)z_1}\}, \quad (14)$$

$$z_1 \leq \sum_{i=1}^n \bar{E}_1^i.$$

Funkcję złożenia Γ_Z i nałożenia Γ_N definiuje się następująco:

$$\Gamma_N(e_{(1)k_1}^i, e_{(1)k_2}^j) = w_{(1)j} \Leftrightarrow (sym_{(1)k_1}^i \equiv sym_{(1)k_2}^j) \wedge (int_{(1)k_1}^i \equiv int_{(1)k_2}^j), \quad (15)$$

$$\Gamma_Z(e_{(1)k_1}^i, e_{(1)k_2}^j) = \{w_{(1)j_1}, w_{(1)j_2}\} \Leftrightarrow \neg [(sym_{(1)k_1}^i \equiv sym_{(1)k_2}^j) \wedge \wedge (int_{(1)k_1}^i \equiv int_{(1)k_2}^j)],$$

przy czym

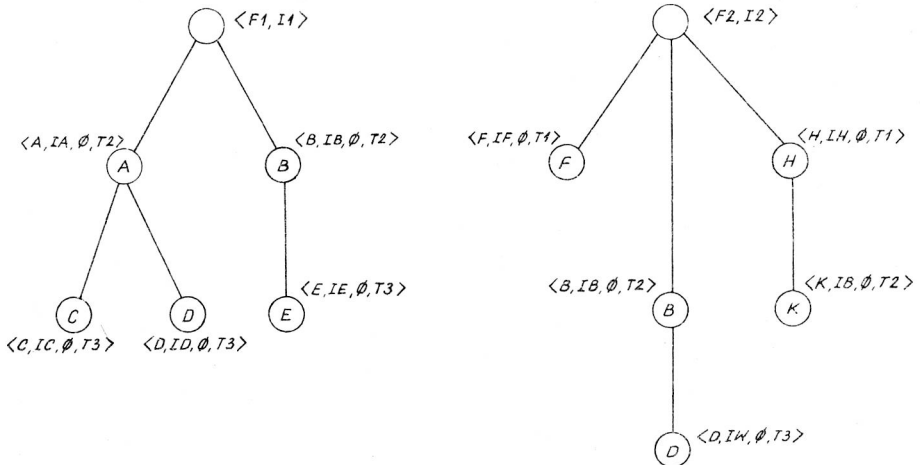
$$e_{(1)k_1}^i = \langle sym_{(1)k_1}^i, int_{(1)k_1}^i, zbw_{(1)k_1}^i, t_1 \rangle, \quad (16)$$

$$e_{(1)k_2}^j = \langle sym_{(1)k_2}^j, int_{(1)k_2}^j, zbw_{(1)k_2}^j, t_1 \rangle.$$

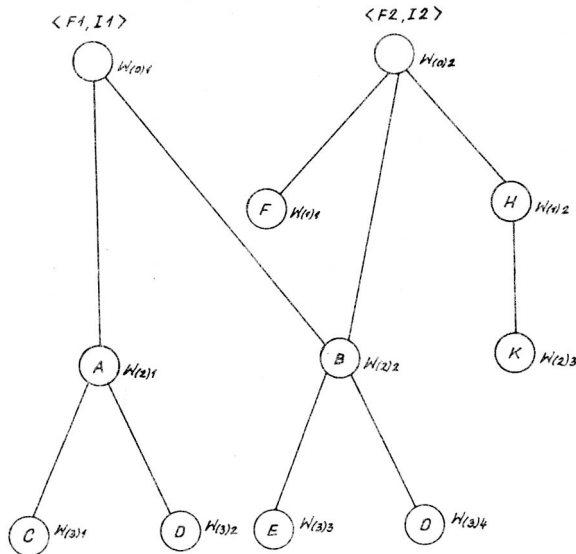
Znaczenia symboli użytych we wzorach (15) i (16) są takie same jak we wzorze (8).

3. Zbiór W_k wierzchołków końcowych grafu powstaje ze złożenia wierzchołków końcowych drzew i (lub) nałożenia ich na inne wierzchołki.

Na rysunku 2 zamieszczono dwa drzewa przykładowych makrofunkcji, a na rysunku 3 graf utworzony w wyniku syntezy tych drzew według podanych zasad.



Rys.2. Przykładowe drzewa makrofunkcji
Fig.2. Examples of macrofunction trees



$$\begin{aligned}
 W_0 &= \{W_{(0)1}, W_{(0)2}\} \\
 W_K &= \{W_{(0)1}, W_{(2)3}, W_{(3)1}, W_{(3)2}, W_{(3)3}, W_{(3)4}\} \\
 W_1 &= \{W_{(1)1}, W_{(1)2}\} \\
 W_2 &= \{W_{(2)1}, W_{(2)2}, W_{(2)3}\} \\
 W_3 &= \{W_{(3)1}, W_{(3)2}, W_{(3)3}, W_{(3)4}\}
 \end{aligned}$$

Rys.3. Graf struktury modułu wejściowego
Fig.3. Graph of input module structure

Każda ścieżka grafu od jednego z wierzchołków początkowych do odpowiedniego wierzchołka końcowego odpowiada, podobnie jak w drzewie makrofunkcji, określonej funkcji elementarnej. W ogólnym przypadku jednak uzyskany graf może zawierać, oprócz zadanych funkcji elementarnych (1) i (2), pewne funkcje przypadkowe (np. funkcje określone przez ścieżki $w_{(0)1}$, $w_{(2)2}$, $w_{(3)4}$ i $w_{(0)2}$, $w_{(2)2}$, $w_{(3)3}$ w podanym przykładzie). W celu wyeliminowania tego niepożądanego przypadku dla każdego wierzchołka, z wyjątkiem wierzchołków początkowych, wprowadza się tzw. listę makrofunkcji. Lista ta zawiera wykaz symboli makrofunkcji, do których - definiując określone ich funkcje elementarne - należy dany węzeł.

Węzły grafu - z wyjątkiem węzłów początkowych - są więc określone następująco:

$$w_{(1)1} = \langle \text{sym}_{(1)1}, \text{int}_{(1)1}, \text{zbw}_{(1)1}, \text{typ}_{(1)1}, \text{lis}_{(1)1} \rangle \quad (17)$$

gdzie $\text{lis}_{(1)1}$ - lista makrofunkcji węzła $w_{(1)1}$.

W podanym przykładzie węzły $w_{(2)2}$, $w_{(3)3}$, $w_{(3)4}$ są określone przez uporządkowane piętki

$$\begin{aligned}w_{(2)2} &= \langle B, IB, \emptyset, T2, \{F_1, F_2\} \rangle, \\w_{(3)3} &= \langle E, IE, \emptyset, T3, \{F_1\} \rangle, \\w_{(3)4} &= \langle D, IW, \emptyset, T3, \{F_2\} \rangle,\end{aligned}\tag{18}$$

co umożliwia wyeliminowanie przypadkowych funkcji elementarnych opisanych wyżej.

W razie syntezy rozbudowanych drzew makrofunkcji o dużej liczbie wspólnych fragmentów, wprowadzenie listy makrofunkcji umożliwia budowę grafu o minimalnej liczbie węzłów.

Tak uzyskany graf przedstawia wszystkie zadane funkcje elementarne i stanowi model struktury modułu wejściowego procedury dialogowej.

3.3. Model modułu wejściowego

Podstawową funkcją modułu wejściowego procedury dialogowej jest dokonanie transformacji polecenia użytkownika na postać umożliwiającą zainicjowanie działania oraz wybór wariantu realizacji odpowiedniego fragmentu modułu wykonawczego.

Działanie modułu wykonawczego jest związane z wykonaniem określonej funkcji elementarnej i kończone przekazaniem sterowania użytkownikowi, który może sformułować następne polecenie.

Na rysunku 4 przedstawiono ogólny schemat funkcjonalny procedury dialogowej, działającej w opisany sposób.

Komunikacja pomiędzy użytkownikiem a procedurą o przedstawionej strukturze odbywa się za pomocą sformalizowanego języka, który powinien zawierać elementy umożliwiające sprecyzowanie:

- makrofunkcji,
- zbioru parametrów definiujących funkcję elementarną danej makrofunkcji,
- zbiorów wartości parametrów.

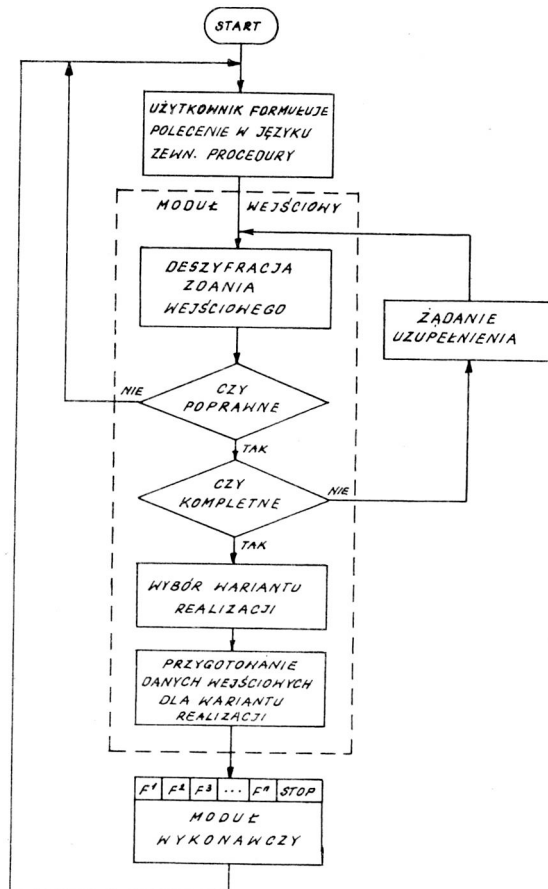
Jedną z możliwych postaci zdań języka, mających wymienione cechy, jest

$$\text{makro } p_1 x_1, p_2 x_2, \dots, p_k x_k.\tag{19}$$

gdzie makro - symbol makrofunkcji,

p_i - symbol parametru o określonej interpretacji i typie,

x_i - zbiór wartości parametru p_i .



Rys.4. Schemat funkcjonalny procedury dialogowej
Fig.4. Functional scheme of dialogue procedure

Zdanie o powyższej postaci, definiujące określoną funkcję elementarną, ma strukturę ścieżki grafu, której węzłem początkowym jest symbol makrofunkcji, a kolejnymi węzłami symbole parametrów. Zbiór x wartości parametru odpowiada analogicznemu zbiorowi zwb opisu węzła. Wobec tego graf jest zbiorem ścieżek odpowiadających funkcjom elementarnym, z których każda może być zdefiniowana prawidłowo skonstruowanym zdaniem języka (19). Wybór odpowiedniej funkcji elementarnej (odpowiedniego wariantu wejścia modułu wykonawczego) sprowadza się więc do deszyfracji zdania wejściowego i wskazania odpowiedniej ścieżki w grafie. Wybraną ścieżkę można przedstawić w postaci tzw. maski - uporządkowanego zbioru zawierającego cechy kolejnych węzłów ścieżki. Postać maski przedstawiono na rysunku 5.

	sym	int	zbw	typ	lis
W_{101}^1					
W_{111}^1					
W_{121}^1					
W_{131}^1					
W_{141}^1					
W_{151}^1					
W_{161}^1					
W_{171}^1					
W_{181}^1					
W_{191}^1					
W_{201}^1					
W_{211}^1					
W_{221}^1					
W_{231}^1					
W_{241}^1					
W_{251}^1					
W_{261}^1					
W_{271}^1					
W_{281}^1					
W_{291}^1					
W_{301}^1					
W_{311}^1					
W_{321}^1					
W_{331}^1					
W_{341}^1					
W_{351}^1					
W_{361}^1					
W_{371}^1					
W_{381}^1					
W_{391}^1					
W_{401}^1					
W_{411}^1					
W_{421}^1					
W_{431}^1					
W_{441}^1					
W_{451}^1					
W_{461}^1					
W_{471}^1					
W_{481}^1					
W_{491}^1					
W_{501}^1					
W_{511}^1					
W_{521}^1					
W_{531}^1					
W_{541}^1					
W_{551}^1					
W_{561}^1					
W_{571}^1					
W_{581}^1					
W_{591}^1					
W_{601}^1					
W_{611}^1					
W_{621}^1					
W_{631}^1					
W_{641}^1					
W_{651}^1					
W_{661}^1					
W_{671}^1					
W_{681}^1					
W_{691}^1					
W_{701}^1					
W_{711}^1					
W_{721}^1					
W_{731}^1					
W_{741}^1					
W_{751}^1					
W_{761}^1					
W_{771}^1					
W_{781}^1					
W_{791}^1					
W_{801}^1					
W_{811}^1					
W_{821}^1					
W_{831}^1					
W_{841}^1					
W_{851}^1					
W_{861}^1					
W_{871}^1					
W_{881}^1					
W_{891}^1					
W_{901}^1					
W_{911}^1					
W_{921}^1					
W_{931}^1					
W_{941}^1					
W_{951}^1					
W_{961}^1					
W_{971}^1					
W_{981}^1					
W_{991}^1					
W_{1001}^1					

Rys.5. Graficzne przedstawienie maski
Fig.5. Graphic representation of mask

Wiersz maski, odpowiadający węzłowi początkowemu grafu, określa wariant wejścia do modułu wykonawczego, pozostałe natomiast precyzują zadanie oraz dostarczają dane wejściowe dla wybranego fragmentu modułu.

O poprawności postaci utworzonej maski orzeka:

- nieidentyczność typów par węzłów dla wszystkich możliwych par,
- występowanie symbolu danej makrofunkcji na listach wszystkich węzłów w masce.

Rozmiar pionowy maski musi zapewniać możliwość opisanie ścieżki o największej liczbie węzłów, tzn. nie może być mniejszy niż liczba poziomów w grafie. Dla ścieżek nie mających węzła w danym poziomie grafu odpowiedni wiersz maski nie jest wykorzystany. Aby zapewnić tworzenie maski w sposób systematyczny, opracowano metodę, której algorytm przedstawiono w punkcie 3.4.

3.4. Algorytm działania modułu wejściowego

Algorytm działania modułu wejściowego można przedstawić w postaci następujących kroków:

KROK 1. Ustawić stan zerowy maski (wszystkie pola puste).

KROK 2. Pobrać zdanie wejściowe.

KROK 3. Zidentyfikować symbol makrofunkcji w zdaniu wejściowym i utożsamić go z odpowiednim węzłem początkowym grafu. Jeśli występujący w zdaniu wejściowym symbol makrofunkcji nie ma swojego odpowiednika w grafie, przejść do kroku 7. W przeciwnym wypadku przepisać cechy wybranego węzła do pierwszego wiersza maski. Zejść na pierwszy z poziomów dostępnych z wybranego węzła początkowego.

KROK 4. W zbiorze parametrów makrofunkcji w zdaniu wejściowym znaleźć parametr mający swój odpowiednik w postaci węzła na aktualnym poziomie grafu. Węzeł ten musi być incydentny z ostatnio rozpatrywanym węzłem. Jeśli taki odpowiednik nie występuje, zażądać podania paramet-

ru określonego typu, umieścić go w zdaniu wejściowym i powtórzyć krok 4. Jeśli występuje jedna para parametr-węzeł, to uzupełnić cechy węzła o zbiór wartości parametru i przejść do kroku 5. W pozostałych przypadkach przejść do kroku 7.

KROK 5. Przepisać cechy wybranego węzła do wiersza maski odpowiadającego danemu poziomowi grafu. Usunąć rozpatrywany parametr ze zdania wejściowego. Jeśli rozpatrywany poziom jest ostatnim z dostępnych dla danej makrofunkcji, przejść do kroku 6. W przeciwnym wypadku zejść na kolejny dostępny poziom grafu i przejść do kroku 4.

KROK 6. Jeśli zbiór parametrów w zdaniu wejściowym jest pusty, to koniec pracy modułu wejściowego. W przeciwnym wypadku przejść do kroku 7.

KROK 7. Sygnalizować błąd i przejść do kroku 1.

4. PRZYKŁAD DZIAŁANIA MODUŁU WEJŚCIOWEGO

Przedstawmy działanie modułu wejściowego prostej procedury obsługi bazy danych, zawierającej kartotekę studentów. Kartoteka składa się z rekordów (zapisów) jednego typu o strukturze jak na rysunku 6.

<i>numer rekordu</i>	<i>nazwisko i imię</i>	<i>data ur.</i>	<i>miejsce zam.</i>	<i>numer albumu</i>	<i>wydział</i>	<i>rok</i>	<i>grupa</i>
IR	NI	DU	MZ	NA	WY	RO	GR

Rys.6. Postać rekordu studenta .

Fig.6. Form of record of a student

Procedura ma realizować następujące makrofunkcje:

- zapis (Z),
- odczyt (O),
- aktualizacja (A).

Wprowadzmy trzy poziomy parametry makrofunkcji:

- przedmiot działania (T1),
- adres (T2),
- wartość poprawki (T3).

Na poziomie T1 mogą wystąpić:

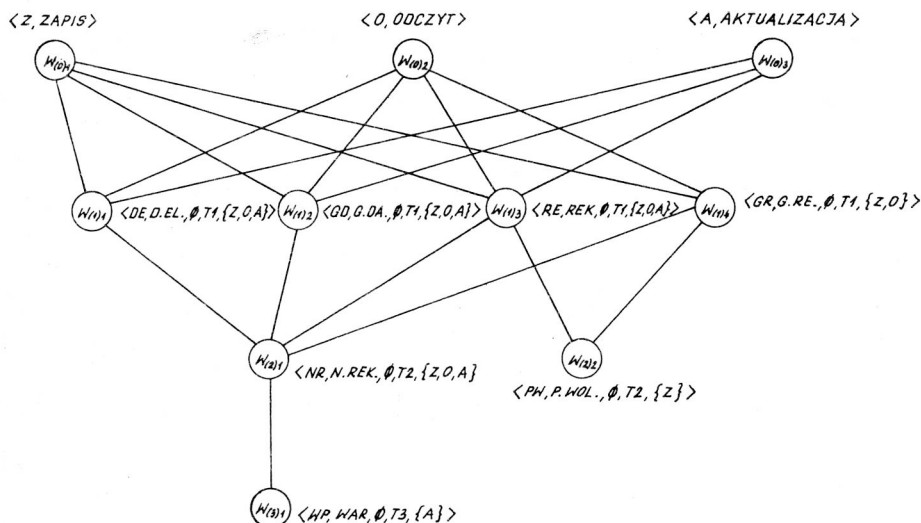
- dana elementarna (DE) - pojedyncze pole informacji w rekordzie studenta,
- grupa danych (GD) - zbiór danych elementarnych,
- rekord (RE),

- grupa rekordów (GR) (z wyjątkiem aktualizacji).

Na poziomie T2 mogą wystąpić:

- wskazanie numeru rekordu (NR),
- żądanie działania na pierwszym wolnym polu kartoteki (FW) (tylko dla zapisu rekordu lub grupy rekordów).

Poziom T3 (WP) dotyczy jedynie aktualizacji.



Rys.7. Graf struktury przykładowego modułu wejściowego
Fig.7. Structure graph of an example of input module

Na rysunku 7 przedstawiono graf syntezy zadanych makrofunkcji. Maska dla zbudowanego grafu ma 4 wiersze. Aby zapewnić możliwość określenia poszczególnych funkcji elementarnych, wprowadzmy język wejściowy postaci (19).

Każdemu poziomowi grafu odpowiada zbiór parametrów określonego typu. Dla wygody symbole makrofunkcji i parametrów w zdaniach języka będą identyczne z odpowiednimi symbolami w grafie.

Rozważmy działanie modułu wejściowego według podanego algorytmu (pkt 3.4) dla zdania: "zapisz w rekordzie 20 grupę danych: nazwisko i imię, numer albumu, wydział".

W sformalizowanym języku wejściowym zdanie to ma postać: Z GD(NI: KOWALSKI JAN, NA: 30893, WY: ELEKTRONIKA), NR(20).

Kolejne kroki algorytmu doprowadzą do następującego zapełniania maski:

1. Zapisanie cechy makrofunkcji

W(0)1	Z	ZAPIS
-------	---	-------

2. Spośród węzłów poziomu T1 parametrowi GD odpowiada węzeł $w_{(1)2}$

$w_{(1)2}$	GD	G.DA.	NI:KOWALSKI JAN, NA:30893, WY:ELEKTRONIKA	T1	Z, O, A
------------	----	-------	---	----	---------

3. Po usunięciu rozpatrywanego parametru zdanie wejściowe ma postać

Z NR(20)

Poziom T1 nie jest ostatnim z dostępnych poziomów makrofunkcji Z - następuje zejście na poziom T2. Spośród węzłów tego poziomu wybrany zostaje węzeł $w_{(2)1}$.

$w_{(2)1}$	NR	N.REK.	20	T2	Z, O, A
------------	----	--------	----	----	---------

Gdyby np. w zdaniu wejściowym zabrakło parametru NR, wówczas nastąpiłoby przekazanie sterowania użytkownikowi z żądaniem wprowadzenia brakującej informacji. Po usunięciu parametru NR zdanie wejściowe zawiera pusty zbiór parametrów, a poziom T2 jest ostatnim z dostępnych poziomów makroinstrukcji Z. Moduł kończy pracę i utworzona maska ma postać jak na rysunku 8.

Z	Zapis			
GD	G.DA	NI:KOWALSKI JAN, NA:30893, WY:ELEKTRO-NIKA	T1	Z, O, A
NR	N.REK	20	T2	Z, O, A
—	—	—	—	—

Rys.8. Maska przykładowej funkcji elementarnej
Fig.8. Mask of the example of elementary function

5. UWAGI KOŃCOWE

Zaprezentowana koncepcja budowy modułu wejściowego procedury dialogowej jest propozycją rozwiązania problemu obsługi dialogu pomiędzy użytkownikiem i procedurą. Przedstawienie funkcji realizowanych przez procedurę w postaci grafu pozwala na rozszerzenie możliwości procedury zarówno o funkcje elementarne (dołączenie ścieżki), jak też o nowe makrofunkcje przez dołączenie do grafu drzew makrofunkcji. Przyjęty model modułu wejściowego i postać zdań języka w dostateczny sposób zapewniają efektywność wykorzystania procedury, elastyczność dialogu i komfort jego prowadzenia.

