ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1980

Seria: INFORMATYKA z. 1 Nr kol. 653

Jan BRUSKI
Bogdan GRUDZIŃSKI
Ośrodek Elektronicznej Techniki Obliczeniowej
Politechnika Śląska

KONCEPCJA SYSTEMU SŁUŻĄCEGO DO PRZETWARZANIA STRUKTUR DANYCH W JĘZYKU ALGOL 1900

Streszczenie. W pracy przedstawiono genezę powstania systemu przetwarzania struktur danych w języku ALGOL 1900 i uzasadnienie wyboru języka, którego potrzebom służy system. Szczegółowo omówiono organizację pamięci wykorzystywaną przez system oraz budowę pól pamięci w tej organizacji. Pokazano również sposoby przechowywania informacji elementarnych przyjętych jako standardowe dla systemu.

Najbardziej obecnie popularne języki programowania, jakimi są FORTRAN i ALGOL 60, posiadają jedną zasadniczą wadę. Nie zezwalają mianowicie na tworzenie i używanie w programach nieregularnych obiektów, zwanych strukturami danych. Niektóre później zdefiniowane, algorytmiczne języki programowania (np. PASCAL czy ALGOL 68) zawierają już konstrukcje gramatyczne i semantyczne pozwalające na korzystanie z tego typu obiektów. Jest to bardzo cenna zaleta, gdyż wiele najnowszych algorytmów, zwłaszcza w dziedzinie tzw. przetwarzania danych oraz w procesach przetwarzania symboli (np. w celu translacji języków), wykorzystuje takie właśnie nieregularne obiekty. Wydawałoby się więc, że najlepiej byłoby zarzucić stosowanie starszych języków programowania, a korzystać z języków zezwalających na przetwarzanie struktur danych [6], [8]. Nie jest to jednak proste. Wymaga bowiem istnienia translatorów określonych języków dla stosowanych maszyn cyfrowych, a zbudowanie dowolnego translatora jest zawsze zadaniem niezmiernie trudnym i nadzwyczaj pracochłonnym. Znacznie łatwiej jest dobudować do języka, którego translatorem dysponuje już maszyna cyfrowa, odpowiednie mechanizmy zezwalające na rozszerzenie możliwości języka o przetwarzanie struktur danych.

Tego rodzaju pracę podjęto w Politechnice Śląskiej w odniesieniu do maszyn cyfrowych ODRA serii 1300, które są obecnie najbardziej w Polsce popularne, a dla których nie istnieje implementacja języka służącego do przetwarzania struktur danych. Na wstępie zainicjowanych prac powstał jednak dylemat wyboru: jaki język należy uzupełnić mechanizmami przetwarzania struktur danych?

Język COBOL posiada pewne niewielkie możliwości przetwarzania nieregularnych obiektów, a więc wystarczyłoby nieco go rozszerzyć. Jednak język COBOL jest zdecydowanie wyspecjalizowany i zakres jego zastosowań jest stanowczo za mały.

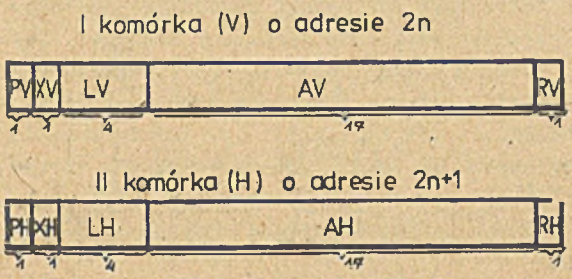
Język FORTRAN 1900 jest stosunkowo uniwersalnym językiem programowania, posiada rozbudowany system wejścia i wyjścia i dysponuje bardzo sprawnym kompilatorem. Język ten ma jednak bardzo poważną wadę: jest ogromnie "sztywny", nieelastyczny. Język FORTRAN jest doskonałym narzędziem do programowania prostych zadań technicznych, natomiast jest mało wygodny, gdy trzeba programować złożone problemy wymagające wiele inwencji i polotu. Zwłaszcza niemożność stosowania rekursji w programach w FORTRAN-ie nieraz znacznie utrudnia, a czasem wręcz uniemożliwia programowanie w tym języku niektórych algorytmów.

Znacznie doskonalszy pod tym względem jest język ALGOL. Programy w tym języku mogą być opracowane zgodnie z regułami programowania systematycznego, ale równocześnie mogą być pełne wdzięku i elegancji. Skomplikowane algorytmy przetwarzania można w nim programować prosto i przejrzyście.