Opisana metoda została zastosowana w konwersacyjnych procedurach wyprowadzania informacji z kartoteki studentów oraz w procedurach zakładania, aktualizacji i listowania zadań z kartoteki zadań Informatycznego Systemu Obsługi Procesu Nauczania DIALOS/MLK, eksploatowanych pod kontrolą systemów operacyjnych George 3 i Minimop.

LITERATURA

- [1] M a r t i n J., Dialog człowieka z maszyną cyfrową, Warszawa 1976, WNT.
- [2] S p r o w l s R.C., Management Data Bases, Santa Barbara 1976, Wiley/Hamilton.
- [3] Y o u r d o n E., Projektowanie systemów o działaniu bezpośrednim, Warszawa 1976, WNT.

Praca wpłynęła do Redakcji 1.12.1978 r.
Po poprawieniu 19.02.1980 r.

A CONCEPT OF PROGRAM MODULE FOR USE IN CONVERSATIONAL PROCEDURES

A concept of a module for servicing conversational procedures is given. Formal description (based on the graph illustrating the functions performed in the procedure), is presented as well as the description of input module algorithm. The concept of module has been implemented in procedures programmed for CAI-system DIALOS/MLK. This system has been built in the Computer Centre of Technical University of Wrocław.

Verified by R. Wiernik

КОНЦЕПЦИЯ МОДУЛЯ ОБСЛУЖИВАНИЯ ДИАЛОГА ДЛЯ КОНВЕРСАЦИОННЫХ ПРОЦЕДУР

В работе представлена концепция построения модуля обслуживания диалога для конwersационных процедур. Приводится формальное описание структуры модуля, основанной на графе, отображающем функции, осуществляемые процедурой. Описывается алгоритм работы входного модуля. Концепция была использована в процессе разработки программных процедур Информатической системы обслуживания процесса обучения ДИАЛОС/МЛК, созданной в Вычислительном центре Вроцлавского технического университета.

Проверил Чеслав Штуковский

*Baza danych,
student*

Franciszek KL'USKA^{*)}

Base des données sur les étudiants

La formation et l'utilisation - effectivement automatisées - de la base des données exigent une certaine formalisation d'éléments et une structure convenable de leur ensemble. Nous voudrions présenter, dans cette contribution, quelques résultats de nos recherches concernant la formation de la base des données sur les étudiants.

1. FORMALISATION D'ÉLÉMENTS

Il est utile d'introduire pour chaque élément l'identificateur dit numérique ([1], [3], [5]).

Nous avons introduit, pour un étudiant, l'identificateur numérique (Ie) du type

a	:	n
---	---	---

où la partie "a" représente par ses deux places décimales une année académique, au début de laquelle l'étudiant a été immatriculé à une grande école; la partie "n" a autant de places pour qu'elles rendent possible de différencier tous les étudiants de l'année. P. ex.: le 72089 représente le 89^e étudiant qui s'est inscrit à la Faculté au cours

^{*)} Ecole Slovaque des Etudes Supérieures Techniques, Faculté d'Electrotechnique, Chaire de Cybernétique, Bratislava, Tchécoslovaquie

2. STRUCTURE DE L'ENSEMBLE D'ÉLÉMENTS

Il est possible d'avoir beaucoup de structures. Du point de vue de besoin de la capacité d'unité de mémoire de l'ordinateur digital et du besoin du temps-machine la structure optimale est, semble-t-il ([1], [4], [6]) la suivante.

L'ensemble a quelques sous-ensembles:

a) Pour chaque année de la Faculté une liste d'identificateurs numériques de branches d'études (Ib), de spécialisations (Is) et de matières d'enseignement correspondante (im) sous la forme:

$$\begin{array}{c}
 Ib_1 \\
 Is_{11} \\
 Im_{111} \quad Im_{112} \quad \dots \quad Im_{11t_{11}} \\
 Is_{12} \\
 Im_{121} \quad Im_{122} \quad \dots \quad Im_{12t_{12}} \\
 \vdots \\
 Is_{1k_1} \\
 Im_{1k_11} \quad Im_{1k_12} \quad \dots \quad Im_{1k_1t_{1k_1}} \\
 Ib_2 \\
 Is_{21} \\
 Im_{211} \quad Im_{212} \quad \dots \quad Im_{21t_{21}} \\
 \vdots \\
 Is_{2k_2} \\
 Im_{2k_21} \quad Im_{2k_22} \quad \dots \quad Im_{2k_2t_{2k_2}} \\
 \vdots \\
 Ib_J \\
 Is_{J1} \\
 Im_{J11} \quad Im_{J12} \quad \dots \quad Im_{J1t_{J1}} \\
 \vdots
 \end{array}$$

$$I_{s_{Jk_J}} \\ I_{m_{Jk_J1}} \ I_{m_{Jk_J2}} \ \dots \ I_{m_{Jk_Jt_{Jk_J}}}$$

$$t_{ju} \leq T; \quad j = 1, 2, \dots, J; \quad u = 1, 2, \dots, k_j; \quad k_j \leq K.$$

b) Les résultats d'études des étudiants de chaque année de la Faculté sous la forme:

$$\begin{array}{c} I_{e_1} I_{bs_1} \\ r_{11} \ r_{12} \ \dots \ r_{1t_1} \\ \vdots \\ I_{e_N} I_{bs_N} \\ r_{N1} \ r_{N2} \ \dots \ r_{Nt_N} \end{array}$$

I_{bs_1} = l'identificateur pour la branche d'études et la spécialisation, les-quelles suit étudiant ayant l'identificateur I_{e_1} , $i = 1, 2, \dots, N$; l'ordre des étudiants étant comme sub d) et celui de leurs résultats d'études et de matières d'enseignement pour la spécialisation correspondante I_{bs} (sub a)) est biunivoque; schématiquement:

$$\begin{array}{c} I_{m_1} \longleftrightarrow r_1 \\ I_{m_2} \longleftrightarrow r_2 \\ \vdots \\ I_{m_t} \longleftrightarrow r_t \end{array}$$

Le résultat d'étude peut comprendre non seulement la mention obtenue à l'examen, mais aussi les identificateurs numériques d'autres indices, p. ex. par quelle tentative la mention a été obtenue, le temps de l'examen, l'examineur, etc.

c) Pour chaque année les valeurs numériques des vecteurs des étudiants sous la forme:

$$\begin{array}{c} v_{11}, v_{12}, \dots, v_{1C} \\ v_{21}, v_{22}, \dots, v_{2C} \\ \vdots \\ \vdots \end{array}$$

$$V_{N1}, V_{N2} \dots, V_{NC}$$

où N = nombre des étudiants, l'ordre des vecteurs étant en accord avec d).

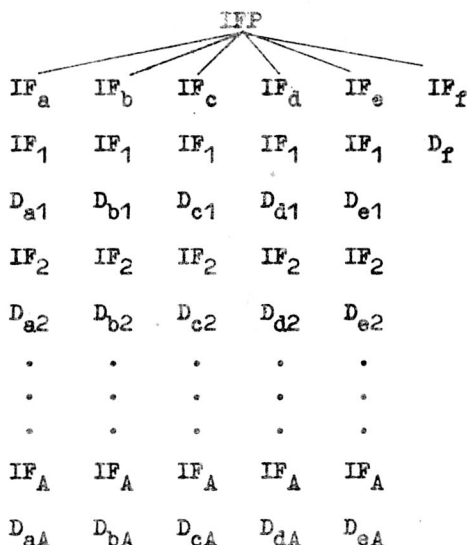
d) Noms et prénoms des étudiants ordonnés alphabétiquement par le program ([3]).

e) Dénominations des branches d'études, des spécialisations et des matières d'enseignement en accord avec a).

f) Dénominations des paramètres du vecteur p .

Les données sub d), e), f) ne s'emploient que pour la sortie des données de texte de l'ordinateur digital - pour une compréhension plus facile par les clients. Pour la communication des informations avec l'ordinateur digital on n'emploie que les identificateurs numériques.

Toute la base des données est précédée par "l'information principale fondamentale" (IFP) signifiante la Faculté correspondante, l'année académique et ainsi de suite. Chaque sous-ensemble est précédé par une information fondamentale (IF) signifiante p. ex. l'année, le nombre d'étudiants dans une année et ainsi de suite. Schématiquement:



D_{uv} = sous-ensemble des données

$u = a, b, c, e$ en accord avec a), b), c), d), e)

$v = 1, 2, \dots, A$

A = nombre des années

D_f = sous-ensemble des données correspondantes en accord avec f).

L'information principale ainsi que les autres informations fondamentales sont introduites pour rendre possible l'accès des travailleurs compétents des ensembles répertoriés dans l'ordinateur digital et elles ont aussi une certaine signification pour le programmeur.

On peut former à partir des ensembles de Facultés des ensembles réduits ainsi que des ensembles hiérarchiquement élargis selon les besoins.

RÉFÉRENCES

- [1] G v o z d j a k L., K l' u s k a F., Teoretické základy efektívne automatizovaného informačného systému o pedagogickom procese na vysokej škole. Vyskumná správa č. 14. Katedra matematických strojov EF SVST v Bratislave, 1973.
- [2] K l' u s k a F., Algoritmy a samočinné počítače. Matematika a fyzika ve škole, č. 6, február 1975, s. 401-413.
- [3] K l' u s k a F., Možnosti abecedného usporiadania textových informácií na samočinnom počítači. INORGA, č.1, 1970, s. 31-34.
- [4] K l' u s k a F., Possibilities for Automation of University Activities, Wiss. Z. Techn. Univers. Dresden, 23 (1974), H. 5, 1065-1067.
- [5] K l' u s k a F., Samočinné počítače pri riadení a správe vysokých škôl. Vysoká škôla č. 7, 1973/74, s. 300-311.
- [6] K l' u s k a F., The Use of Computers in Assessing the Teaching Process. Zb. ČSAV "Educational Cybernetics", 1973, s. 155-158.

Praca wpłynęła do Redakcji 1.12.1978 r.
Po poprawieniu 19.02.1980 r.

BAZA DANYCH ZAWIERAJĄCA INFORMACJE O STUDENTACH

Artykuł przedstawia strukturę zbioru informacji o studentach. Struktura ta wydaje się optymalna pod względem wielkości zajętej pamięci i czasu pracy maszyny cyfrowej.

БАЗА ДАННЫХ, СОДЕРЖАЩАЯ ИНФОРМАЦИЮ О СТУДЕНТАХ

В статье представлена структура базы данных, содержащей информацию о студентах. По занятости памяти и времени работы ЭВМ она представляется оптимальной.

Проверил Чеслав Штуковски

Dariusz KRAJEWSKI *)

Badanie efektywności kompilatorów

Opisano metodę porównywania kompilatorów języków programowania pod względem ich przydatności do translacji różnego typu zadań. Metoda polega na badaniu efektywności wykorzystania systemu cyfrowego podczas przetwarzania określonego typu zadań, tłumaczonych przez różne translatory języka. Zdefiniowane współczynniki i miary pozwalają ocenić przydatność danego translatora do tłumaczenia określonych typów zadań. Metodę wykorzystano do przebadania kompilatorów języka Fortran pod kątem ich użyteczności w przetwarzaniu programów studenckich.

1. WPROWADZENIE

Zadania użytkownika, przetwarzane przez system cyfrowy, odznaczają się pewnymi cechami, jak np.: czas przetwarzania, zajętość pamięci, liczba transmisji między pamięcią operacyjną i urządzeniami zewnętrznymi, charakter obliczeń itp. W zależności od tych cech, zadania można pogrupować na odpowiednie typy.

Istotną sprawą jest dobór odpowiedniego kompilatora do tłumaczenia zadania danego typu. Podczas przetwarzania zadania etap tłumaczenia może być ważny dla niektórych typów zadań ze względu na takie parametry, jak: szybkość tłumaczenia programu, wielkość i postać modułu wynikowego itp.

*) Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

Przedstawiona w dalszej części analiza efektywności systemu odnosi się do pracy systemu w reżimie jednoprogramowym.

Zdefiniowano współczynniki i miary określające efektywność wykorzystania systemu dla pojedynczego zadania i wsadu. Wielkości występujące w definiowanych wzorach w większości są funkcjami czasu.

Metody pozwalające w sposób programowy oszacować czasy wykonania pojedynczych instrukcji programu oraz średnie czasy przetworzenia zadania i wsadu opisano w pracy [3].

Pierwsza metoda polegała na cyklicznym wykonywaniu w programie badanej instrukcji. Następnie porównano czasy wykonania podobnych programów, różniących się jedynie brakiem badanej instrukcji w jednym programie i występowaniem jej w drugim. Dokładność szacowania czasu wykonania instrukcji zależała od ilości cykli. Czasy były odczytywane z zegara programowego maszyn.

Druga metoda zastosowana do pomiarów wykorzystywała podprogramy biblioteczne, odczytujące stan zegara maszynowego. Zasada pomiaru czasu polegała na wielokrotnym wykonaniu w zadanym czasie wybranych instrukcji i zliczaniu powtórzeń cyklu. Podany przykład fragmentu programu w języku Fortran ilustruje tę metodę

```

- - -
- - -
L,K=0
- - -
CALL ITIME (J)
10  K=K+1
20  CALL ITIME (N)
30  IF (J+T.NE.N) GOTO10
C   ZMIENNA T OKRESLA CZAS POWTARZANIA W CYKLU INSTRUKCJI
C   Z ETYKIETAMI 10, 20, 30
40  S=A+B
C   POWYZSZA INSTRUKCJA JEST MIERZONA
50  L=L+1
60  CALL ITIME (M)
70  IF (N+T.NE.M) GOTO40
    Z=T/L-T/K
C   ZMIENNA Z OKRESLA
C   CZAS WYKONANIA INSTRUKCJI Z ETYKIETA 40

```

Czas odczytywany przez podprogram ITIME jest wyrażony w sekundach z maksymalnym błędem bezwzględnym, równym okresowi aktualizacji zegara.

2. ANALIZA JAKOŚCIOWA KOMPILATORÓW

Zdefiniowane i wyprowadzone niżej współczynniki i miary pozwalają porównywać badane kompilatory pod względem ich zastosowania do translacji różnego typu zadań.

Wprowadźmy dwa współczynniki W_K i W_W , które określają odpowiednio stosunek czasu przebiegu translatora lub programu użytkownika do czasu przetworzenia zadania:

$$W_K = \frac{T_K}{T_P}, \quad W_W = \frac{T_W}{T_P}, \quad (1)$$

gdzie T_P - średni czas przetworzenia zadania,
 T_K - średni czas przebiegu translatora w zadaniu,
 T_W - średni czas przebiegu programu użytkownika w zadaniu.
 Niech T_{PW} oznacza średni czas przetworzenia wsadu. Wówczas

$$T_{PW} = T_{KW} + T_{LW} + T_{WW} + T_S + T_O,$$

gdzie T_{KW} - średni czas przebiegu translatora dla wsadu,
 T_{WW} - średni czas przebiegu programów użytkowych dla wsadu,
 T_{LW} - średni czas konsolidacji (przebiegu programu łączącego),
 T_O - średni czas wprowadzenia komend operatora, niezbędnych podczas przetworzenia wsadu.

Czasy T_{KW} , T_{WW} , T_{LW} wyznacza się w następujący sposób:

$$\begin{aligned} T_{KW} &= \frac{1}{p} \sum_{i=1}^p \sum_{j=1}^{n_1} T_{Kij}, \\ T_{WW} &= \frac{1}{p} \sum_{i=1}^p \sum_{j=1}^{n_1} T_{Wij}, \\ T_{LW} &= \frac{1}{p} \sum_{i=1}^p \sum_{j=1}^{n_1} T_{Lij}, \end{aligned} \quad (2)$$

gdzie T_{Kij} , T_{Lij} i T_{Wij} - odpowiednio czasy trwania kroku kompilacji, konsolidacji i wykonania j -tego zadania w i -tym wsadzie,

n_i - liczba zadań w i -tym wsadzie,

p - liczba wsadów.

Przez analogię do (1) wzory dla wsadu zadań będą następujące:

$$W_{KW} = \frac{T_{KW}}{T_{FW}}, \quad W_{WW} = \frac{T_{WW}}{T_{FW}}. \quad (3)$$

Drugą wprowadzoną miarą, pozwalającą oszacować ilościowo zastosowanie kompilatora do translacji pojedynczych programów, jest efektywna szybkość przetwarzania przez system określonego typu zadań.

Niech E_S oznacza efektywną szybkość przetwarzania danego typu zadania w systemie cyfrowym. Dodatkowo założymy, że niezawodność systemu równa jest jedności. Wówczas E_S wyrazić można wzorem:

$$E_S = E \cdot K_{UZ}, \quad (4)$$

gdzie E - średnia szybkość systemu,

K_{UZ} - współczynnik zmniejszający E_S z powodu występowania w zadaniu operacji wejścia-wyjścia.

Średnią szybkość systemu E uzależnić można od średniego czasu wykonania instrukcji w zadaniu T_{SR}

$$E = \frac{1}{T_{SR}} = \frac{\sum_{i=1}^m F_i}{\sum_{i=1}^m F_i T_i}, \quad (5)$$

gdzie F_i - częstość występowania i -tej operacji w zadaniu,

T_i - czas trwania i -tej operacji,

m - liczba różnych operacji w zadaniu.

Wartość współczynnika K_{UZ} w przybliżeniu wyznaczyć można jako stosunek różnicy średniego czasu wykonania zadania T_W i czasu wykonania operacji wejścia-wyjścia $T_{WE/WY}$ do średniego czasu wykonania zadania, tj.

$$K_{UZ} \approx \frac{T_W - T_{WE/WY}}{T_W}. \quad (6)$$

Wartość $T_{WE/WY}$ jest zależna od rodzaju operacji wejścia-wyjścia oraz od liczby występowania tych operacji w programie. Na przykład dla instrukcji WRITE w języku Fortran inna będzie wartość $T_{WE/WY}$ w razie

przesyłania rekordów niezredagowanych z pamięci operacyjnej do pamięci dyskowej, a inna w razie przesłania rekordów zredagowanych na drukarkę wierszową. W razie wielokrotnego występowania różnego typu operacji wejścia-wyjścia czas $T_{WE/WY}$ będzie wyrażać się jako suma czasów T_{WE/WY_j}

$$T_{WE/WY} = \sum_{j=1}^l k_j T_{WE/WY_j} \quad (7)$$

gdzie l - liczba typów (różnych rodzajów) operacji wej-wyj w programie,

k_j - liczba określająca ile razy j -tego typu (rodzaju) operacja wej-wyj występuje w zadaniu,

T_{WE/WY_j} - średni czas wykonania j -tego typu (rodzaju) operacji wej-wyj.

Wstawiając wartości (5), (6), (7) do wzoru (4) otrzymamy ostateczny wzór efektywnej szybkości systemu E_S

$$E_S = \frac{\sum_{i=1}^m F_i}{\sum_{i=1}^m F_i T_i} \left(1 - \frac{\sum_{j=1}^l k_j T_{WE/WY_j}}{T_W} \right). \quad (8)$$

Dotychczasowe rozważania dotyczyły idealnego systemu cyfrowego, tj. przy założeniu, że niezawodność takiego systemu była równa jedności. W dalszej części przyjęto model niezawodnościowy systemu z odnową o strukturze szeregowej i następujących założeniach:

- procesy awarii w systemie są niezależne,
- proces awarii i odnowy systemu jest opisany procesem Poissona (urodzin i śmierci),
- jeżeli podczas przetwarzania zadania wystąpi uszkodzenie systemu, to po zakończeniu procesu odnowy zadanie będzie przetwarzane od początku.

Jeżeli uwzględni się powyższe założenia, to efektywna szybkość systemu E_S zmniejszy się. Zmianę tę charakteryzować będą współczynniki wynikające z zawodności urządzeń zewnętrznych $\eta_{WE/WY}$ i jednostki centralnej η_{JC}

$$E_{SR} = \eta_{WE/WY} \cdot \eta_{JC} \cdot E_S. \quad (9)$$

Wartości współczynników $\eta_{WE/WY}$ i η_{JC} określić można w przybliżeniu ze wzorów:

$$\eta_{WE/WY} = P_{WE/WY}(t) \cdot P_{WE/WY}(T_{p \max}), \quad (10)$$

$$\eta_{JC} = P_{JC}(t) \cdot P_{JC}(T_{p \max}),$$

$P(t)$ jest to prawdopodobieństwo poprawnej pracy urządzenia w chwili t pod warunkiem, że pracowało ono poprawnie w chwili $t = 0$. Prawdopodobieństwo $P(t)$ jest nazywane funkcją gotowości systemu. W praktyce przy podanych założeniach określa się wartość graniczną tej funkcji, nazywaną współczynnikiem gotowości (zob. np. prace [2], [4]). Prawdopodobieństwo poprawnej pracy urządzenia przez czas $T_{p \max}$ określa $P(T_{p \max})$. Prawdopodobieństwo to jest nazywane funkcją niezawodności systemu. $T_{p \max}$ jest maksymalnym czasem potrzebnym do przetworzenia zadania określonego typu. Zastępując więc prawdopodobieństwa $P_{WE/WY}(t)$ i $P_{JC}(t)$ ich wartościami granicznymi $A_{WE/WY}$ i A_{JC} , otrzymamy:

$$E_{SR} = E_S \cdot A_{WE/WY} \cdot P_{WE/WY}(T_{p \max}) \cdot A_{JC} \cdot P_{JC}(T_{p \max}).$$

Oznaczamy:

$$A_S = A_{WE/WY} \cdot A_{JC}, \quad P_S(T_{p \max}) = P_{WE/WY}(T_{p \max}) \cdot P_{JC}(T_{p \max}),$$

wówczas wzór efektywnej szybkości systemu przyjmuje ostateczną postać:

$$E_{SR} = E_S \cdot A_S \cdot P_S(T_{p \max}). \quad (11)$$

Przyjmując za miarę efektywności systemu stosunek efektywnej szybkości systemu $E_{SR}(P(T_{p \max}) < 1)$ do efektywnej szybkości $E_S(P(T_{p \max}) = 1)$, mamy:

$$EF = \frac{E_{SR}}{E_S} = A_S \cdot P_S(T_{p \max}). \quad (12)$$

Miara ta pozwala porównywać efektywność tego samego systemu cyfrowego, z stosowaniem różnych translatorów, do tłumaczenia określonego zadania.

Podobnie jak dla pojedynczych zadań, można zdefiniować trzy wielkości, tj. E_{SW} , E_{SRW} , określające efektywne szybkości systemu E_{SW} dla $P(T_{p \max W}) = 1$, E_{SRW} dla $P(T_{p \max W}) < 1$ oraz EF_W , określającą efektywność systemu podczas przetwarzania wsadu zadań.

Dla pierwszej z tych wielkości mamy:

$$E_{SW} = E_W \cdot K_{UZW}, \quad (13)$$

gdzie E_W - średnia szybkość systemu dla wsadu,

K_{UZW} - współczynnik zmniejszający E_{SW} z powodu występowania w programach operacji we-wy,

przy czym

$$E_W = \frac{1}{T_{SRW}} = \frac{\sum_{j=1}^z \sum_{i=1}^{m_j} F_{ij}}{\sum_{j=1}^z \sum_{i=1}^{m_j} T_{ij} F_{ij} + \sum_{j=1}^z (T_{sj} + T_{oj})}, \quad (14)$$

$$K_{UZW} = \frac{T_{WW} - T_{WE/WY}}{T_{WW}} = 1 - \frac{1}{T_{WW}} \sum_{j=1}^z \sum_{i=1}^{l_j} k_{ij} T_{WE/WY ij} \quad (15)$$

Użyte symbole we wzorach (14) i (15) oznaczają:

- T_{SRW} - średni czas trwania operacji we wsadzie,
- T_{Sj} - czas wykonania programów systemowych, koniecznych do realizacji j-tego zadania,
- T_{oj} - czas wprowadzania komend z monitora (przez operatora), niezbędnych do realizacji j-tego zadania,
- T_{ij} - czas trwania i-tej operacji w j-tym zadaniu,
- F_{ij} - częstość występowania i-tej operacji w j-tym zadaniu,
- m_j - liczbę operacji w j-tym zadaniu,
- z - liczbę zadań we wsadzie,
- T_{WW} - średni czas przebiegu programów użytkowych dla wsadu,
- $T_{WE/WY}$ - średni czas wykonania operacji we-wy dla wsadu,
- k_{ij} - liczbę określającą częstość występowania i-tego typu operacji we-wy w j-tym zadaniu,
- $T_{WE/WY ij}$ - średni czas wykonania j-tego rodzaju operacji we-wy w i-tym zadaniu,
- l_j - liczbę rodzajów operacji we-wy w j-tym zadaniu.

Ponieważ czas trwania i-tego rodzaju (i-tej) operacji jest taki sam w każdym zadaniu, więc $T_{ij} = T_i$, a $T_{WE/WY ij} = T_{WE/WY i}$. Wzory (14) i (15) przyjmą zatem postać:

$$E_W = \frac{\sum_{j=1}^J \sum_{i=1}^{M_j} E_{ij}}{\sum_{j=1}^J \left(\sum_{i=1}^{M_j} T_i E_{ij} + T_{Sj} + T_{Oj} \right)}, \quad (16)$$

$$K_{UZW} = 1 - \frac{1}{T_{WW}} \sum_{j=1}^J \sum_{i=1}^{M_j} k_{ij} T_{WE/WY} \quad (17)$$

Bezpośrednie korzystanie z zależności (16) jest dość kłopotliwe, dlatego wprowadzono pewne uproszczenia, które umożliwiają w prosty sposób obliczenie wielkości E_W .

Niech M oznacza liczbę wszystkich operacji we wsadzie, T'_{WW} - czas zajętości jednostki centralnej na czas wykonania programów, T_{SW} - sumę czasów T_O i T_S dla wsadu. Wówczas wzór (16) można napisać w postaci:

$$E_W = \frac{M}{T'_{WW} + T_{SW}} = \frac{\frac{M}{T'_{WW}}}{1 + \frac{T_{SW}}{T'_{WW}}}.$$

Dla tej samej klasy zadań stosunek M i T'_{WW} oznaczać będzie w przybliżeniu średnią szybkość systemu E (dla pojedynczego zadania p. (5)). Ostateczna postać wzoru (16) będzie następująca:

$$E_W = \frac{E}{1 + T_{SW}/T'_{WW}}. \quad (18)$$

Z zależności tej widać bezpośrednio jaki wpływ na wartość E_W ma czas T_{SW} . Jeżeli średnia szybkość systemu dla wsadu zadań zmaleje o ponad 50% w porównaniu z tą samą szybkością dla pojedynczego zadania, oznaczać to będzie, że czas przetwarzania programów użytkownika jest krótszy od czasu wykonania programów systemowych i czasu traconego na wykonanie czynności operatorskich, niezbędnych do realizacji zadań.

Efektywna szybkość systemu dla wsadu E_{SRW} jest wyrażona zależnością:

$$E_{SRW} = E_{SW} \cdot A_S \cdot P_S (T_{p \max W}). \quad (19)$$

$P_S(T_{p \max W})$ jest to prawdopodobieństwo poprawnej pracy systemu przez maksymalny czas przetwarzania wsadu. Jeżeli wsad składać się będzie z z zadań oraz czas przetwarzania i -tego zadania oznaczyć przez T_{P_i} , to efektywną szybkość systemu E_{SRW} obliczyć można z następującej zależności:

$$E_{SRW} = E_{SW} \cdot A_S \cdot P_S(T_S) \cdot P_S(T_O) \cdot \prod_{i=1}^z P_S(T_{P_i}); \quad (20)$$

$P_S(T_S)$, $P_S(T_O)$ i $P_S(T_{P_i})$ określają odpowiednio prawdopodobieństwa poprawnej pracy systemu w czasie T_S , T_O i T_{P_i} .

Miarą efektywności systemu dla wsadu będzie stosunek efektywnych szybkości E_{SRW} i E_{SW} .

$$EF_W = \frac{E_{SRW}}{E_{SW}} = A_S \cdot P_S(T_{p \max W}). \quad (21)$$

Efektywność systemu EF_W wyrazić można również przez efektywność EF_i dla każdego zadania we wsadzie (patrz zależność (12)). Korzystając z zależności (12) i (20), otrzymamy:

$$EF_i = A_S \cdot P_S(T_{P_i})$$

$$\begin{aligned} EF_W &= A_S \cdot P_S(T_S + T_O) \cdot \prod_{i=1}^z P_S(T_{P_i}) = \\ &= A_S^{(1-z)} P_S(T_S + T_O) \cdot \prod_{i=1}^z P_S(T_{P_i}) \cdot A_S = \\ &= A_S^{(1-z)} \cdot P(T_S + T_O) \cdot \prod_{i=1}^z EF_i. \end{aligned} \quad (22)$$

Prawdopodobieństwo $P(T)$ poprawnej pracy systemu w przedziale czasu T sprowadza się do postaci wykładniczego prawa niezawodności przy założeniu, że intensywność uszkodzeń systemu jest stała, $\lambda_S(t) = \lambda_S = \text{const}([2], [4])$.

Uwzględniając zawodność systemu i wykorzystując wykładniczą funkcję niezawodności, zależności (9), (12), (19) i (21) przyjmą ostatecznie postać:

$$E_{SR} = E_S \cdot A_S \cdot \exp(-\lambda_S T_{p \max}),$$

$$EF = A_S \exp(-\lambda_S T_{p \max}),$$

$$E_{SRW} = E_{SW} \cdot A_S \cdot \exp\{-\lambda_S(T_S + T_0)\} \prod_{i=1}^Z \exp(-\lambda_S T_{P_i})$$

$$EF_W = A_S \cdot \exp\left\{-\lambda_S \left(\sum_{i=1}^Z T_{P_i} + T_S + T_0\right)\right\}.$$

3. PODSUMOWANIE

Przedstawiona analiza kompilatorów nie ujmuje wszystkich możliwych przypadków przetwarzania zadań w systemie. Nie rozpatrywano modelu systemu wieloprogramowego. Nie uwzględniono również przypadków przetwarzania zadania, podczas którego następuje uszkodzenie systemu. Przyjęto najprostsze rozwiązanie, tj. z chwilą pojawienia się uszkodzenia chwilowego (przekłamania) zadanie jest przetwarzane od początku.

Jednostką miar efektywnych szybkości systemu jest liczba instrukcji na jednostkę czasu. Celowe byłoby wprowadzenie miary efektywności systemu, której jednostką będzie liczba zadań na jednostkę czasu. Wielkość tę można wyznaczyć z zależności

$$EF = \lim_{t \rightarrow \infty} \frac{1}{t} \gamma_Z(t),$$

gdzie $\gamma_Z(t)$ jest średnią liczbą zadań zrealizowanych w przedziale czasu $(0, t)$ (zob. [1]).

Opisane metody badania kompilatorów wykorzystano w pracy [3] do oceny porównawczej kompilatorów języka Fortran. Ocenę przeprowadzono pod kątem zastosowania tych kompilatorów w dydaktyce.

Do testowania kompilatorów wykorzystano programy napisane przez studentów trzeciego roku w ramach przedmiotu Zaawansowane metody pro-

gramowania. Dostępne kompilatory języka Fortran 1900, tj. XFAT, XFAE, XFAM i SOFOR4B, były testowane na maszynie Odra 1305, kompilator IEYFORT natomiast na maszynie R-32 Jednolitego Systemu. Testowanie dotyczyło między innymi szybkości przetwarzania programów i szybkości przetwarzania wsadu zadań.

T a b e l a 1

Miary systemu cyfrowego określone dla pojedynczych zadań

Miara systemu (1/ms)	Odra 1300				R-32
	XFAT	XFAE	XFAM	SOFOR4B	IEYFORT ^{*)}
E	1.96	1.96	2.00	1.38	4.34
E _S	1.84	1.84	1.86	1.36	3.90
E _{SR}	1.64	1.64	1.66	1.22	-
EF ^{*)}	0.8966	0.8966	0.8966	0.8966	-

^{*)} Dla kompilatora IEYFORT nie obliczono wartości E_{SR} i EF, ponieważ nie dysponowano odpowiednimi wynikami pomiarów, pozwalającymi obliczyć współczynnik gotowości systemu A_S.

^{*)} Wielkość bezwymiarowa.

T a b e l a 2

Miary systemu cyfrowego określone dla wsadu zadań

Nazwa kompilatora w systemie Miara systemu (1/ms)	Kompilator XFAT egzekutor	SOFOR egzekutor	Kompilator IEYFORT OS/JS ^{*)}
E _W	0.133	1.588	0.213
E _{SW}	0.123	1.492	0.202
E _{SRW}	0.111	1.338	-
EF _W ^{*)}	0.8947	0.8965	-

^{*)} Dla kompilatora IEYFORT nie obliczono wartości E_{SRW} i EF_W, ponieważ nie dysponowano odpowiednimi wynikami pomiarów pozwalającymi obliczyć intensywność uszkodzeń systemu λ_S i współczynnik gotowości systemu A_S.

^{*)} Wielkość bezwymiarowa.

Na podstawie otrzymanych wyników (tab. 1 i 2) można wytypować kompilator, który najbardziej będzie się nadawał do tłumaczenia zadań studenckich.

Duże ilości niewielkich programów najefektywniej są przetwarzane pod kontrolą systemu Sofor. Efektywna szybkość przetwarzania pojedynczego zadania w systemie Sofor jest 1,4 razy mniejsza od tej samej wielkości dla egzekutora i 3 razy mniejsza niż w systemie OS/JS. Efektywność systemu cyfrowego dla zadania o krótkim czasie przetwarzania, zarówno dla egzekutora jak i systemu Sofor, jest taka sama i równa w przybliżeniu współczynnikowi gotowości A_S . Związane jest to ze średnim czasem międzyawaryjnym, nieporównywalnie dłuższym od czasu wykonania zadania.

Za stosowaniem systemu Sofor do grupowej kompilacji i wykonania krótkich programów przemawiają wyniki uzyskane dla wsadu zadań. Mimo, że efektywna szybkość systemu E_{SRW} zmalała o 20%, jest ona jednak 12 razy większa od E_{SRW} dla egzekutora i 7 razy większa od E_{SW} dla systemu OS/JS. Efektywne szybkości E_{SW} i E_{SRW} dla systemu egzekutor zmalały 14-krotnie w stosunku do odpowiednich wielkości E_S i E_{SR} , a dla systemu OS/JS prawie 20-krotnie. Oznacza to, że czasy wykonania programów systemowych i ewentualnie komend operatorskich, związane z przetwarzaniem wsadu, są wielokrotnie dłuższe niż czasy wykonania programów użytkowych.

Jedynie w systemie Sofor czasy te są krótsze niż czasy wykonania programów użytkownika. Większa wartość współczynnika efektywności systemu cyfrowego EF_W dla systemu Sofor wiąże się z krótszym czasem przetwarzania wsadu.

LITERATURA

- [1] B e r l o w R. E., P r o s c h a n F., Mathematical theory of reliability, New York 1970.
- [2] K o p o c i ń s k i B., Zarys teorii odnowy i niezawodności, Warszawa 1973, PWN.
- [3] K r a j e w s k i D., Metody porównywania kompilatorów z punktu widzenia ich użyteczności w dydaktyce, Komunikaty Centrum Obliczeniowego Politechniki Wrocławskiej nr 46, Wrocław 1978.
- [4] Z a m o j s k i W., Teoria i technika niezawodności, Wrocław 1976, Wyd. PWr.

Praca wpłynęła do Redakcji 6.03.1979 r.
Po poprawieniu 19.02.1980 r.

COMPILER EFFICIENCY TESTING

Methods for comparing different compilers are described. These methods have been used for the analysis of five chosen Fortran compilers. The results obtained are presented and discussed from the point of view of their utility in the teaching process. Coefficients and measures of system effectiveness for different types of tasks are defined.

Verified by R. Wiernik

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ КОМПИЛЯТОРОВ

В статье представлен метод сравнения компиляторов языков программирования с точки зрения их полезности для перевода различных типов заданий. Метод состоит в исследовании эффективности использования вычислительной системы во время разработки определённого типа задач, переводимых с использованием различных трансляторов. Введение коэффициенты и меры дают возможность оценки полезности конкретного транслятора для перевода определенных типов заданий. Метод был использован для исследования компиляторов языка Фортран с точки зрения их полезности в обработке студенческих программ.

Проверил Чеслав Штуковский

*Kompilator konwersacyjny,
język programowania,
analiza gramatyczna*

Ludwik KUŹNIARZ^{*)}

Projekt konwersacyjnego kompilatora języka Fortran

Przedstawiono projekt konwersacyjnego kompilatora języka Fortran, opracowanego w celu zwiększenia efektywności nauczania programowania prowadzonego w pracowni wielodostępu. Opisano sposób działania kompilatora, zasady budowy jego składowych oraz ogólną strukturę danych wykorzystywanych przez te składowe.

1. WSTĘP

Jednym z elementów nauczania informatyki w Politechnice Wrocławskiej są zajęcia prowadzone w pracowni wielodostępu. Studenci podczas tych zajęć mają nauczyć się pracować pod kontrolą systemu Minicp, wprowadzać informacje do zbiorów i aktualizować je oraz uruchomić kilka prostych programów. Zajęcia te mają za zadanie zapoznać studentów z pracą pod kontrolą systemu wielodostępnego. Nauka programowania albo nie jest na nich prowadzona wcale, albo jest prowadzona w bardzo wąskim zakresie. Przyczyną tego stanu jest między innymi fakt, iż nie ma dostępnych wygodnych konwersacyjnych kompilatorów języków programowania (wyjątkiem jest interpreter języka Jean, lecz język ten nie jest przeznaczony do obliczeń naukowo-technicznych i nie jest językiem obecnie powszechnie stosowanym). Za pomocą obecnie dostępnych kompilatorów języka Fortran czy Algol można tłumaczyć jedynie kompletne pro-

^{*)} Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

gramy, co wymaga od uczących się wcześniejszego opanowania często dość obszernego materiału. Jeśli ponadto program jest kilkakrotnie poprawiany, co u początkujących programistów zdarza się często, to taki sposób kompilacji jest mało efektywny zarówno ze względu na uczącego się, jak i system.

W celu zwiększenia efektywności zajęć prowadzonych w laboratorium podjęto liczne działania, określone jako projekt LABOR [2]. Jednym z zadań realizowanych w jego ramach jest opracowanie konwersacyjnego kompilatora języka Fortran oraz programu interpretującego kod wynikowy. Idea pierwszego z nich zostanie przedstawiona w niniejszej pracy.

W pracy są używane dwa terminy: wiersz programu źródłowego oraz elementarny fragment programu. Znaczenie tych terminów wyjaśnione jest poniżej.

Wiersz programu źródłowego jest to informacja, która potencjalnie może być dyrektywą lub instrukcją w języku. Jest to ciąg fizycznych (tzn. zapisanych w podzbiorze, na kartach lub czytanych z konsoli) wierszy programu, połączonych znakiem kontynuacji.

Elementarny fragment programu jest to taki ciąg wierszy programu źródłowego, który po uzupełnieniu brakującymi dyrektywami i ewentualnie instrukcjami tworzy poprawny formalnie program w języku Fortran. Mówiąc poglądowo, jest to dowolny z fragmentów programu, jaki można wyciąć jednym lub dwoma poziomymi cięciami z kompletnego, poprawnego formalnie programu.

Jeśli np. przyjmiemy, że programem jest segment główny w języku Fortran, to formalnie syntaktykę elementarnego fragmentu programu można zdefiniować w następujący sposób:

```

<nagłówek segmentu> ::= MASTER | MASTER <nazwa>
<początek segmentu> ::= <nagłówek segmentu> | <deklaracja> |
                        <początek segmentu> <deklaracja> |
                        <puste>
<koniec segmentu> ::= END | <instrukcja> | <instrukcja>
                        <koniec segmentu> | <puste>
<elementarny fragment programu> ::= <początek segmentu>
                                     <koniec segmentu>

```

Przyjęte tutaj zostało, iż definicja klas: <nazwa>, <deklaracja>, <instrukcja>, <puste> jest oczywista.

Ogólne zasady uzupełniania fragmentów programu do kompletnego programu są następujące:

1. Dopisuje się deklaracje tablic ze standardowymi rozmiarami i liczbą wskaźników, zgołą z ich pierwszym użyciem,
2. Dopisuje się standardowy wiersz MASTER, jeśli fragment rozpoczyna się od deklaracji lub instrukcji,

3. Wszystkie nieprzyporządkowane numery programowe urządzeń wejścia przypisuje się do CRØ, a wyjścia do LPØ,

4. Jeśli segment nie został zakończony, to dopisuje się wiersz PAUSE lub RETURN, w zależności od tego czy był to segment główny, czy też podprogram lub funkcja, oraz END i FINISH,

5. Jeśli został użyty niezdefiniowany podprogram, to dopisuje się definicję tego segmentu, zgodną ze sposobem pierwszego odwołania się do niego i z treścią "nie nie rób".

2. PRZEBIEG PRACY KOMPILATORA

Na rysunku 1 pokazano schemat działania kompilatora z punktu widzenia użytkownika. Schemat ten wymaga kilku wyjaśnień i uzupełnień.

Nazwy poprzedzone znakiem * oznaczają pewne zmienne, których wartości ustala użytkownik przed rozpoczęciem właściwego procesu tłumaczenia.

Wartości zmiennych * WEJSCIE, * PROGRAM WYNIKOWY, * RODZAJ SYGNALIZACJI BŁĘDU określają odpowiednio skład będączytywane wiersze programu wynikowego, czy ma być wytwarzany program wynikowy oraz w jaki sposób mają być sygnalizowane błędy wykryte podczas tłumaczenia.

Wielkimi literami napisano pytania, na które odpowiada sam kompilator, a małymi te, na które odpowiada użytkownik (sam proces zadania pytania i otrzymania odpowiedzi nie został zaznaczony).

W bloku decyzyjnym zawierającym zmienną * RODZAJ SYGNALIZACJI BŁĘDU nie opisano wyjść. Opisują je dokładnie bloki funkcyjne, związane z odpowiednimi wyjściami.

Przez pojęcie ANALIZA rozumie się tutaj sprawdzenie poprawności syntaktycznej i semantycznej. Sposób jej przeprowadzenia zostanie dokładniej opisany w rozdz. 3.

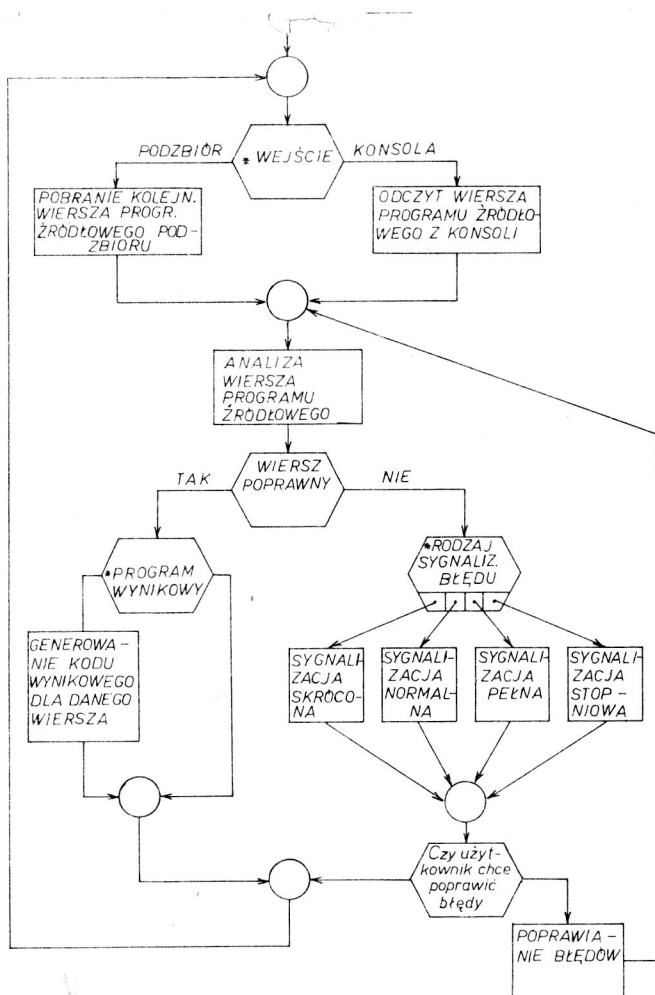
Znaczenie rodzajów sygnalizacji błędów jest następujące:

1. SYGNALIZACJA SKRÓCONA - podaje jedynie informację o tym, że w odpowiednim wierszu wystąpił błąd.

2. SYGNALIZACJA NORMALNA - podaje dodatkowo rodzaj błędu, jaki wystąpił.

3. SYGNALIZACJA PEŁNA - podaje dodatkowo (do SYGNALIZACJI NORMALNEJ) miejsce, w którym został wykryty odpowiedni błąd.

4. SYGNALIZACJA STOPNIOWA - podaje sygnalizację skróconą, a następnie na żądanie użytkownika kolejną sygnalizację normalną i pełną. Użytkownik może na dowolnym poziomie zażądać zakończenia sygnalizacji i wtedy następuje przejście do następnego bloku w schemacie. Proces ten także nie został jawnie zaznaczony na schemacie.



Rys.1. Schemat działania kompilatora
Fig.1. Compiler operation scheme

Wiersze programu źródłowego są numerowane i przechowywane w specjalnej tablicy roboczej. Użytkownik może zażądać wydrukowania dotychczas przetworzonej treści programu i może poprawić dowolny wiersz programu.

Kod wynikowy jest generowany dla wszystkich poprawnych wierszy programu, niezależnie od tego czy w poprzednich wierszach były błędy czy też nie. Przyczyny przyjęcia takiego rozwiązania podane zostaną w rozdz. 4.

3. BUDOWA KOMPILATORA

Zaprojektowany kompilator jest kompilatorem jednoprzebiegowym i zgodnie z ogólnie przyjętą koncepcją takich kompilatorów [1], składa się z trzech głównych części:

1. skanera,
2. analizatora,
3. generatora.

Skaner dokonuje wstępnej analizy tekstu - tłumaczy go na postać wewnętrzną i rozpoznaje jaką instrukcję może on potencjalnie reprezentować.

Analizator sprawdza formalną poprawność wiersza programu oraz to czy analizowany ciąg wierszy tworzy elementarny fragment programu. Jest on dwupoziomowy. Poziom pierwszy (A1) sprawdza poprawność poszczególnych wierszy, a poziomy drugi sprawdza poprawność analizowanego ciągu wierszy jako całości.

Prezentacja elementów analizatora zostanie przedstawiona na podstawie konstrukcji dostępnej w języku Pascal, z tym że w wywołaniach procedur wyraźnie zostaną zaznaczone parametry wejściowe poprzedzone IN, oraz parametry wyjściowe poprzedzone OUT.

Pracę analizatora i skanera można przedstawić następująco:

SKANER IN (WIERSZ ZRODŁOWY)

OUT (NR INSTRUKCJI, WIERSZ WEWNETRZNY, KONIEC);

while not KONIEC do

begin

A1 IN (NR INSTRUKCJI, WIERSZ WEWNETRZNY)

OUT (KOD WYNIKOWY, SYMBOL);

A2 IN (SYMBOL, STAN)

OUT (STAN);

SKANER IN (WIERSZ ZRODŁOWY)

OUT (NR INSTRUKCJI, WIERSZ WEWNETRZNY, KONIEC)

end

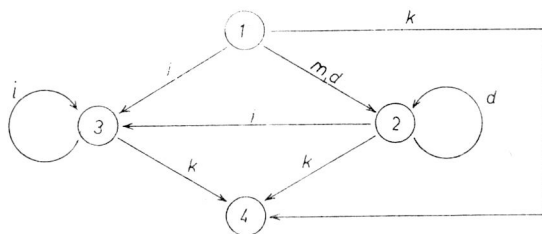
Analizator A2 jest to automat deterministyczny. Jeśli ponownie jak w rozdz. 1 ograniczymy się do segmentu głównego w języku Fortran, to można go przedstawić w postaci grafu (rys.2). Oznaczenia na tym rysunku są następujące:

m - nagłówek segmentu,

d - deklaracja,

i - instrukcja,

k - koniec programu (END lub koniec tekstu źródłowego).



Rys.2. Graf analizatora A1
Fig.2. Graph of A1 analyser

Analizator A2 działa jak następuje

```

case NR INSTRUKCJI of
  INSTRUKCJA NR 1 : A21;
  INSTRUKCJA NR 2 : A22;
  .
  .
  INSTRUKCJA NR n : A2n

```

end

gdzie $A21, \dots, A2n$ są analizatorami poszczególnych rodzajów instrukcji. Są to analizatory sterowane przez składnię. Algorytm analizy użyty we wszystkich analizatorach jest taki sam, operuje jedynie na różnych danych (gramatykach). Pozwala to w bardzo oszczędny sposób gospodarować pamięcią. Taki analizator można dowolnie również rozbudować czy aktualizować.

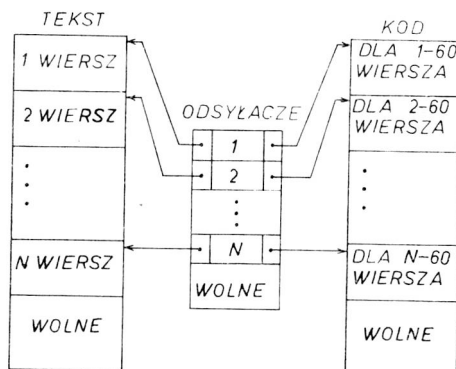
Generator tworzy ciąg wywołań podprogramów (nazywany tutaj kodem wynikowym), symulujących działanie danej instrukcji. W istocie pracuje on jednocześnie z analizatorem, a działanie jego można również opisać za pomocą przełącznika, podobnie jak to zrobiono dla analizatora A2, lecz sterowanego regułami gramatyki.

4. STRUKTURA DANYCH KOMPILATORA

Przedstawiony tu zostanie jedynie ten fragment struktury danych, który jest istotny dla objaśnienia jego własności użytkowych. Są to trzy tablice przedstawione na rysunku 3. Zawartość ich jest następująca:

TEKST - zapamiętane znakowo wiersze programu źródłowego,
KOD - zapamiętana informacja o wywołaniach podprogramów symulujących działanie instrukcji,

ODSYŁACZE - pary odsyłaczy (t_i, k_i); t_i - wskazuje na początek i -tego wiersza w tablicy TEKST, a k_i - na początek kodu dla i -tego wiersza w tablicy KOD.



Rys.3. Tablice kompilatora
Fig.3. Compiler tables

W każdej z tablic jest pewien nie wykorzystany do danego momentu pracy kompilatora obszar, do którego są przesyłane informacje napływające podczas procesu tłumaczenia.

Sposób, w jaki jest tutaj pamiętana informacja wejściowa i wynikowa jest wygodny do poprawienia programu. Usunięcie wiersza z programu wymaga jedynie aktualizacji odsyłaczy. Dołączenie lub zamiana wiersza wymaga dopisania go w nie wykorzystanym obszarze tablicy TEKST, a następnie sprawdza się czy wiersz może być przetłumaczony oddzielnie. Jeśli tak, to tłumaczy się go, odpowiednie wywołania umieszcza się w wolnym obszarze tablicy KOD i ustala odpowiednio odsyłacze, w przeciwnym przypadku tłumaczy się pewien fragment programu. Taka metoda aktualizacji opiera się na obserwacji, iż w wielu przypadkach wymiana lub dopisanie wiersza do programu w języku Fortran nie wymaga powtórnej analizy całego programu, aby taki wiersz poprawnie przetłumaczyć. Postępowanie tutaj zaprezentowane zwiększa w istotny sposób efektywność procesu kompilacji.

5. UWAGI KOŃCOWE

Kompilator, którego zarys konstrukcji został tutaj przedstawiony jest przystosowany do nauczania programowania. W prosty sposób można go modyfikować. Można mianowicie zmienić definicję pewnych instrukcji, jak również dodać nowe instrukcje. W rezultacie można otrzymać kompi-

lator nowego języka typu Fortran, tzn. języka bez struktury blokowej i o strukturze wierszowej (1 wiersz = 1 instrukcja). Językami takimi są np. Basic i Jean.

LITERATURA

- [1] G r i e s D., Compiler Construction for Digital Computers, New York 1971.
- [2] K u ź n i a r z L., P i s a r s k i J., R ý c h l i k o w s k i E., T u z i n k i e w i c z L., System KONFORT. Założenia projektowe i opis ogólny. Raport Centrum Obliczeniowego Politechniki Wrocławskiej, Nr 105, Wrocław 1977.

Praca wpłynęła do Redakcji 1.12.1978 r.
Po poprawieniu 19.02.1980 r.

A PROJECT OF A CONVERSATIONAL FORTRAN COMPILER

A project of a conversational Fortran compiler is described. The compiler was designed to improve the efficiency of using terminals for teaching the programming. The facilities the compiler provides and also its main modules and the most important data structures are presented.

Verified by R. Wiernik

ПРОЕКТ КОНВЕРСАЦИОННОГО КОМПИЛЯТОРА ФОРТРАНА

В работе представлен проект конверсационного компилятора Фортрана. Целью разработки является повышение эффективности использования терминалов в обучении программирования. Описан способ работы компилятора, принципы построения его компонентов, а также структуры данных, используемых в этих компонентах.

Проверил Чеслав Штуковский

*Zadanie kontrolne, odpowiedź
konstruowana, sprawność
funkcjonalna, komputerowe
wspomaganie dydaktyki*

Andrzej PILAWSKI^{*)}

O zagadnieniu stopnia trudności zadań kontrolnych w systemach CAI

Dla pewnej klasy zadań, stosowanych w systemach komputerowego wspomagania dydaktyki CAI, podano metodę oszacowania stopnia trudności zadania z odpowiedzią konstruowaną. Przedstawiono także metodę oceny intelektualnej sprawności funkcjonalnej studenta oraz metodę oceny dopasowania stopnia trudności zadania do średniej intelektualnej sprawności funkcjonalnej testowanych studentów. Następnie sformułowano i zweryfikowano hipotezę dotyczącą wpływu sposobu sformułowania zadania na jego stopień trudności.

1. WSTĘP

Zadania służące do kontroli wiadomości dużej liczby studentów powinny być opracowane szczególnie starannie pod względem treści. Postulat ten odnosi się również do zadań opracowanych dla systemu komputerowego wspomagania dydaktyki DIALOS/MLK¹⁾. System ten obsługuje od kilku lat sprawdziany programowania w języku Fortran, przeprowadzane dla ponad 2000 studentów rocznie.

W toku prac dotyczących zadań z języka Fortran dla systemu DIALOS/MLK pojawiło się pytanie, w jakiej mierze sam sposób zredagowania zadania wpływa na stopień jego trudności, tj. trudności jego rozwiązania. Zbadanie tego wpływu ma istotne znaczenie dla określenia, w jakim stopniu przeprowadzony sprawdzian informuje o rzeczywistych

^{*)} Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

¹⁾ Dokładniej mówiąc, dla podsystemu DIALOS/MLK w systemie DIALOS

umiejętnościach studentów z programowania. Również porównanie wyników sprawdzianów, dokonywanych w różnych grupach za pomocą różnych zadań, powinno uwzględniać ten wpływ.

Zasadniczym problemem, jaki tu wystąpił, było zdefiniowanie i oszacowanie stopnia trudności zadania z tzw. odpowiedzią konstruowaną, brak jest bowiem dostępnej literatury na ten temat.

W niniejszej pracy do określenia stopnia trudności zadania z odpowiedzią konstruowaną zaproponowano metodę opartą na znanych metodach statystycznych. Metodami statystycznymi posłużono się również w określaniu dostosowania stopnia trudności zadania do średniej intelektualnej sprawności funkcjonalnej testowanej grupy studentów oraz w badaniu wpływu sposobu zredagowania zadania na jego stopień trudności.

Przeprowadzone badania objęły zadania kontrolne z odpowiedzią konstruowaną, stosowane w systemie DIALOS/MLK, ale wnioski mają charakter ogólniejszy.

2. STOPIEŃ TRUDNOŚCI ZADANIA

Stopień trudności zadania określa się zwykle na podstawie analizy odpowiedzi udzielonych (na to zadanie) przez odpowiednio liczną grupę studentów. Ustalenie uniwersalnej metody oceny stopnia trudności zadania jest niemożliwe, ponieważ zadania testowe stosowane w systemach CAI mogą (w zależności od celu sprawdzianu) zakładać różnorodne, często nieporównywalne postacie odpowiedzi.

2.1. Klasyfikacja zadań testowych stosowanych w systemach CAI

Sposób analizowania odpowiedzi przez system CAI zależy od przyjętej przez projektanta zadania formy udzielania odpowiedzi. W klasyfikacji zadań testowych pod względem formy udzielanej odpowiedzi przyjęto następujące 4 kryteria klasyfikacji zadań testowych:

- (Z1) Przedmiot testu; kryterium to umożliwia podział na:
 - (Z1.1) testy umiejętności rozwiązywania problemów,
 - (Z1.2) testy wiadomości (zasobu wiedzy);
- (Z2) Struktura odpowiedzi - umożliwia podział na zadania, w których:
 - (Z2.1) odpowiedź można podzielić na ciąg odpowiedzi elementarnych; podczas oceniania całości odpowiedzi, odpowiedziom elementarnym przypisuje się zazwyczaj tę samą wagę, co pozwala na wystawienie oceny binarnej (ocena za odpowiedź poprawną wynosi 1, a za błędną - 0) za każdą odpowiedź elementarną; gdy odpowiedzi elementarne mają różne wagi, ocena

binarna za każdą odpowiedź mnożona jest przez jej wagę; łączna ocena za wykonanie zadania jest sumą ocen za odpowiedzi elementarne;

(Z2.2) odpowiedź stanowi niepodzielną całość, ocenianą binarnie.

(Z3) Kryterium czasu – umożliwiające podział na:

(Z3.1) zadania, w których czas odpowiedzi jest mierzony i wpływa na ocenę,

(Z3.2) zadania, dla których odpowiedzi powinny się zmieścić w zadanym czasie.

(Z4) Krotność odpowiedzi; to kryterium umożliwia podział zadań na takie, w których odpowiedź jest podawana:

(Z4.1) jednorazowo,

(Z4.2) wielokrotnie (w toku kolejnych powtórzeń w zadanych odstępach czasowych).

W zadaniach klasy (Z3.1) ocena jest funkcją czasu i może być miarą intensywności wysiłku umysłowego podczas rozwiązywania zadania. Ocena za zadania klasy (Z4.2) w każdym odcinku czasowym jest określona identycznie jak w zadaniach klasy (Z4.1); ostateczna ocena jest funkcją czasu i w zależności od przedmiotu testu może określać parametry tzw. krzywej uczenia, krzywej pamięci, krzywej zapominania itp.

Podana klasyfikacja wskazuje nauczycielowi na różnorodność form odpowiedzi i związane z tym duże możliwości projektowania zadań. Jednakże przyjęta forma odpowiedzi musi uwzględniać możliwości systemu CAI pod względem automatycznej analizy odpowiedzi. Należy dokonać jeszcze klasyfikacji systemów CAI pod względem sposobu analizy odpowiedzi. Klasyfikację tę można oprzeć na dwóch – jak się wydaje najważniejszych – kryteriach:

(S1) Kryterium przedmiotu analizy; stosując to kryterium dzielimy systemy na takie, które:

(S1.1) prowadzą bieżącą analizę toku rozumowania studenta i ewentualną korektę błędów,

(S1.2) dokonują analizy i oceny tylko końcowej postaci rozwiązania.

(S2) Kryterium języka odpowiedzi; językiem tym może być:

(S2.1) podzbiór języka naturalnego,

(S2.2) język sformalizowany.

Z porównania obydwu klasyfikacji widać, że np. zadania klasy

$(Z1.2 \cap Z2.2)$ wymagają systemu klasy $(S1.2 \cap S2.2)$, zadania klasy

$(Z1.1 \cap Z2.1)$ muszą być realizowane w bardziej skomplikowanym systemie klasy $(S1.1 \cap S2.2)$, zadania klasy $(Z1.2 \cap Z2.1)$ natomiast wymagają

najczęściej zastosowania najbardziej rozbudowanego systemu $(S1.1 \cap S2.1)$.

Do najpopularniejszych typów zadań, stosowanych w systemach CAI klasy (S2.2), można zaliczyć

- a) zadania z uzupełnianiem luk,
- b) zadania z wielowymoborem (repertuarem odpowiedzi),
- c) zadania z odpowiedzią konstruowaną.

W systemie DIALOS/MLK wykorzystuje się wszystkie trzy typy zadań.

Zadania z uzupełnianiem luk oraz zadania z wielowymoborem można zakwalifikować do klasy $(Z1.2 \cap Z2.2)$, a zadania z odpowiedzią konstruowaną do klasy $(Z1.1 \cap Z2.1)$.

Najbardziej popularną formą zadań testowych, oceniających zasób wiedzy, są zadania oparte na zasadzie wielowymoboru. Zadania tego typu są oceniane binarnie (co znacznie upraszcza analizę wyników) i w związku z tym mogą być przetwarzane w najprostszych systemach CAI, a nawet w maszynach egzaminacyjnych bez wspomagania komputerowego.

Powszechne stosowanie zadań z wielowymoborem wpłynęło na znaczny postęp w pracach teoretycznych dotyczących analizy odpowiedzi zadań klasy $(Z1.2 \cap Z2.2)$. Do najciekawszych prac z tej dziedziny można zaliczyć metrologiczną teorię egzaminatora liniowego Jadwigi Muciek (p. [8], s. 527-532) oraz fizyczną teorię pomiaru funkcji intelektualnych Zygmunta Zimnego (p. [7], s. 195-273).

2.2. Stopień trudności zadania w świetle teorii pomiarów funkcji intelektualnych

Teoria pomiarów funkcji intelektualnych Zimnego dotyczy między innymi oceny trudności zadania testowego (wykorzystywanego powszechnie w diagnostyce psychologicznej i pedagogicznej) na podstawie pośredniego pomiaru pracy umysłowej, niezbędnej do wykonania zadania. W zależności od klasy zadania, określonej na podstawie przyjętej przez siebie klasyfikacji. ([7], s. 212), Zimny proponuje różne metody obliczania intelektualnej sprawności funkcjonalnej osoby testowanej oraz obliczania stopnia trudności zadania.

Zadania z odpowiedzią konstruowaną można traktować jako zadania, w których odpowiedź jest ciągiem odpowiedzi elementarnych (patrz pkt 2.3); w przypadku równych wag będzie to zadanie I rodzaju według klasyfikacji Zimnego. Klasa $(Z2.1)$ obejmuje także zadania z odpowiedzią konstruowaną, dopuszczając przy tym różne wagi odpowiedzi elementarnych. Na tej podstawie rozszerzymy metodę Zimnego obliczania stopnia trudności zadań na zadania klasy $(Z2.1)$.

Wyniki badań, cytowane przez Zimnego, potwierdzają słuszność założenia, że miarą efektywnej pracy umysłowej L_e , włożonej w rozwiąza-

nie zadania I rodzaju jest liczba P poprawnie rozwiązanych elementów, a miarą umysłowej pracy nieefektywnej L_{ne} jest liczba B błędnie rozwiązanych elementów.

Tak więc ocena postaci $P/(P+B)$ za zadanie I rodzaju jest miarą intelektualnej sprawności funkcjonalnej osoby testowanej [7], s.209, 218

$$S = \frac{P}{P+B}.$$

Indywidualna trudność D zadania jest funkcją sprawności funkcjonalnej; dla zadań I rodzaju można przyjąć, że $D = 1 - S = B/(P+B)$.

Interindywidualna trudność zadania^{*}, zwana najczęściej stopniem trudności zadania, jest średnią arytmetyczną rozkładu indywidualnych trudności w grupie testowanych studentów.

2.3. Metoda oszacowania stopnia trudności zadania z odpowiedzią konstruowaną oraz sprawności funkcjonalnej studenta

Zajmiemy się tu zadaniami z odpowiedzią konstruowaną, w których rozwiązaniem układanym przez studenta jest k -elementowy ciąg $\bar{b} = (b, c, d \dots)$ tzw. bloków, należących do znanego studentowi zbioru n -elementowego B , $n \geq k$. Zbiór B może być np. zbiorem wyrazów, zbiorem liczb lub zbiorem zdań. Parę uporządkowaną $q = (\text{blok } b \in B, \text{ numer bloku } b \text{ w rozwiązaniu } \bar{b})$ możemy traktować jako odpowiedź elementarną i tym samym odpowiedź studenta - jako ciąg $(q_1, \dots, q_k)$ odpowiedzi elementarnych. W tym ujęciu zadanie z odpowiedzią konstruowaną staje się zadaniem klasy (Z2.1).

2.3.1. Przypisywanie wag odpowiedziom elementarnym

Przypisywanie wag odpowiedziom elementarnym jest łatwe wówczas, gdy możemy przypisać je wszystkim blokom zbioru B . Ale na ogół tak nie jest; waga odpowiedzi elementarnej zależy nie tylko od merytorycznej treści bloku, ale i od jego funkcji w rozwiązaniu (od numeru w ciągu $\bar{b}$):

W zadaniach kontrolnych systemu DIALOS/MLK rozwiązanie zadania (czyli ciąg $\bar{b}$) jest zapisem pewnego algorytmu. Dla każdego zadania nauczyciel wyznacza zbiór tzw. algorytmów dopuszczalnych (a więc zbiór rozwiązań dopuszczalnych), które są przez system analizowane i oceniane.

^{*} Pojęcie interindywidualnej trudności zadania wprowadził Zimny w pracy [7], s. 243.

Przypisywanie wag odpowiedziom elementarnym powinno przebiegać w specyficzny sposób.

Algorytm $\bar{a}$, należący do zbioru algorytmów dopuszczalnych, przedstawimy w postaci ciągu $\bar{a} = (a_1, \dots, a_m)$ kroków. Kroki te są na ogół opisami czynności nieelementarnych, realizujących algorytm (np. "wyznacz sumę dodatnich wyrazów ciągu"). Wspomniane rozbitcie algorytmów na kroki powinno być wykonane w ten sposób, aby można było określić wagę każdego kroku w zbiorze A , którego elementami są kroki wszystkich algorytmów dopuszczalnych (dla określonej klasy zadań systemu). Przypisywanie wag krokom algorytmów nie jest trudne - określa się je według roli, jaką zdaniem nauczyciela gra znajomość kroków w nauce programowania lub algorytmizacji zagadnień.

Każdemu algorytmowi dopuszczalnemu $(a_1, \dots, a_m)$ odpowiada rozwiązanie dopuszczalne $(b^{\#}, c^{\#}, d^{\#}, \dots)$ lub inaczej mówiąc - odpowiedź dopuszczalna $(q_1^{\#}, \dots, q_k^{\#})$. Każdy krok algorytmu jest realizowany za pomocą jednego lub kilku bloków. Znając wagę kroku przypisujemy wagę realizującym go blokom i tym samym wszystkim odpowiedziom elementarnym $(q_1^{\#}, \dots, q_k^{\#})$.

Opisywaną metodę przypisywania wag odpowiedziom elementarnym można zatem zapisać następująco:

1^o każdemu krokowi $a \in A$ przypisujemy wagę $g(a) = w \in N$, $g: A \rightarrow N$,

2^o dla każdego kroku $a \in A$ przypisujemy wagi blokom $b \in B$, realizującym ten krok,

3^o korzystając z wag bloków, realizujących kroki algorytmu $(a_1, \dots, a_m)$, przypisujemy wagę w_i każdej odpowiedzi elementarnej $q_i^{\#}$ z odpowiedzi dopuszczalnej $(q_1^{\#}, \dots, q_k^{\#})$.

Podane przez studenta rozwiązanie $(b, c, d, \dots)$, czyli odpowiedź $(q_1, \dots, q_k)$, jest przez system porównywane kolejno ze wszystkimi odpowiedziami dopuszczalnymi ze zbioru $Q = \{(q_1^{\#}, \dots, q_k^{\#})\}$.

Gdy porównujemy odpowiedzi studenta $(q_1, \dots, q_k)$ z ustaloną odpowiedzią dopuszczalną $(q_1^{\#}, \dots, q_k^{\#})$, wówczas tworzymy ciąg ocen elementarnych, tzn. ciąg $(h_1, \dots, h_k)$ liczb określonych następująco:

$$h_i = \begin{cases} w_i & , \text{ gdy } q_i = q_i^{\#} \\ 0 & , \text{ gdy } q_i \neq q_i^{\#} \end{cases}$$

a następnie obliczamy sumę $\sum_{i=1}^k h_i$ ocen elementarnych. Po wykonaniu tych czynności dla wszystkich odpowiedzi dopuszczalnych przyjmujemy jako ocenę h^S odpowiedzi studenta - największą z otrzymanych sum. Uzyskana w ten sposób ocena h^S wymaga jeszcze punktu odniesienia,

umożliwiającego porównanie jej z innymi ocenami za to zadanie. Tym punktem odniesienia może być waga zadania, zdefiniowana jako liczba w , równa maksymalnej ocenie h^s , jaką może otrzymać student za rozwiązanie zadania.

Opisany sposób określania oceny studenta przez porównywanie ze wszystkimi odpowiedziami dopuszczalnymi jest bardzo czasochłonny (i pamięciochłonny) i niezbędne jest poszukiwanie bardziej efektywnych metod. W systemie DIALOS/MLK problem ten rozwiązano zastępując zbiór Q tzw. grafem analizy rozwiązania [2,9]. Węzłami tego grafu są wszystkie niejednakowe odpowiedzi elementarne ze wszystkich odpowiedzi dopuszczalnych, a ścieżki odwzorowują wszystkie odpowiedzi dopuszczalne.

2.3.2. Statystyczna analiza wyników sprawdzianu

Przeprowadzając sprawdzian dysponujemy zbiorem $U = \{u_1, \dots, u_l\}$ testowanych studentów oraz zbiorem $Z = \{z_1, \dots, z_r\}$ zadań kontrolnych i każdemu studentowi ze zbioru U zadajemy wybrany zestaw zadań ze zbioru Z .

Aby przedstawić metodę określania trudności zadania oraz sprawności funkcjonalnej studenta, wyodrębniemy trzy sytuacje, w których zachodzą relacje między elementami zbioru U a elementami zbioru Z :

- 1) student $u_i \in U$ rozwiązuje zadanie $z_j \in Z$,
- 2) podzbiór $X \subset U$ studentów rozwiązuje zadanie $z_j \in Z$,
- 3) student $u_i \in U$ rozwiązuje podzbiór $Y \subset Z$ zadań.

Sytuacja 1

Po analizie rozwiązania j -tego zadania zostaje wystawiona i -temu studentowi ocena P_{ij} , obliczona ze wzoru:

$$P_{ij} = \frac{h_{ij}}{w_j}, \quad i \in \{1, \dots, l\}, \quad j \in \{1, \dots, r\},$$

gdzie h_{ij} oraz w_j są to określone, w sposób podany w pktcie 2.3.1, ocena h^s i -tego studenta za j -te zadanie oraz waga j -tego zadania.

Ocenę P_{ij} można uznać za miarę trudności, jaką sprawia i -temu studentowi rozwiązanie j -tego zadania. Jest to tzw. trudność indywidualna j -tego zadania. Przyjmujemy, zgodnie z pktm 2.2, że $D_{ij} = 1 - P_{ij}$ jest stopniem trudności indywidualnej j -tego zadania.

Sytuacja 2

Trudnością interindywidualną j -tego zadania będziemy nazywali średnią arytmetyczną S_j^x stopni trudności indywidualnych j -tego zadania w grupie studentów $X \subset U$, liczącej k -osób:

$$S_j^x = \frac{\sum_{i=1}^k D_{ij}}{k}.$$

Jeżeli rozkład ocen w grupie studentów X jest normalny, to trudność interindywidualną S_j^x można odnieść do całej badanej populacji studentów U . Interindywidualną trudność zadania nazywać będziemy odąd stopniem trudności zadania.

Sytuacja 3

Sprawność funkcjonalną i -tego studenta określać będziemy jako średnią arytmetyczną S_i^y ocen, otrzymanych przez i -tego studenta, za rozwiązywanie zestawu zadań $Y \subset Z$, liczącego t zadań:

$$S_i^y = \frac{\sum_{j=1}^t P_{ij}}{t}.$$

Średnią sprawność funkcjonalną S^u badanej populacji studentów U możemy oszacować, w przypadku normalności rozkładu indywidualnych sprawności funkcjonalnych w grupie studentów $X \subset U$ liczącej k osób, w następujący sposób:

$$S^u = \frac{\sum_{i=1}^k S_i^y}{k}.$$

Oprócz stopnia trudności zadania i sprawności funkcjonalnej studenta, organizatora procesu dydaktycznego interesuje odpowiedź na pytanie, czy stopień trudności zadania jest dostosowany do średniej sprawności funkcjonalnej testowanej grupy studentów.

Guilford, jako miarę dostosowania trudności zadania do średniej sprawności funkcjonalnej populacji studentów, proponuje współczynnik asymetrii rozkładu w próbie, określony przez Pearsona [6], s. 129 i n.

$$As = \frac{\bar{x} - Mo}{s_x},$$

gdzie $\bar{x}$ - średnia arytmetyczna,

Mo - wartość modalna,

s_x - odchylenie standardowe.

Jednakże miara Pearsona jest niewygodna w obliczeniach ze względu na konieczność określania dodatkowego parametru rozkładu - wartości

modalnej. Dlatego proponuję przyjęcie jako miary dostosowania stopnia trudności zadania do średniej sprawności funkcjonalnej tzw. współczynnika skośności rozkładu w populacji. Współczynnik ten, oznaczany zwykle jako γ , jest określony wzorem [1], s. 19:

$$\gamma = \frac{\mu_3}{\sigma^3},$$

gdzie μ_3 - trzeci moment centralny,
 σ - dyspersja.

Dla rozkładów symetrycznych (m.in. dla rozkładu normalnego) parametr $\gamma = 0$.

Analogicznie współczynnik skośności w próbie można obliczyć ze wzoru

$$g = \frac{M_3}{s^3},$$

gdzie M_3 oraz s są odpowiednio estymatorami μ_3 i σ w próbie.

Dla j -tego zadania współczynnik skośności w próbie liczącej k -studentów będzie określony wzorem

$$g_j = \frac{\frac{1}{k} \sum_{i=1}^k (p_{ij} - \bar{p}_j)^3}{\left[\frac{1}{k} \sum_{i=1}^k (p_{ij} - \bar{p}_j)^2 \right]^{3/2}},$$

gdzie $\bar{p}_j$ - średnia wartość ocen w próbie.

Jeżeli $g_j = 0$, to przyjmujemy, że trudność j -tego zadania została trafnie dobrana do średniej sprawności funkcjonalnej grupy studentów. Gdy $g_j > 0$, wówczas możemy uznać zadanie za zbyt łatwe, a gdy $g_j < 0$ - za zbyt trudne.

3. WPLYW SPOSOBU SFORMULOWANIA TEMATU NA STOPIEŃ TRUDNOŚCI ZADANIA

3.1. Hipoteza robocza

Przyjęto następującą hipotezę roboczą H : Na zmniejszenie stopnia trudności zadania wpływają istotnie dwa czynniki:

H_1 - sformułowanie treści zadania w sposób sugerujący plan rozwiązania,

H2 - osadzenie tematyki zadania w typowej problematyce technicznej.

Czynniki wymienione w tej hipotezie zostały uwzględnione w istotny sposób w tematach zadań, będących przedmiotem eksperymentu dydaktycznego.

Opracowując tematy tych zadań, wyodrębniono w treści zadania dwa zasadnicze elementy:

1. Przedstawienie zagadnienia obejmujące opis słowny lub matematyczny problemu,
2. Informacje dotyczące kolejności wykonywania czynności, niezbędnych do ułożenia programu.

3.2. Tematy zadań eksperymentalnych

Z katalogu zadań systemu DIALOS/MLK wybrano zadanie 34 o typowo sformułowanym temacie i należące do kategorii przeciętnie trudnych zadań. Treść tego zadania zawiera wskazówki dotyczące ułożenia programu, są one jednak włączone w opis problemu.

Do zadania 34 ułożono dwa nowe tematy, otrzymując w ten sposób trzy różne zadania z identycznym rozwiązaniem. Pierwszy z nowo opracowanych tematów, oznaczony numerem 34/1 spełnia obydwa postulaty zawarte w hipotezie H; drugi temat, 34/2, pomija zupełnie słowny opis zagadnienia, a zawarte w treści sugestie dotyczące sposobu ułożenia programu są podobne jak w zadaniu 34/1.

Można powiedzieć, że temat 34/1 jest "tematem dla inżyniera", a temat 34/2 jest "dla programisty".

34

28/09/76

Napisać fragment programu, w którym wczytane są wartości średnie temperatury dla kolejnych dwunastu miesięcy dla 100 różnych miejscowości. Wartości te są umieszczone w macierzy TEMP o rozmiarach 100×12 . Dla każdej miejscowości należy wyznaczyć średnią wartość temperatury w ciągu dwunastu miesięcy. Obliczone wartości średnie należy umieścić w tablicy jednowymiarowej DTEMP. Należy wypisać wartość średnią temperatury wraz z numerem kolejnych miejscowości dla wszystkich 100 miejscowości.

zał. 12 bil.

Temat zadania 34

34/1

28/09/76

W laboratorium podzespołów elektronicznych, dla tranzystorów dużej mocy, zbadano zdolność odprowadzania ciepła do otoczenia. Badania przeprowadzono na próbce liczącej 100 tranzystorów, mierząc w odstępach godzinowych przez czas 12 godzin temperaturę każdego tranzystora. Ułożyć fragment programu, który:

- 1) wczytuje wartości pomiarów do tablicy $TEMP(i,j)$, $i \in \langle 1, 100 \rangle$, $j \in \langle 1, 12 \rangle$,
- 2) oblicza średnią arytmetyczną temperatur każdego tranzystora w czasie 12 godzin,
- 3) obliczone wartości średnie umieszcza w tablicy $DTEMP(k)$, $k \in \langle 1, 100 \rangle$,
- 4) drukuje wartość średnią temperatury tranzystora wraz z jego numerem - dla wszystkich 100 tranzystorów

zał. 12 bil.

Temat zadania 34/1

34/2

28/09/76

Ułożyć fragment programu, który:

- 1) wczytuje dane do tablicy $TEMP(i,j)$, $i \in \langle 1, 100 \rangle$, $j \in \langle 1, 12 \rangle$,
- 2) oblicza średnią arytmetyczną elementów każdego wiersza,
- 3) obliczane wartości średnie umieszcza w tablicy $DTEMP(k)$, $k \in \langle 1, 100 \rangle$,
- 4) drukuje wartości kolejnych elementów tablicy $DTEMP$ wraz z numerem tego elementu

zał. 12 bil.

Temat zadania 34/2

4. EKSPERYMENT I JEGO WYNIKI

4.1. Opis eksperymentu

Na wyniki zadania kontrolnego z języka Fortran wpływają również inne czynniki, nie wymienione w hipotezie H. Ważniejsze z nich to:

1. Zrozumienie problemu przedstawionego w zadaniu i umiejętność jego rozwiązania.
2. Poziom opanowania materiału z programowania w języku Fortran.
3. Zaznajomienie się z rozwiązywaniem zadań w systemie DIALOS/MLK.

Ponieważ celem pracy było zbadanie zależności między sposobem sformułowania treści zadania kontrolnego z języka Fortran a stopniem trudności tego zadania, więc przez odpowiedni dobór grup studenckich uczestniczących w eksperymencie ograniczono do minimum wpływ ostatniego czynnika na ocenę i wyrównano wpływ dwóch pierwszych czynników.

Ułożone przez studentów rozwiązania zadań zostały poddane analizie w systemie DIALOS/MLK i w wyniku analizy studenci otrzymali protokoły analizy rozwiązania. Protokół taki zawiera między innymi ocenę punktową składającą się z dwóch liczb: liczby punktów otrzymanej przez studenta oraz maksymalnej liczby punktów, które za to zadanie student może otrzymać (p. [2,3]).

Uzyskane przez studentów oceny punktowe przedstawiono na histogramach w pracy [10].

4.2. Hipoteza statystyczna i jej weryfikacja

Przyjęcie hipotezy H, sformułowanej w pktcie 3.1, należy uznać za uzasadnione, jeżeli stopień trudności zadań o numerach 34/1 i 34/2 będzie mniejszy niż stopień trudności zadania 34 (hipoteza H1) oraz, gdy stopień trudności zadania 34/1 będzie mniejszy niż stopień trudności zadania 34/2 (hipoteza H2).

Hipoteza H stanowi podstawę do sformułowania trzech hipotez statystycznych H' , H'' i H''' :

$$H'_0 : m_2 \geq m_1 \quad \text{przy} \quad H'_1 : m_2 < m_1,$$

$$H''_0 : m_3 \geq m_1 \quad \text{przy} \quad H''_1 : m_3 < m_1,$$

$$H'''_0 : m_2 \geq m_3 \quad \text{przy} \quad H'''_1 : m_2 < m_3,$$

gdzie m_1 - wartość oczekiwana stopnia trudności zadania 34,
 m_2 - wartość oczekiwana stopnia trudności zadania 34/1,
 m_3 - wartość oczekiwana stopnia trudności zadania 34/2.

Hipotezę H należy uznać za prawdziwą w razie odrzucenia hipotez zerowych H'_0 , H''_0 i H'''_0 .

Do zweryfikowania hipotez H' , H'' i H''' posłużono się testami t-Studenta [1], s. 111 i v-Welcha i Aspin [4], s. 97. Wybór jednego z tych testów jest uzależniony od weryfikacji pomocniczej hipotezy H_p o równości wariancji w dwóch badanych populacjach. Hipoteza H_p ma następującą postać:

$$H_{p_0} : \sigma_1^2 = \sigma_2^2 \quad \text{przy} \quad H_{p_1} : \sigma_1^2 \neq \sigma_2^2,$$

gdzie σ_1^2 , σ_2^2 - wariancje w porównywanych zbiorowościach.

Do zweryfikowania hipotezy H_p wykorzystano test F-Snedecora [1], s. 108.

Jeżeli wariancje w badanych populacjach są równe, to stosuje się test t-Studenta. W przeciwnym razie należy wykorzystać test v-Welcha i Aspin.

Wspomniane testy są parametryczne i wymagają znajomości funkcji rozkładu. Najczęściej przyjmuje się założenie o normalności rozkładu, gdyż ułatwia to w znacznej mierze obliczenia i pozwala korzystać z opracowanych i powszechnie dostępnych tablic.

Mimo, że rozkład normalny jest zjawiskiem częstym, nie ma pewności, że rozkład stopni trudności indywidualnych zadania kontrolnego w testowanej grupie studentów jest również normalny. Dlatego, jeżeli niezbędne jest przeprowadzenie szczegółowej analizy statystycznej, w której istotne jest założenie o rozkładzie normalnym badanej populacji, należy sprawdzić czy oceniany rozkład jest dobrze przybliżony przez rozkład normalny. W niniejszej pracy wykorzystano do tego celu test zgodności λ -Kołmogorowa [5], s. 245.

Weryfikacji hipotezy o zgodności rozkładu stopni trudności indywidualnych z rozkładem normalnym, jak również weryfikacji hipotez H_p , H' , H'' i H''' dokonano za pomocą (specjalnie do tego celu napisanych przez autora) programów o nazwach API2 i API3. Opisy algorytmów programów API2 i API3 oraz wydruki wyników obliczeń przedstawiono w pracy [10].

Wyniki obliczeń programu API2 potwierdziły założenie o normalności rozkładu stopni trudności indywidualnych wszystkich trzech zadań. W pracy [10] zamieszczono zbiorcze wyniki testu zgodności λ -Kołmogorowa oraz wyniki testów F-Snedecora, t-Studenta i v-Welcha i Aspin.

Wyniki obliczeń programu API3 dały podstawę do odrzucenia hipotez H'_0 , H''_0 oraz nie dały podstawy do odrzucenia hipotezy H'''_0 , gdyż stopień trudności zadania 34 był większy niż stopnie trudności zadań 34/1 i 34/2 ($m_1 > m_2$ i $m_1 > m_3$), a stopnie trudności zadań 34/1 i 34/2 nie różniły się istotnie ($m_2 = m_3$).

5. PODSUMOWANIE

W punkcie 3.1 sformułowano hipotezę dotyczącą wpływu zredagowania tematu zadania na jego stopień trudności (określony według metody zaproponowanej w pierwszej części pracy w rozdz. 2). Zbadanie tej hipotezy miało dostarczyć informacji do określenia zasad opracowywania tematów zadań kontrolnych z języka Fortran.

Analiza statystyczna wyników eksperymentu dydaktycznego nie dała podstaw do odrzucenia pierwszego punktu hipotezy oraz odrzuciła punkt drugi. Wyniki analizy statystycznej stanowiły ponadto podstawę do wyciągnięcia następujących wniosków, dotyczących zarówno analizy stopni trudności zadań, jak i metody przedstawiania treści zadania:

1. Ścisłejsze związanie tematu zadania z typowymi zagadnieniami technicznymi konkretnego kierunku studiów nie ma istotnego wpływu na stopień trudności tego zadania.

2. Istotny wpływ na stopień trudności zadania ma taki sposób zredagowania tematu zadania, w którym wyraźnie są wypunktowane kolejne elementy, mające się znaleźć w rozwiązaniu. Sugerowanie kolejności tych elementów w sposób opisowy jest istotnym utrudnieniem zadania.

3. Rozkłady stopni trudności indywidualnych porównywanych zadań są w przybliżeniu normalne, co nie wyklucza wykorzystania współczynnika skośności jako miary dopasowania stopnia trudności zadania do średnie sprawności funkcjonalnej populacji studentów. Wyniki analizy skośności rozkładów ocen otrzymanych przez studentów pokrywają się z wynikami testów istotności różnic: zadanie 34 było zbyt trudne ($g_{34} = -0,176$), a zadania 34/1 i 34/2 - zbyt łatwe ($g_{34/1} = 0,295$, $g_{34/2} = 0,269$).

Przedstawione wnioski należy uzupełnić uwagą ogólniejszą: Mimo, że osadzenie tematu zadania w realiach technicznych nie wpływa istotnie na stopień trudności zadania, nie należy automatycznie preferować zadań o tematach oderwanych od konkretnych zastosowań. Opracowywanie zadań zawierających konkretny problem techniczny jest bez wątpienia bardziej kłopotliwe. Ze względów dydaktycznych należy jednak rozpowszechniać takie właśnie zadania.

LITERATURA

- [1] Brandt S., Metody statystyczne i obliczeniowe analizy danych, Warszawa 1976, PWN.
- [2] Cieślak A., Informatyczny system obsługi procesu nauczania DIALOS/MLK. Podsystem KONTROLA, Komunikaty Centrum Obliczeniowego PWR. Nr 26, Wrocław 1977.

- [3] Cieślak A., Pilański A., Zasady przygotowania i realizacji zajęć sprawdzających z wykorzystaniem wspomagania komputerowego (na przykładzie systemu MLK), Komunikaty Centrum Obliczeniowego PWR. Nr 17, Wrocław 1976.
- [4] Czermiński I. i in., Metody statystyczne w doświadczalnictwie chemicznym, Warszawa 1974, PWN.
- [5] Fisz M., Rachunek prawdopodobieństwa i statystyka matematyczna, Warszawa 1967, PWN.
- [6] Guilford J.P., Podstawowe metody statystyczne w psychologii i pedagogice, Warszawa 1964, PWN.
- [7] Praca zbiorowa pod red. Kozieleckiego J., Problemy psychologii matematycznej, Warszawa 1971, PWN.
- [8] Muciek J., Model metrologiczny zautomatyzowanego systemu kontroli i oceny wiedzy, Technologia Kształcenia, Zbiór referatów XI międzynarodowego sympozjum, Z. 13, Poznań, 21-23 września 1978.
- [9] Pilański A., Informatyczny system obsługi procesu nauczania DIALOS/MLK. Metoda opracowania zadań kontrolnych z języka FORTRAN, Komunikaty Centrum Obliczeniowego PWR. Nr 28, Wrocław 1977.
- [10] Pilański A., Informatyczny system obsługi procesu nauczania DIALOS/MLK. Wpływ sposobu sformułowania treści zadania na ocenę, Komunikaty Centrum Obliczeniowego PWR. Nr 27, Wrocław 1978.

Praca wpłynęła do Redakcji 9.03.1979 r.
Po poprawieniu 19.02.1980 r.

ON THE PROBLEM OF THE DIFFICULTY DEGREE OF EXERCISES USED IN CAI SYSTEMS

A method of the exercise difficulty degree estimation for certain class of exercises with constructed answer used in CAI systems is presented. The method of estimation of student's intellectual-functional efficiency, as well as the method of estimation of exercise difficulty degree fitting to the mean intellectual-functional efficiency of tested students are presented. The hypothesis about the influence of the exercise theme formulation upon the exercise difficulty degree is made and verified.

Verified by R. Wiernik

О ПРОБЛЕМАХ УРОВНЯ ТРУДНОСТИ ЗАДАЧ, РЕШАЕМЫХ В СИСТЕМАХ АСО

Для некоторого класса задач, решаемых в системах АСО, представлен метод оценки уровня трудности задачи с построенным ответом. Кроме того, описывается метод оценки интеллектуальной - функциональной эффективности и метод оценки адаптации уровня трудности задачи к средней интеллектуальной - функциональной эффективности проверяемых студентов. Представлена также и проверена гипотеза, касающаяся влияния метода формулирования задачи на уровень её сложности.

Проверил Чеслав Штуковский

*System konwersacyjny, język
programowania, wyrażenie
arytmetyczne*

Jerzy PISARSKI^{*)}, Lech TUZINKIEWICZ^{**)}

Konwersacyjny minisystem wyrażań arytmetycznych PT78

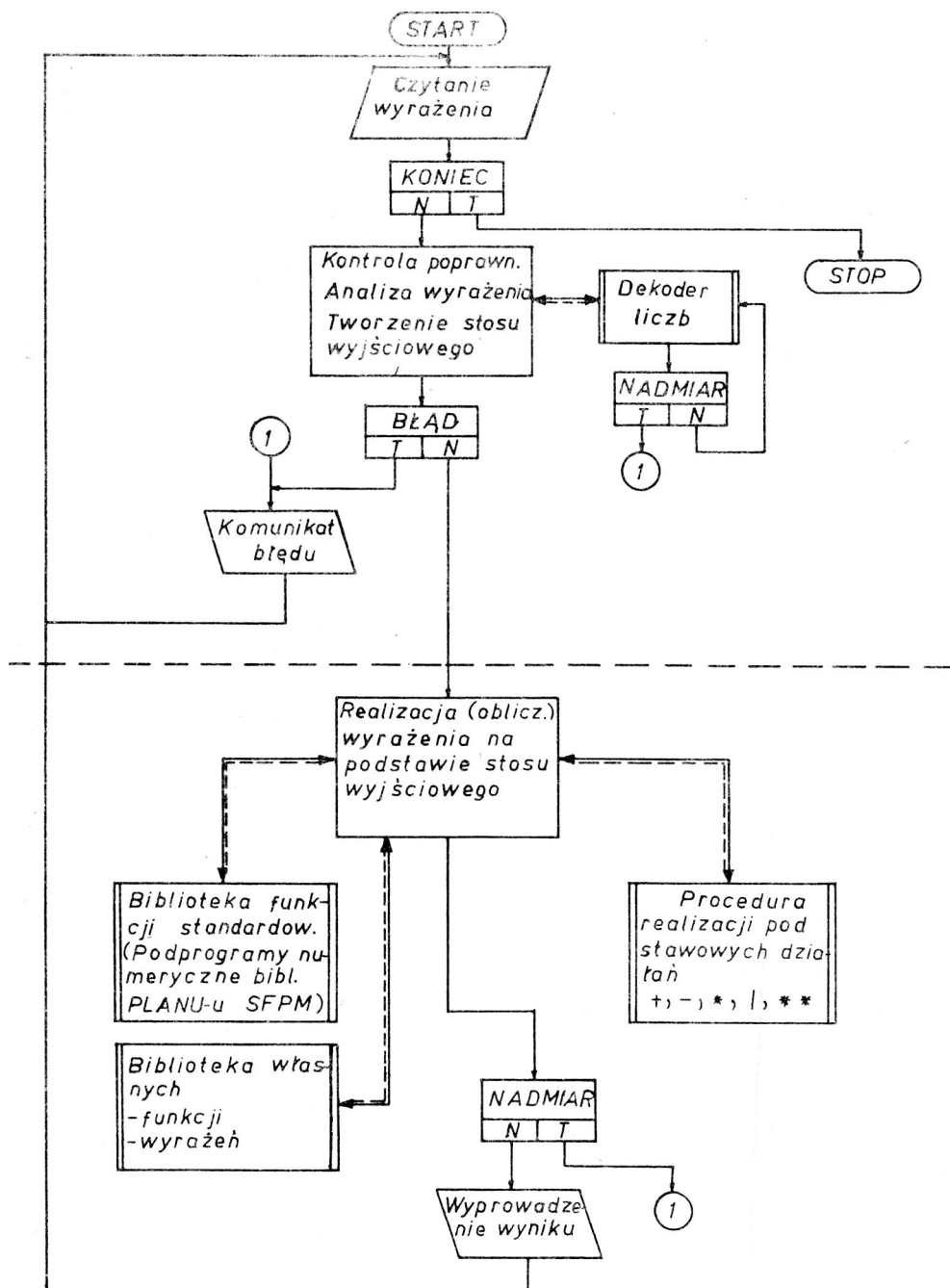
W ramach prac prowadzonych w Centrum Obliczeniowym Politechniki Wrocławskiej nad systemem komputerowego wspomagania dydaktyki powstała potrzeba wykonania "kalkulatora" do obliczenia wartości prostych wyrażań arytmetycznych. Stało się to podstawą opracowania minisystemu wyrażań arytmetycznych. Praca zawiera formalny opis koncepcji takiego minisystemu oraz przykład zrealizowanej przez autorów wersji praktycznej.

1. WSTĘP

Prezentowany tu minisystem PT78 do obliczania wartości wyrażań arytmetycznych może pracować na dowolnej maszynie cyfrowej serii Odra 1300, nie obciążając zbytnio pamięci operacyjnej. Minisystem ten jest systemem otwartym, tzn. użytkownik z łatwością może rozszerzać zakres wykonywanych przez system czynności przez dołączenie własnej biblioteki podprogramów lub wykorzystanie oprogramowania standardowego m.c., na której pracuje (szczegóły są podane w rozdz. 4). Minisystem został zbudowany jako jeden z modułów komputerowego systemu wspomaganie dydaktyki Labor [1], jest on jednak od tego systemu niezależny i może być wykorzystywany oddzielnie, np. aby umożliwić studentom (lub innym użytkownikom nie mającym doświadczenia w programowaniu) oblicza-

^{*)} Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

^{**)} Instytut Energoelektryki Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław



Rys.1. Struktura minisystemu PT78

Fig.1. PT78 minisystem structure

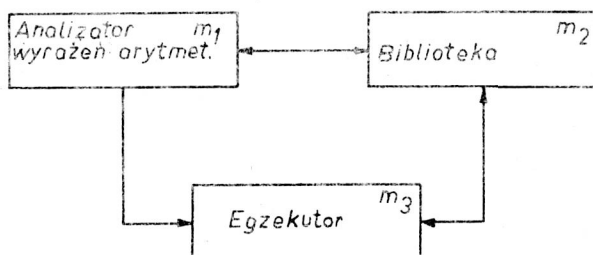
nia za pomocą m.c. wartości wyrażeń arytmetycznych o dużym repertuarze działań. Zamieszczony w pracy opis systemu PT78 ma charakter sformalizowany. Pewne pojęcia jednak zostały przedstawione w sposób nieformalny w celu zwiększenia czytelności. Ogólną strukturę minisystemu przedstawiono na rysunku 1.

2. KONCEPCJA SYSTEMU PT78

2.1. Założenia

Założeniem systemu PT78 jest struktura modułowa, przedstawiona na rys. 2. Poszczególne moduły są autonomiczne i ich zadaniem jest w sposób optymalny realizować kolejne etapy obliczania wartości wyrażeń arytmetycznych. Taka struktura umożliwia dokonywanie zmian wewnątrz poszczególnych modułów lub wymiany całych modułów. Dokonując najogólniejszego podziału funkcjonalnego można w minisystemie wyróżnić:

- analizator wyrażenia arytmetycznego,
- egzekutor zdekodowanego wyrażenia arytmetycznego.



Rys.2. Modułowa struktura minisystemu
Fig.2. Module structure of the minisystem

Definicja 1. Gramatykę generacyjną będziemy określali jako uporządkowaną czwórkę

$$G_G = (A_k, A_p, R, WYR AR),$$

gdzie A_k - alfabet końcowy gramatyki G_G ,

A_p - alfabet pomocniczy gramatyki G_G ,

R - skończony zbiór reguł generacyjnych gramatyki G_G ,

$WYR AR$ - cel gramatyki G_G .

Definicja 2. Językiem generowanym przez gramatykę G_G będziemy nazywali zbiór słów utworzonych na podstawie tej gramatyki.

D e f i n i c j a 3. Wyrażeniem arytmetycznym będziemy nazywali ciąg symboli należących do alfabetu języka wyrażeń arytmetycznych, którego poprawność formalną można sprawdzić za pomocą reguł generacyjnych języka.

Ponieważ celem gramatyki G_G jest wyrażenie arytmetyczne, będziemy dalej nazywali język generowany przez G_G językiem wyrażeń arytmetycznych.

2.2. Język wyrażeń arytmetycznych

2.2.1. Alfabet języka. Znaki alfabetu

1. Litery alfabetu łacińskiego od A do Z.
2. Cyfry dziesiętne od 0 do 9.
3. Operatory arytmetyczne $+$ $-$ $*$ $/$ $**$
4. Ograniczniki $() . =$

Alfabet pomocniczy:

$$A_p = \{ \langle Z_n \rangle, \langle L \rangle, \langle C \rangle, \langle OP \rangle, \langle OP_A \rangle, \langle OP_B \rangle, \langle FUN \rangle, \langle Z_m \rangle, \langle LIT \rangle, \langle OPERAND \rangle, \langle TERM \rangle, \langle WYR AR \rangle, \langle F \rangle \}.$$

Alfabet końcowy:

A_k - słowa generowane przez gramatykę, nie należące do A_p .

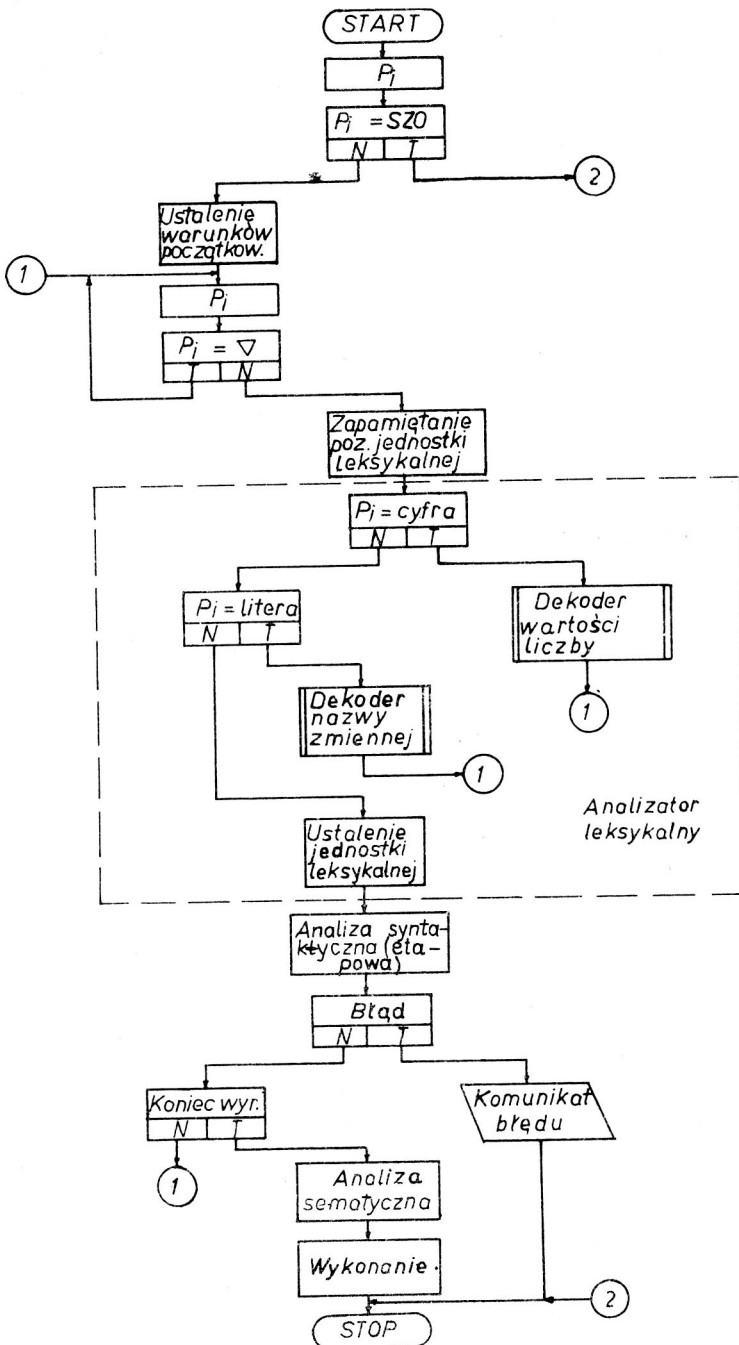
$$A_k \cap A_p = \emptyset.$$

2.3.2. Reguły gramatyczne G_G

$$\begin{aligned} \langle Z_n \rangle &::= \langle L \rangle | \langle C \rangle | \langle OP \rangle | \langle OG \rangle^*) \\ \langle L \rangle &::= A | B | \dots | Y | Z \\ \langle C \rangle &::= 0 | 1 | 2 | \dots | 9 \\ \langle OP \rangle &::= \langle OP_A \rangle | \langle OP_B \rangle \\ \langle OP_A \rangle &::= + | - \\ \langle OP_B \rangle &::= * | / \\ \langle FUN \rangle &::= SIN | COS | \dots | LOG \\ \langle Z_m \rangle &::= \langle L \rangle | \langle Z_m \rangle \langle L^* \rangle | \langle Z_m \rangle \langle C^* \rangle^{**}) \\ \langle LIT \rangle &::= \langle C \rangle \langle C^* \rangle | . \langle C \rangle \langle C^* \rangle | \langle C \rangle \langle C^* \rangle . \langle C^* \rangle \\ \langle OPERAND \rangle &::= \langle LIT \rangle | \langle Z_m \rangle | \langle WYR AR \rangle | \langle OPERAND \rangle ** \langle WYR AR \rangle \\ \langle TERM \rangle &::= \langle OPERAND \rangle | \langle TERM \rangle \langle OP_B \rangle \langle OPERAND \rangle \\ \langle F \rangle &::= \langle FUN \rangle (| \langle F \rangle \langle WYR AR \rangle) \\ \langle NEG \rangle &::= - \langle WYR AR \rangle \\ \langle WYR AR \rangle &::= \langle TERM \rangle | \langle F \rangle | \langle WYR AR \rangle \langle OP_A \rangle \langle TERM \rangle | \langle NEG \rangle \end{aligned}$$

*) $\langle A \rangle ::= B | C$ oznacza, że $\langle A \rangle$ może przyjąć znaczenie B lub C

**) Element z gwiazdką określa ciąg danego typu; w szczególnym przypadku może to być ciąg pusty.



Rys. 3. Schemat analizatora wyrażeń arytmetycznych
 Fig. 3. Scheme of arithmetic expression analyser

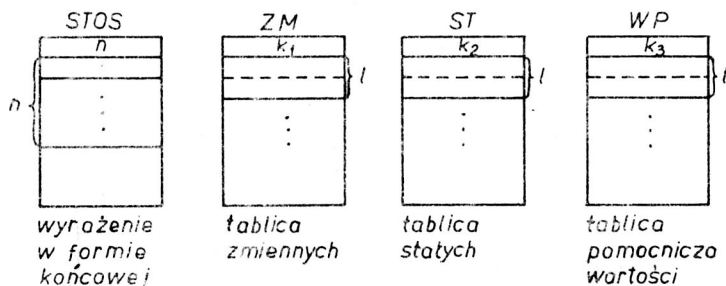
2.3. Analizator wyrażeń arytmetycznych

Analizator wyrażeń arytmetycznych pełni rolę filtru przepuszczającego tylko taki ciąg znaków, który może być wyprowadzony przez gramatykę G_G języka wyrażeń arytmetycznych. Każda niezgodność jest wyszukiwana, a następnie jest przekazywana informacja o rodzaju błędu z podaniem pozycji w ciągu znaków. Gdy kontrola poprawności zostanie przeprowadzona do końca bez wykrycia błędu, wówczas ciąg znaków jest interpretowany jako wyrażenie arytmetyczne i jest obliczana jego wartość.

Analizator składa się z trzech podstawowych części:

- analizatora leksykalnego, którego zadaniem jest wyodrębnienie poszczególnych jednostek leksykalnych (słów) z całości,
- analizatora syntaktycznego - sprawdzającego poprawność zdania pod względem reguł konstrukcji wyrażeń arytmetycznych,
- analizatora semantycznego interpretującego zdanie (określającego sens zdania).

Cały proces analizy od strony ideowej jest przedstawiony na rysunku 3.



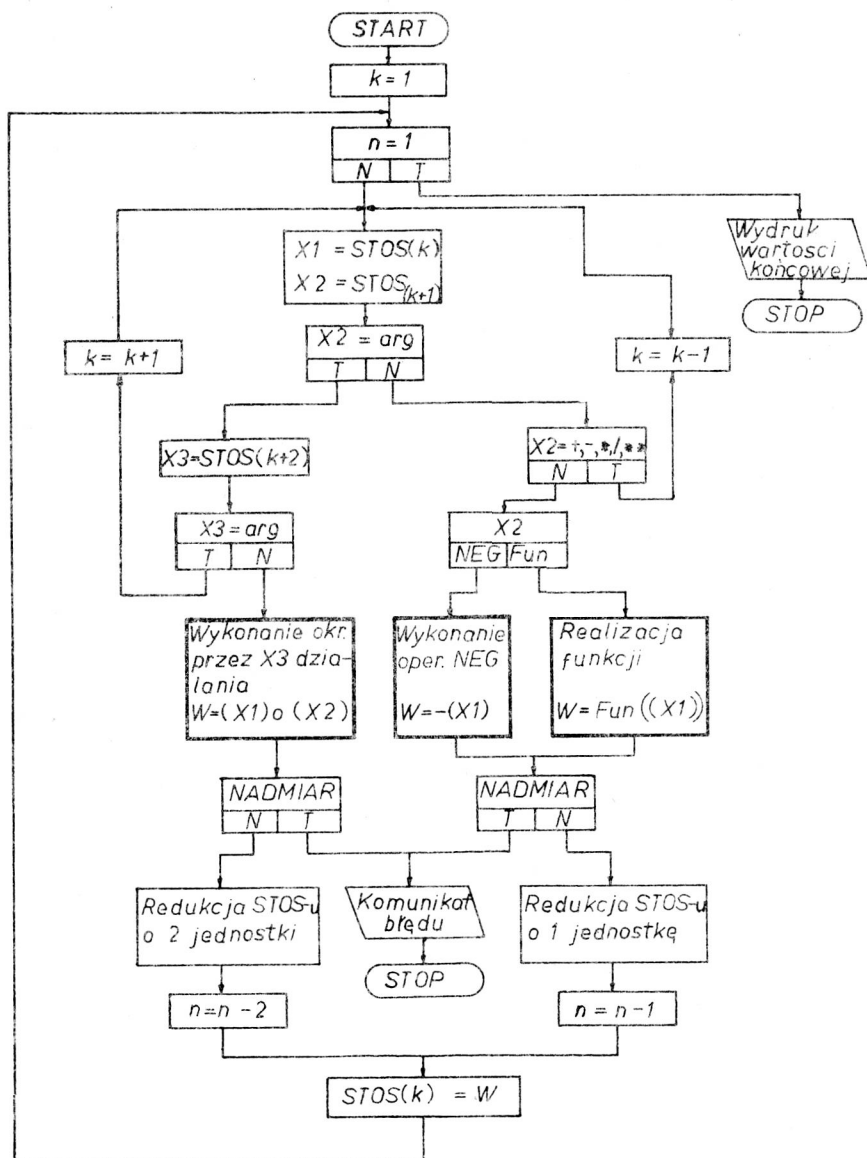
2.4. Egzekutor zakodowanego wyrażenia arytmetycznego

Obliczenie wartości wyrażenia arytmetycznego, rozłożonego na ciąg elementów i umieszczonego w tablicy STOS (patrz rys. 4), odbywa się zgodnie z algorytmem przedstawionym na rys.5. Tablica ta zawiera, jak wspomniano wyżej, kody operacji i funkcji oraz adresy argumentów. Same zaś argumenty są umieszczone w tablicy zmiennych (ZM), tablicy stałych (ST) oraz - podczas przetwarzania - w tablicy pomocniczej (WP). Podstawowe operacje arytmetyczne są działaniami dwuargumentowymi, funkcje natomiast są traktowane jako działania jednoargumentowe, podobnie jak to się dzieje z operacją negacji.

Algorytm realizacji stosu - jak to pokazano na rysunku 5 - polega na wybraniu z tablicy STOS pary argumentów i następującego po nich kodu operacji, a następnie wykonaniu wskazanego działania z jednoczesną redukcją stosu. Dzieje się tak w przypadku podstawowych działań arytmetycznych, w przypadku funkcji natomiast jest pobierany jej argument oraz kod identyfikacyjny, a następnie argument jest przekazany do segmentu realizującego funkcję. Wynik zostaje umieszczony w tablicy WP, a jego adres w tablicy STOS, zarazem jest dokonywana odpowiednia redukcja stosu. W trakcie wykonywania kolejnych działań i obliczania wartości funkcji są testowane wyniki operacji ze względu na dopuszczalną wielkość liczb (nadmiar). W razie przekroczenia dopuszczalnego zakresu jest wyprowadzany komunikat błędu i realizacja obliczania wartości wyrażenia zostaje przerwana.

Objaśnienia do rysunku 5:

- n - liczba elementów stosu,
- k - bieżący wskaźnik wierzchołka stosu,
- arg - adres argumentu,
- X1, X2, X3 - zmienne pomocnicze, zawierające elementy stosu (adres argumentu lub kod działania),
- (X1), (X2) - wartości o adresach X1, X2,
- NEG - negacja,
- = - podstawienie,
- - podstawowe operacje arytmetyczne.



Rys.5. Algorytm realizacji wyrażenia arytmetycznego
Fig.5. Algorithm for arithmetic expression realization

3. REALIZACJA I OPIS UŻYTKOWY PT78

3.1. Opis użytkowy

Sposób korzystania z minisystemu PT78 jest niezależny od tego, czy program jest uruchamiany w trybie wsadowym czy konwersacyjnym (on-line). Różnice polegają jedynie na aktywacji minisystemu.

Wyrażenia arytmetyczne, podawane przez użytkownika, muszą mieć strukturę wyrażen języków wyższego poziomu, a postać ich musi być taka sama jak postać wyrażen języka Fortran. Wyrażenia realizowane w minisystemie mogą zawierać stałe, zmienne bądź funkcje jednoargumentowe. Niedopuszczalne jest używanie zmiennych ze wskaźnikami oraz funkcji wieloargumentowych. Zmienne, którym nie nadano wartości, przyjmują wartość zero. Po obliczeniu wartości wyrażenia użytkownik otrzymuje wynik liczbowy, którego typ jest zależny od podanych argumentów według zasad przyjętych w języku Fortran dla zmiennych typu integer oraz real.

W razie niepoprawnego zapisu wyrażenia, analiza zostaje przerwana, a użytkownik otrzymuje komunikat informujący o miejscu i rodzaju popełnionego błędu *).

Dyrektwy minisystemu muszą być poprzedzone standardowym znakiem ostrzeżenia : . W przeciwnym przypadku słowo dyrektywy zostanie zinterpretowane jako nazwa zmiennej. Liczba wykonywanych obliczeń po uruchomieniu minisystemu jest dowolna, a o zakończeniu pracy informuje użytkownik podaniem dyrektywy KONIEC.

Użytkownik ma do dyspozycji następujące działania i funkcje:

- dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie,
- podstawienie,
- funkcje i wyrażenia własne,
- funkcje standardowe; są to w zasadzie używane w dydaktyce funkcje standardowe języków wyższego poziomu.

Możliwość zadawania własnych funkcji pozwala na rozszerzenie zakresu zastosowania minisystemu. Sposób korzystania z tych funkcji jest identyczny jak dla funkcji bibliotecznych (standardowych).

Funkcje własne użytkownika zadaje się w postaci

'FUN(X)'f(x)

gdzie FUN - nazwa funkcji (max. czteroznakowa),
X - argument,
f(x) - wyrażenie definiujące funkcję.

*) Poprawność wyrażenia w dalszym ciągu nie jest sprawdzana

Przykład: 'ROT(X)'(A+B)*RKP(X)

Podczas pracy minisystemu można korzystać z następujących dyrektyw, dotyczących funkcji użytkownika:

:USUN FUN - kasowanie poprzednio zdefiniowanej funkcji,

:DRUKUJ FUN - wydruk zdefiniowanej funkcji.

Minisystem stwarza również możliwość zapamiętania dowolnego wyrażenia arytmetycznego (tzw. wyrażenia własnego), nie mającego jednak własności funkcji. Wyrażenie takie przyjmuje postać: 'nr' wa gdzie nr - numer identyfikacyjny ($0 \div 9$),

wa - wyrażenie arytmetyczne lub instrukcja podstawienia arytmetycznego.

Przykład: '5' SIN(X) + COS(Y).

W przykładzie tym pod zmienne X i Y można podstawiać dowolne wartości i za pomocą dyrektywy WYKONAJ żądać podania wartości danego wyrażenia. Stwarza to możliwość pośredniej realizacji funkcji wieloargumentowej oraz wygodniejszego tablicowania wartości wyrażenia. Patrz również przykład 3. w pktcie 3.1.1. Dyrektywami używanymi w tym przypadku są:

:WYKONAJ i - wykonanie wyrażenia o numerze i,

:USUN i - kasowanie wyrażenia o numerze i,

:DRUKUJ i - wydruk postaci wyrażenia i.

3.1.1. Przykłady

1. Wprowadzenie zmiennej

←A = 34.5 0.3450000000E 02

←B = 1 + A 0.3550000000E 02

2. Obliczenie wyrażenia

- prostego

←SQRT(1024) 0.3200000000E 02

Uwaga. Argument funkcji może być liczbą całkowitą; w takim przypadku jest dokonywana konwersja automatyczna.

- złożonego

←(1 + 2)/(2. + 4.) + ALOG(10.) 0.2802585093E 02

3. Wyrażenia własne

←'1' X = X + 2

←X = 3

←:WYKONAJ 1

5

←:WYKONAJ 1

7

itd.

4. Funkcje własne

```
←'SC(X)' SIN(X)/COS(X)
←TAN(1.2)          0.2572151622E 01
←SC(1.2) + 1      0.3572151622E 01
```

itp.

Uwaga. Słowo dyrektywy można podawać w postaci skróconej, np.

:W 2

:DRUK FUN

3.1.2. Komunikaty

1. BŁĄD NADMIARU
2. BRAK OPERATORA
3. BRAK LICZBY
4. ZA DUŻA LICZBA
5. NIEPOPRAWNA POSTAC WYKŁADNIKA LICZBY
6. ZŁE UŻYTY OPERATOR, OGRANICZNIK LUB NAWIAS
7. BRAK NAWIASU OTWIERAJĄCEGO
8. BRAK NAWIASU ZAMYKAJĄCEGO
9. NIEPOPRAWNA POSTAC FUNKCJI
10. NIEROZPOZNANA FUNKCJA
11. NIEPOPRAWNY ARGUMENT FUNKCJI ALOG,_A X,SQRT
12. NIEPOPRAWNY ARGUMENT FUNKCJI ASIN, ACOS
13. BRAK WYRAŻENIA
14. BRAK FUNKCJI
15. WYRAŻENIE nr ZOSTAŁO ZDEFINIOWANE
16. FUNKCJA fun ZOSTAŁA ZDEFINIOWANA
17. NIEPOPRAWNY NUMER WYRAŻENIA

3.2. Możliwości wykorzystania

Minisystem PT78 jest z założenia programem konwersacyjnym, dostosowanym do pracy w systemach wielodostępnych. Uruchomienie i wstępna eksploatacja była przeprowadzona w systemie Minimop. PT78 może być implementowany w innych systemach wielodostępnych, pracujących na maszynach serii Odra 1300, jak np. Georg 3 itp. Można także wykorzystywać program PT78 do pracy wsadowej i to bez jakichkolwiek zmian. Tak szerokie możliwości wykorzystania minisystemu zostały osiągnięte dzięki odpowiednio zorganizowanej strukturze danych i gospodarce pamięcią. Minisystem zawiera wszystkie niezbędne obszary (tablice) do przechowywania przetwarzanych wyrażeń oraz własną bibliotekę funkcji standar-

dowych, nie zachodzi więc konieczność korzystania z zasobów pamięci zewnętrznych, których organizacja jest odmienna w różnych systemach.

PT78 nie następuje praktycznie żadnych kłopotów podczas korzystania z systemu dzięki niewielkiej liczbie dyrektyw potrzebnych do pracy z systemem oraz standardowej postaci wyrażen arytmetycznych, wzorowanych na wyrażeniach języka Fortran. Ułatwia to znacznie pracę użytkownikowi, mającemu podstawowe wiadomości z programowania w językach wyższego poziomu.

Podczas pracy w trybie wsadowym można używać gotowych kart perforowanych z zapisem wyrażen arytmetycznych, stosowanych w programach Fortranowskich. Ograniczeniem jest tu brak wierszy kontynuacyjnych.

Udogodnieniem w każdym reżimie pracy jest możliwość przechowywania, a następnie wykorzystania (w dalszych zdaniach) wcześniej obliczonych wielkości, przechowywanych w odpowiednich tablicach (patrz pkt 2.4) oraz definiowanie wyrażen i funkcji własnych. Daje to możliwość pisanie miniprogramów, realizujących proste algorytmy obliczeniowe.

4. PROPOZYCJE MODYFIKACJI SYSTEMU

Propozycje, które zostaną przedstawione mają na celu zasygnalizowanie użytkownikom możliwości dokonywania zmian, które doprowadzą do znacznego rozszerzenia zakresu zastosowania minisystemu.

1. Rozbudowa biblioteki funkcji standardowych przez dołączanie funkcji języków wyższego poziomu.
2. Dołączenie funkcji wieloargumentowych (forma podobna do realizacji segmentów SUBROUTINE języka Fortran).
3. Przechowywanie (składowanie) i wykonywanie ciągu wyrażen, co umożliwi budowę programów własnych.
4. Zastosowanie wielokrotnej instrukcji podstawienia.
5. Zwiększenie dokładności obliczeń (liczby podwójnej dokładności).
6. Inne formy wyniku, np. postać ułamkowa liczby.
7. Dokładniejsza diagnostyka podczas wykonywania działań - tzw. śledzenie obliczeń.

Naturalnie system można też rozbudowywać w innych kierunkach w zależności od warunków eksploatacyjnych oraz potrzeb użytkownika. Za jego rozbudowę przemawiają następujące argumenty:

- prostota obsługi i łatwość eksploatacji (jak dla typowego kalkulatora),
- standardowa postać wyrażen,
- zadowalająca dokładność (10 cyfr znaczących),
- czytelna postać wydruków oraz dokładna informacja o błędach.

LITERATURA

- [1] Kuźniarz L., Pisarski J., Rychlikowski E., Tuzinkiewicz L., System KONFORT. Założenia projektowe i opis ogólny, Raporty Centrum Obliczeniowego PWr. nr 105, Wrocław 1977.
- [2] Battek J., Rudakowa B., Rudak B., System operacyjny MINIMOP, Wrocław 1975, Wyd. PWr.
- [3] Turski W., Podstawy użytkowania maszyn cyfrowych w ośrodkach naukowo-technicznych, Warszawa 1973, PWN.
- [4] Lebedev V.N., Vvedenie v sistemy programirovaniya, Moskva 1975, Statistika.

Praca wpłynęła do Redakcji 10.12.1978 r.
Po poprawieniu 19.02.1980 r.

CONVERSATIONAL MINISYSTEM OF ARITHMETICAL EXPRESSION PT78

The need of constructing a "calculator" for computing simple arithmetical expressions arose in the course of realization of the KONFORT system. It constituted the basis for the presented minisystem for arithmetical expression computing. Short description of both the concepts and the data referring to the practical version presented by the authors of the Wrocław Technical University Computer Center is included.

Verified by R. Wiernik

ДИАЛОГОВАЯ МИНИСИСТЕМА АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ ПТ78

При реализации системы КОНФОРТ возникла потребность в создании "устройства" для вычисления простых арифметических выражений. Это стало основанием для изложенной ниже минисистемы решения арифметических выражений. Кратко описаны концепция и данные, касающиеся практической версии, изготовленной авторами в Вычислительном центре Wrocławского технического университета.

Проверил Чеслав Штуковский

*System informatyczny, baza
danych, wyprowadzanie informacji*

Zbigniew SZPUNAR^{*)}

Metoda generacji dokumentów w procesie wyprowadzania informacji z baz danych systemów informatycznych

Opisana w pracy metoda generacji dokumentów umożliwia wyprowadzanie informacji wyszukiwanych w bazie danych w formie tabelarycznych dokumentów. Postać tych dokumentów jest dostosowywana do treści wyprowadzonej informacji. Metoda może być stosowana w dialogowych procedurach obsługi baz danych.

1. WSTĘP

Wyszukiwanie i wyprowadzanie informacji ze zbiorów informacji w pamięci zewnętrznej maszyny cyfrowej jest jedną z najistotniejszych funkcji systemów zarządzania bazami danych, decydującą przede wszystkim o ich użyteczności. Operacje te są realizowane zazwyczaj przez zespół procedur programowych, wykorzystujących zdefiniowany dla danego systemu język manipulacji danymi, za pomocą którego określa się zespół żądań użytkownika dotyczących cech wyszukiwanej informacji [1]. Sprezycyzowanie takiego żądania powoduje zazwyczaj wyprowadzenie określonych treści w pewnym stałym formacie, w niezmiennym układzie graficznym.

Możliwość wpływania za pomocą zewnętrznego języka na zawartość i układ graficzny dokumentu grupującego wyszukaną informację wydaje się problemem szczególnie istotnym w systemach dialogowych, gdzie nierzadko selektywny wybór treści do wyprowadzenia decyduje nie tylko o przejrzystości dokumentu, lecz również o szybkości uzyskania odpowiedzi

^{*)} Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

[3, 5]. Sytuacja taka występuje np. w informatycznych systemach obsługi dydaktyki.

W niniejszej pracy podjęto próbę przedstawienia metody, która nie określając sposobu przeglądania zbioru w procesie wyszukiwania informacji, rozwija elementy parametryzacji i automatyzacji edycji dokumentów. Przedstawiono próbę formalnego opisu modelu systemu edycji dokumentów o zadanej postaci graficznej oraz podano odpowiednią metodę, która została z powodzeniem zastosowana i przetestowana w procedurach programowych informatycznego systemu Obsługi Procesu Nauczania DIALOS/MLK, zbudowanego w Centrum Obliczeniowym Politechniki Wrocławskiej.

2. SFORMUŁOWANIE PROBLEMU

Dany jest zbiór Z informacji, zapisanych w s rekordach logicznych jednego typu $z^1, z^2, \dots, z^s$. Każdy z rekordów $z^i, i = 1, 2, \dots, s$ zawiera n danych elementarnych $r_j^i, j = 1, 2, \dots, n$

$$z^i = \{r_1^i, r_2^i, \dots, r_n^i\}. \quad (1)$$

Dana elementarna r_j^i jest parą uporządkowaną

$$r_j^i = \langle v_j, w_j^i \rangle, \quad (2)$$

gdzie v_j jest nazwą danej elementarnej, w_j^i jest jej wartością, tzn. treścią informacji.

Przez R_j oznaczmy zbiór danych elementarnych o tej samej nazwie v_j , znajdujących się we wszystkich rekordach $z^1, z^2, \dots, z^s$

$$R_j = \{r_j^1, r_j^2, \dots, r_j^s\}. \quad (3)$$

Przyjmujemy, że dany jest system wyszukiwania informacji, który pozwala wybrać ze zbioru Z informacje żądane przez użytkownika. W pracy tej zajmujemy się metodą wyprowadzania żądanych informacji w postaci tabelarycznego dokumentu o określonej oddzielnie postaci graficznej.

3. PARAMETRIZACJA DOKUMENTU

W dokumencie zawierającym wyprowadzone informacje można wyróżnić treść, składającą się z wyprowadzonych informacji oraz układ graficzny dokumentu (formularz) organizujący tę treść i objaśniający ją w określony sposób (np. w postaci oddzielonych kreskami wierszy lub kolumn o dobranych odpowiednio nagłówkach). Postać formularza jest dostosowywana zazwyczaj do treści, która ma zostać umieszczona w odpowiednich jego rubrykach. Istnieje zatem potrzeba określenia maksymalnego formatu formularza na podstawie znajomości cech treści do wyprowadzenia. Przyjmujemy tu, że w celu wyprowadzenia wartości danych elementarnych, należących do R_j , należy określić postać graficzną kolumny (lub grupy kolumn) w formularzu, przez podanie sformalizowanych opisów poszczególnych, funkcjonalnie wyodrębnionych, fragmentów formularza tworzących kolumnę. Takim fragmentem może być np. nagłówek o ustalonej treści, nr kolumny, miejsce na umieszczenie treści informacji, linie oddzielające, itp. Fragment formularza, mający swój własny opis, jest traktowany jako najmniejsza, niepodzielna część dokumentu i jest nazywany w dalszej części modułem elementarnym formularza. W ogólnym przypadku moduł elementarny może zawierać fragmenty wielu wierszy dokumentu.

Formularz dokumentu w swym maksymalnym formacie powinien zawierać co najmniej tyle kolumn (lub grup kolumn), ile grup danych R_j zawiera zbiór informacji Z . W celu określenia postaci graficznej dokumentu wprowadza się zbiór F opisów modułów elementarnych, wchodzących w skład formularza. Zbiorowi temu nadaje się pewną strukturę, grupując w podzbiory $F_k \subset F$

$$F_k = \{f_k^1, f_k^2, \dots, f_k^m\}, \quad (4)$$

opisy modułów elementarnych f_k^i , $i = 1, 2, \dots, m$, tworzących kolejne kolumny (lub grupy kolumn), z których każda jest przeznaczona do przedstawienia informacji należących do grupy danych R_k , $k = 1, 2, \dots, n$. Zbiór modułów elementarnych, opisanych za pomocą podzbioru F_k , nazywamy modułem pionowym dokumentu. Wartość m określa liczbę modułów elementarnych, wyodrębnionych w pojedynczym module pionowym.

Za pomocą podzbiorów F_k , można przedstawić zbiór F w postaci

$$F = \bigcup_{k=1}^n F_k \quad (5)$$

Moduły elementarne, tworzące układ graficzny dokumentu, można podzielić na dwie klasy:

- moduły elementarne typu 0 - są to moduły o stałej postaci, nie zawierające pól przeznaczonych do wyprowadzenia treści informacji (np. linie oddzielające, nagłówki o stałej treści, symbole kolumn),

- moduły elementarne typu 1 - są to moduły zawierające pola przeznaczone do wyprowadzenia treści informacji.

Znajomość typów modułów elementarnych ułatwia interpretację opisów zawartych w zbiorze F . W tym celu wprowadza się zbiór P , którego elementy p_k^i odpowiadają elementom $f_k^i \in F$ i wyrażają typy modułów, opisanych za pomocą elementów f_k^i . Podobnie jak w zbiorze F , w zbiorze P można wydzielić podzbiory P_k

$$P_k = \{p_k^1, p_k^2, \dots, p_k^i, \dots, p_k^m\}, \quad (6)$$

określające typy poszczególnych elementów k -tego modułu pionowego dokumentu, opisanego za pomocą elementów zbioru F_k . Za pomocą podzbiorów P_k można przedstawić zbiór P jako

$$P = \bigcup_{k=1}^n P_k \quad (7)$$

W redagowanym dokumencie, oprócz modułów pionowych z treścią informacji, mogą wystąpić moduły pionowe o charakterze pomocniczym, nie zawierające wartości grup danych (np. pionowe linie, identyfikatory wierszy, itp.). Takie moduły pionowe składają się wyłącznie z modułów elementarnych typu 0.

Opisy poszczególnych modułów elementarnych $f_k^i \in F_k$, tworzą opis i -tego wiersza (lub i -tej grupy wierszy) w redagowanym dokumencie. Kolejne wiersze (lub ich grupy) są tworzone na podstawie podzbiorów opisów $F^i \subset F$,

$$F^i = \{f_1^i, f_2^i, \dots, f_k^i, \dots, f_n^i\} \quad (8)$$

i zawierają w ogólnym przypadku moduły elementarne różnych typów. Tworzą one kolejne moduły poziomą dokumentu. Należy podkreślić, że o wzajemnej kolejności kolumn w dokumencie decyduje porządek występowania opisów modułów elementarnych w zbiorze F^i .

Aby określić liczbę wierszy w dokumencie, wprowadza się zbiór G

$$G = \{g^1, g^2, \dots, g^i, \dots, g^m\}, \quad (9)$$

którego elementy g^1 są liczbami wierszy w kolejnych, i -tych modułach poziomych, opisanych za pomocą podzbiorów F^i . Wartości elementów g^1 , odpowiadające modułom poziomym, złożonym wyłącznie z modułów elementarnych typu 0, muszą być z góry określone. W odniesieniu do modułów poziomych, zawierających moduły elementarne typu 1, wartości $w(g^1)$ elementów g^1 są określane podczas tworzenia dokumentu na podstawie bieżących informacji o liczbie rekordów z treścią do wyprowadzenia. Przy założeniu, że treść pojedynczego rekordu ze zbioru informacji jest wyprowadzana w jednym wierszu dokumentu i zbiór informacji Z zawiera n rekordów, wartości elementów g^1 , odpowiadających modułom poziomym typu 1 muszą spełnić warunek

$$\sum_{i \in \{1, 2, \dots, m\}} w(g^1_i) = n. \quad (10)$$

Opisana metoda nie narzuca sposobu określania wartości wymienionych elementów g^1 w zależności od n , pozostawiając określenie "pionowych" rozmiarów poszczególnych modułów poziomych dokumentu potrzebom użytkownika.

Zbiory F , P oraz G określają układ graficzny dokumentu o maksymalnym formacie "poziomym" oraz w sposób parametryczny opisują jego format "pionowy".

4. FORMUŁOWANIE POLECEŃ UŻYTKOWNIKA

W omawianym procesie wyprowadzania informacji ze zbioru Z użytkownik formułuje polecenie, zawierające dwa elementy:

- pytanie A służące do wydzielenia z Z pewnego podzbioru Z^* , zawierającego odpowiedź na pytanie,
- żądanie B , określające postać formularza tabelarycznego dokumentu.

Pytanie A ma postać koniunkcji warunków elementarnych

$$A = a_{j_1} \wedge a_{j_2} \wedge \dots \wedge a_{j_k} \wedge \dots \wedge a_{j_p}, \quad p \leq n. \quad (11)$$

Warunek elementarny a_j jest określony jako para $a_j = \langle \delta_j, c_j \rangle$, w którym δ_j jest relacją, zdefiniowaną w systemie wyszukiwania informacji, i dotyczy wartości w_j^1 , $i \in \{1, 2, \dots, s\}$, danych elementarnych $r_j^1 \in R_j$, c_j jest natomiast stałą. Relacją δ_j , w odniesieniu do grupy danych R_j , zawierającej dane liczbowe, może być np. relacja po-

równania wartości liczbowych, w odniesieniu do grupy danych zawierającej teksty - relacja porównania ciągów znaków, itp. Można również powiedzieć, że relacja δ_j dotyczy j -tych danych elementarnych $r_j^i \in z^i$, $z^i \in Z$.

Podzbiór $Z^* \subseteq Z$ jest tworzony za pomocą systemu wyszukiwania informacji i zawiera rekordy z^i , $i \in \{1, 2, \dots, s\}$, w których dla każdego warunku elementarnego a_{jk} , $k = 1, 2, \dots, p$ zachodzą jednocześnie relacje

$$c_{jk} \delta_{jk} w_{jk}^1, \quad (12)$$

gdzie w_{jk}^1 oznacza wartość danej elementarnej $r_{jk}^i \in z^i$.

Żądanie B ma postać uporządkowanego zbioru

$$B = \{v_{i_1}, v_{i_2}, \dots, v_{i_j}, \dots, v_{i_q}\}, \quad q \leq n \quad (13)$$

utworzonego z nazw v_{i_j} danych elementarnych i_j -tych grup danych, $i_j \in \{1, 2, \dots, n\}$. Zbiór B służy do wydzielenia ze zbioru F uporządkowanego podzbioru F^* .

$$F^* = \{F_{i_1}, F_{i_2}, \dots, F_{i_j}, \dots, F_{i_q}\} \subseteq F, \quad (14)$$

którego kolejne elementy F_{i_j} mają postać (4) i tworzą opisy modułów pionowych dokumentu (kolumn), przeznaczonych do przedstawienia informacji należących do grup danych R_{i_j} , $j = 1, 2, \dots, q$, wskazanych za pomocą nazw $v_{i_j} \in B$. Kolejność elementów F_{i_j} podzbioru F^* wyznacza porządek występowania wybranych kolumn w dokumencie tabelarycznym. Opisy poszczególnych modułów elementarnych $f_{i_j}^k \in F_{i_j}$ tworzą opis k -tego wiersza (lub k -tej grupy wierszy) w dokumencie. Kolejne wiersze (lub ich grupy) są tworzone na podstawie podzbiorów opisów F^{*k}

$$F^{*k} = \{f_{i_1}^k, f_{i_2}^k, \dots, f_{i_j}^k, \dots, f_{i_q}^k\} \subset F^*, \quad k = 1, 2, \dots, m. \quad (15)$$

5. OPIS METODY

Zastosowanie metody automatycznego generowania tabelarycznego dokumentu wymaga wcześniejszego określenia zbioru informacji Z oraz zbiorów F, P i G. Zbiory te wyznaczają klasę dokumentów, które mogą zostać utworzone na polecenie użytkownika.

W celu wygenerowania dokumentu, przedstawiającego wybraną informację w określonym przez użytkownika układzie graficznym, należy wykonać następujące czynności:

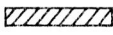
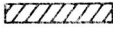
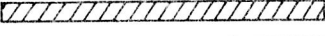
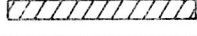
- 1⁰ Wczytać pytanie $A = a_{j1} \wedge a_{j2} \wedge \dots \wedge a_{jp}$. Wykorzystując posiadany system wyszukiwania informacji utworzyć podzbiór $Z^* \subseteq Z$. Jeśli zbiór Z^* jest pusty - wyprowadzić komunikat o braku odpowiednich informacji w Z , przejść do 9⁰. W przeciwnym przypadku wczytać żądanie $B = \{v_{i1}, v_{i2}, \dots, v_{it}, \dots, v_{iq}\}$. Na podstawie elementów v_{it} , $t = 1, 2, \dots, q$ utworzyć podzbiór $F^* \subseteq F$. Przejść do utworzenia pierwszego modułu poziomego, występującego w dokumencie ($k \leftarrow 1$).
- 2⁰ Na podstawie zbioru F^* utworzyć podzbiór $F^{*k} \subset F^*$. Przejść do drukowania pierwszego wiersza ($j \leftarrow 1$) w k -tym module poziomym dokumentu.
- 3⁰ Zbudować opis j -tego wiersza w k -tym module poziomym na podstawie F^{*k} .
- 4⁰ Jeśli $p_i^k = 0$ dla każdego $i \in \{i_1, i_2, \dots, i_q\}$ - przejść do pktu 5⁰. W przeciwnym wypadku przejść do pktu 6⁰.
- 5⁰ Utworzyć wiersz dokumentu na podstawie opisu zbudowanego dla j -tego wiersza w k -tym module poziomym. Przejść do pktu 7⁰.
- 6⁰ Pobrać ze zbioru Z^* pierwszy dostępny rekord z^1 . Na podstawie B oraz z^1 zbudować rekord $x^1 = \{r_{i1}^1, r_{i2}^1, \dots, r_{it}^1, \dots, r_{iq}^1\} \subseteq z^1$ taki, że $r_{it}^1 = \langle v_{it}, w_{it}^1 \rangle$, $t = 1, 2, \dots, q$. Utworzyć wiersz dokumentu na podstawie wartości w_{it}^1 kolejnych danych elementarnych $r_{it}^1 \in x^1$ na podstawie opisu utworzonego dla j -tego wiersza w k -tym module poziomym. Usunąć rekord z^1 ze zbioru Z^* ($Z^* \leftarrow Z^* \setminus z^1$).
- 7⁰ Jeśli j -ty wiersz był ostatnim wierszem w k -tym module poziomym (tzn. jeśli $j = g^k \in G$) - przejść do pktu 8⁰. W przeciwnym przypadku: $j \leftarrow j+1$, przejść do pktu 3⁰.
- 8⁰ Jeśli k -ty moduł poziomy był ostatnim modulem dokumentu - przejść do pktu 9⁰. W przeciwnym wypadku: $k \leftarrow k+1$, przejść do pktu 2⁰.
- 9⁰ Koniec drukowania tabelarycznego dokumentu.

6. PRZYKŁAD DZIAŁANIA METODY

Aby zilustrować metodę opracowano podany niżej przykład.

Zbiór informacji Z zawiera s rekordów. Każdy rekord ma 4 dane elementarne $r_i^j = \langle v_i^j, w_i^j \rangle$ o następującej strukturze:

- $$\begin{aligned}
 r_1^j &= \langle v_1: \text{SYMBOL}, w_1^j: \text{"tekst 8-znakowy"} \rangle, \\
 r_2^j &= \langle v_2: \text{NUMER}, w_2^j: \text{"liczba naturalna z przedziału [1, 99999]} \rangle, \\
 r_3^j &= \langle v_3: \text{NAZWA}, w_3^j: \text{"tekst 24-znakowy"} \rangle, \\
 r_4^j &= \langle v_4: \text{CENA}, w_4^j: \text{"liczba rzeczywista z przedziału [0, 10^9]} \rangle, \\
 j &= 1, 2, \dots, s.
 \end{aligned}$$

	F_1		F_2		F_3		F_4		F_5
F_1	SYMBOL		NUMER		NAZWA		CENA		
F_2									
F_3									
F_4	:	:	:	:	:	:	:	:	:
F_5									

Rys.1. Założony w przykładzie maksymalny układ graficzny dokumentu (zakresowane pola przeznaczone są do wyprowadzenia treści informacji)

Fig.1. Maximum graphical layout of a document assumed in the example (dashed area is designed to contain the information extracted)

Niech założony, maksymalny układ graficzny dokumentu będzie taki, jak przedstawiono na rysunku 1. Dokonujemy podziału dokumentu na moduły pionowe oraz poziome (rys.1). Budujemy zbiory F , P oraz G . Postacie elementów $f_j^i \in F$ w języku programowania Fortran 1900 przedstawiono w tabeli 1.

T a b e l a 1

Postacie elementów $f_j^i \in F$

$i \backslash j$	1	2	3	4	5
1	11(1H-)	8(1H-)	27(1H-)	17(1H-)	1H-
2	11H! SYMBOL	8H! NUMER	1H!,10X,5HNAZWA,11X	1H!,6X,4HCENA,6X	1H!
3	1H!,10(1H-)	1H!,7(1H-)	1H!,26(1H-)	1H!,16(1H-)	1H!
4	2H!,A8,1X	1H!,16,1X	2H!,3A8,1X	1H!,E15.6,1X	1H!
5	11(1H-)	8(1H-)	27(1H-)	17(1H-)	1H-

Zbiór P ma postać:

$$\begin{aligned}
 p_1^1 &= 0, & p_2^1 &= 0, & p_3^1 &= 0, & p_4^1 &= 0, & p_5^1 &= 0, \\
 p_1^2 &= 0, & p_2^2 &= 0, & p_3^2 &= 0, & p_4^2 &= 0, & p_5^2 &= 0, \\
 p_1^3 &= 0, & p_2^3 &= 0, & p_3^3 &= 0, & p_4^3 &= 0, & p_5^3 &= 0, \\
 p_1^4 &= 1, & p_2^4 &= 1, & p_3^4 &= 1, & p_4^4 &= 1, & p_5^4 &= 0, \\
 p_1^5 &= 0, & p_2^5 &= 0, & p_3^5 &= 0, & p_4^5 &= 0, & p_5^5 &= 0,
 \end{aligned}$$

Zbiór G ma postać:

$$g_1 = 1, \quad g_2 = 1, \quad g_3 = 1, \quad g_4 = r, \quad g_5 = 1,$$

gdzie r ma przyjmować wartość równą liczbie rekordów w zbiorze Z^* .

1° Pytanie $A = a_1 \wedge a_4$, gdzie

$a_1 = \langle \delta_1: "c_1 \text{ identyczne z } w_1^j", c_1: \text{MERCEDES} \rangle$,

$a_4 = \langle \delta_4: "c_4 \geq w_4^j", c_4: 1500000 \rangle$, $j = 1, 2, \dots, s$.

Niech utworzony zbiór Z^* zawiera 7 rekordów. Zbiór G przyjmuje więc postać: $G = \{1, 1, 1, 7, 1\}$.

Żądanie $B = \{\text{NAZWA, SYMBOL, CENA}\} = \{v_3, v_1, v_4\}$. Postacie elementów $f_j^1 \in F^* \subset F$ przedstawiono w tabeli 2. Elementy $f_5^1 \in F^*$, $i = 1, 2, \dots, 5$ tworzą moduł pionowy o charakterze pomocniczym, jakim jest linia pionowa.

T a b e l a 2

Postacie elementów $f_j^1 \in F^* \subset F$

$i \backslash j$	3	1	4	5
1	27(1H-)	11(1H-)	17(1H-)	1H!
2	1H!, 10X, 5HNAZWA, 11X	11H! SYMBOL	1H!, 6X, 4HCENA, 6X	1H!
3	1H!, 26(1H/)	1H!, 10(1H-)	1H!, 16(1H-)	1H!
4	2H! ,3A8, 1X	2H! ,A8, 1X	1H!, E15.6, 1X	1H!
5	27(1H-)	11(1H-)	17(1H-)	1H-

Przechodzimy do utworzenia pierwszego modułu poziomego, występującego w dokumencie ($k \leftarrow 1$).

2° $F^{*1} = \{f_3^1 \leftrightarrow 27(1H-), f_1^1 \leftrightarrow 11(1H-), f_4^1 \leftrightarrow 17(1H-), f_5^1 \leftrightarrow 1H-\}$

Przejsć do drukowania pierwszego wiersza dokumentu ($j \leftarrow 1$) w 1. module poziomym.

3° Opis 1. wiersza w 1. module ma postać:

(1X, 27(1H-), 11(1H-), 17(1H-), 1H-).

4° Warunek $p_1^1 = 0$ jest spełniony dla każdego $i \in \{3, 1, 4, 5\}$.

5° Wydrukować wiersz (na rys.2 oznaczony numerem 1).

7° Warunek $j = g^1 \in G$ jest spełniony.

8° Pierwszy moduł poziomy nie był ostatnim modulem poziomym dokumentu, więc $k \leftarrow k+1$ i powrót do pktu 2°.

2° $F^{*2} = \{f_3^2 \leftrightarrow 1H!, 10X, 5HNAZWA, 11X, f_1^2 \leftrightarrow 11H! \text{ SYMBOL}, f_4^2 \leftrightarrow 1H!, 6X, 4HCENA, 6X, f_5^2 \leftrightarrow 1H!\}$

Przejsć do drukowania pierwszego wiersza dokumentu ($j \leftarrow 1$) w 2. module poziomym, itd.

Po wydrukowaniu wierszy o numerach 1, 2 i 3 (rys.2), $k \leftarrow 4$ i kolejny powrót do pktu 2°.

2° $F^{*4} = \{f_3^4 \leftrightarrow 2H! ,3A8, 1X, f_1^4 \leftrightarrow 2H! ,A8, 1X, f_4^4 \leftrightarrow 1H!, E15.6, 1X, f_5^4 \leftrightarrow 1H!\}$

Przejsć do drukowania pierwszego wiersza dokumentu ($j \leftarrow 1$) w 4. module poziomym.

3° Opis 1. wiersza w 4. module poziomym ma postać:

(1X,2H! ,3A8,1X,2H! ,A8,1X,1H! ,E15.7,1X,1H!)

4° Warunek $p_1^4 = 0$ dla $i \in \{3, 1, 4, 5\}$ nie jest spełniony.

6° Niech $z^1 \in Z^*$ ma postać:

$r_1^1 = \langle \text{SYMBOL}, \text{MERCEDES} \rangle$

$r_2^1 = \langle \text{NUMER}, 7931 \rangle$

$r_3^1 = \langle \text{NAZWA}, \text{KABRIOLET 280 SEL (1977)} \rangle$

$r_4^1 = \langle \text{CENA}, 1200000 \rangle$

Rekord $x^1 = \{r_3^1, r_1^1, r_4^1\}$. Wydrukować wiersz (na rys.2 oznaczony numerem 4). Usunąć rekord z^1 ze zbioru Z^* .

7° Warunek $j = g^4 \in G$ nie jest spełniony, $j \leftarrow 2$ i powrót do pktu 3°.

3° Opis 2. wiersza w 4. module poziomym ma postać identyczną jak poprzednio.

4° Warunek $p_1^4 = 0$ dla $i \in \{3, 1, 4, 5\}$ nie jest spełniony. Powtarzając czynności 3, 4, 6, 7 drukujemy dalsze wiersze dokumentu (5÷10 na rys.2), $k \leftarrow 5$ i kolejny powrót do pktu 2°.

2° $F^5 = \{f_3^5 \leftarrow 27(1H-), f_1^5 \leftarrow 11(1H-),$

$f_4^5 \leftarrow 17(1H-), f_5^5 \leftarrow 1H-\}$

Przejsć do drukowania pierwszego wiersza ($j \leftarrow 1$) w 5. module poziomym.

3° Opis 1. wiersza w 5. module ma postać:

(1X,27(1H-),11(1H-),17(1H-),1H-)

4° Warunek $p_1^5 = 0$ jest spełniony dla każdego $i \in \{3, 1, 4, 5\}$.

5° Wydrukować wiersz (na rys.2 oznaczony numerem 11).

7° Warunek $j = g^5 \in G$ jest spełniony.

8° Piąty moduł poziomy był ostatnim modulem poziomym dokumentu.

9° Koniec drukowania dokumentu (rys.2).

NAZWA	SYMBOL	CENA	
KABRIOLET 280 SEL (1977)	MERCEDES	0.120000E 07	- 1
_____	_____	_____	- 2
_____	_____	_____	- 3
_____	_____	_____	- 4
_____	_____	_____	- 5
_____	_____	_____	- 6
_____	_____	_____	- 7
_____	_____	_____	- 8
_____	_____	_____	- 9
_____	_____	_____	- 10
_____	_____	_____	- 11

Rys.2. Utworzony w przykładzie tabelaryczny dokument
Fig.2. Tabulated document generated in the example

7. UWAGI KOŃCOWE

Przedstawiona metoda może być wykorzystana w wersji programowej jako uniwersalna procedura wyprowadzania informacji z baz danych systemów informatycznych. Wówczas zbiory Z, F, P oraz G są wprowadzane jako dane parametryczne procedury, wyznaczając klasę dokumentów, jakie można uzyskać w dowolnym momencie procesu przetwarzania. Zmiana treści zbiorów Z, F, P oraz G może być dokonywana praktycznie w dowolnym momencie na życzenie użytkownika i nie wymaga naruszania struktury programu.

Opisana metoda została zastosowana w konwersacyjnych procedurach wyprowadzania informacji z bazy danych informatycznego systemu Obsługi Procesu Nauczania DIALOS/MLK, eksploatowanych w trybie konwersacyjnym pod kontrolą systemów George 3 i Minimop. Aby ułatwić obsługę procedury, zdefiniowano prosty język komunikacji użytkownika z systemem, umożliwiający wygodne wprowadzanie poleceń.

Możliwość wpływania na rozmiar "poziomy" dokumentu (długość poszczególnych wierszy) okazała się bardzo istotna - zwłaszcza ze względu na szybkość uzyskiwania informacji, pozwala ona na znaczne skrócenie czasu przetwarzania.

Istotną zaletą stosowania opisów $f_j^i \in F$ wydzielonych fragmentów dokumentu (modułów elementarnych) jest możliwość dokonywania dekompozycji, uwzględniającej powtarzalność opisów wyodrębnionych modułów, co umożliwia minimalizację zajętości pamięci niezbędnej do przechowywania zbioru opisów F.

Eksploatacja procedury programowej, wykorzystującej opisaną metodę, umożliwia uzyskanie dokumentów zawierających treść tylko żądanych informacji, co czyni wydruki bardziej czytelnymi i komunikatywnymi. Jest to również ważne wówczas, gdy chodzi o celowe pomijanie w dokumencie pewnych danych, zawartych w rekordach zawierających odpowiedź na pytanie użytkownika.

LITERATURA

- [1] C a g a n C., Data Management System, Los Angeles 1973, Melville Publishing Company.
- [2] J u d d D.R., Use of Files, London and American Elsevier Inc., New York 1973.
- [3] M a r t i n J., Dialog człowieka z maszyną cyfrową, Warszawa 1976, WNT.

- [4] S p r o w l s R.C., Management Data Bases, Santa Barbara 1978, Wiley/Hamilton.
[5] Y o u r d o n E., Projektowanie systemów o działaniu bezpośrednim, Warszawa 1976, WNT.

Praca wpłynęła do Redakcji 10.12.1978 r.
Po poprawieniu 19.02.1980 r.

DOCUMENT GENERATION METHOD FOR USE IN THE PROCESS OF DATA EXTRACTION FROM DATA BASES

A method of automatic generation of documents during information extraction from data base of information retrieval systems, as well as formal description of information output module is presented. This method makes it possible to extract selected data groups, as well as their representation in the form of tabulated documents having separately specified graphical layout. The method can be applied in the conversational procedures of informatic system data base servicing.

Verified by R. Wiernik

МЕТОД ГЕНЕРИРОВАНИЯ ДОКУМЕНТОВ В ПРОЦЕССЕ ВЫВОДА ИНФОРМАЦИИ ИЗ БАЗ ДАННЫХ ИНФОРМАТИЧЕСКИХ СИСТЕМ

В работе представлен метод автоматического редактирования документов в процессе вывода информации из баз данных информатических систем. Приводится также формальное описание модуля вывода информации. Описанный метод дает возможность вывода значений избранных групп данных и представления их в виде табулированных документов. Вид этих документов, приспособленный к содержанию выводимых данных, определяется отдельно. Метод может найти применение в диалоговых процедурах обслуживания баз данных информатических систем.

Прогерил Чеслав Штуковский



BIBLIOTEKA
Politechniki Wrocławskiej

A 4947 I/1

1980 m 1/1

Subscription should be sent (at any time of the year) to:

„Ars Polona”

Krakowskie Przedmieście 7, 00-068 Warszawa
or OR PAN, 00-901 Warszawa, PKiN, POLAND

Bank account number: NBP, VIII OM Nr 1550-6-81574 Warszawa/Poland

Wydawnictwa Politechniki Wrocławskiej

ma stale na składzie Księgarnia Wr 49

Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

oraz Wojewódzka Księgarnia Techniczna

ul. Świdnicka 8, 50-067 Wrocław