Możliwość stosowania tablic dynamicznych i procedur rekursywnych otwiera znacznie szerszy zakres zastosowań aniżeli istnieje dla FORTRAN-u. Blokowa struktura programów w ALGOL-u umożliwia dynamiczne przydziały pamięci, a więc oszczędną gospodarkę pamięcią. Wreszcie nie bez znaczenia jest to, że maszyny cyfrowe ODRA serii 1300 są zaopatrzone w stosunkowo sprawny kompilator. Z wymienionych względów zdecydowano, że właśnie język ALGOL 1900 (wersja dla maszyn cyfrowych ODRA serii 1300) zostanie rozbudowany o możliwości przetwarzania struktur danych. Przyjęto przy tym założenie, że zadanie to zostanie wykonane w sposób proceduralny w oparciu o procedury w języku ALGOL oraz dzięki możliwości łączenia języków - o procedury w języku PLAN (język adresów symbolicznych maszyn cyfrowych ODRA 1300) [9]. Dzięki temu nie ulega zmianie zarówno gramatyka, jak i semantyka języka, a system służący do przetwarzania struktur danych buduje się na tle podstawowej wersji języka.

Pamięć operacyjna maszyn cyfrowych ODRA serii 1300 jest jednorodna i posiada organizację liniową. Stosowanie ogólnych struktur danych wymaga strukturalnej organizacji pamięci [4]. Organizację taką tworzy sam system przetwarzający struktury danych. Ponieważ jest to system proceduralny, nie zakłóca więc ani nie zmienia konstrukcji programów. Program opracowany w języku ALGOL 1900 i wykorzystujący system do przetwarzania struktur danych jest tłumaczony za pomocą tego samego kompilatora jak w przypadku, gdy nie stosuje się struktur danych. Także sposób wykonywania programów nie ulega zmianie. Programy w swych zasadniczych czynnościach korzystają więc z liniowej organizacji pamięci. Dopiero w momencie inicjowania pracy systemu powstaje (częściowo) strukturalna organizacja pamięci i wszystkie struktury danych buduje się w obszarze pamięci tak zorganizowanej. Dzięki temu można uzyskać znaczną szybkość przetwarzania, gdyż z organizacji strukturalnej korzysta się tylko w tych przypadkach, gdy jest to konieczne.

W trakcie przetwarzania programu zawierającego instrukcje procedur systemu w pamięci maszyny cyfrowej tworzą się dwa rozłączne obszary - jeden o organizacji liniowej, drugi o organizacji strukturalnej. Wielkości tych obszarów bezustannie się zmieniają. O wielkości obszaru zorganizowanego liniowo decydują przydziały pamięci w momentach otwierania nowych bloków w programie, natomiast wielkość obszaru o organizacji strukturalnej jest sterowana przez procedury systemu [5].



Rys. 1. Graficzne przedstawienie podstawowego pola pamięci

System przetwarzania struktur danych tworzy strukturalną organizację pamięci w oparciu o teorię struktur binarnych, która została przedstawiona w pracach [2], [3]. Za pomocą struktur binarnych można przedstawiać dowolne obiekty systemu ogólnych struktur danych.

W pamięci maszyny cyfrowej buduje się tzw. pola, które następnie z sobą się wiąże. Przyjęto przy tym sposób określony w pracy [4]. Wszystkie pola mają jednakową budowę, pozycje informacyjne zawarte w polach mają zawsze stałe położenie i długość. Nieco inną tylko budowę mają pola służące do przechowywania wartości atomów. Graficzny obraz podstawowego pola pamięci służącego do konstrukcji struktur binarnych pokazano na rys. 1.

Każde podstawowe pole pamięci składa się zawsze z dwu kolejnych miejsc (komórek) 24-bitowych. Dostęp do dowolnego pola realizuje się przez podanie adresu pierwszej komórki wchodzącej w skład tego pola. Przyjęto, że adres ten jest zawsze liczbą parzystą. Obydwie komórki wchodzące w skład pola są podzielone na pewną ilość części. Położenie poszczególnych części jest niezmienne i przedstawia się następująco:

I komórka (V)

PV - b_0	(1 bit)
XV - b_1	(1 bit)
LV - $b_2 \dots b_5$	(4 bity)
AV - $b_6 \dots b_{22}$	(17 bitów)
RV - b_{23}	(1 bit)

II komórka (H)

PH - b_0	(1 bit)
XH - b_1	(1 bit)
LH - $b_2 \dots b_5$	(4 bity)
AH - $b_6 \dots b_{22}$	(17 bitów)
RH - b_{23}	(1 bit)

Podstawowe znaczenie mają zawartości części AV i AH. Przechowywane w nich informacje uzupełnia się od strony mniej znaczącej dodatkowym bitem o wartości równej 0 i wtedy interpretuje się je jako adresy dowiązań do innych pól pamięci. Uzyskane w ten sposób 18-bitowe adresy, będące zapisem dwójkowym liczb parzystych, pozwalają na kontaktowanie się z polami rozmieszczonymi w obszarze całej pamięci operacyjnej o pojemności do 256 K słów. Adres z części AV odpowiada selektorowi binarnemu v, zaś adres z części AH - selektorowi binarnemu h [2].

Za pomocą adresu w części AH tworzy się dowiązania do innych podstawowych pól stanowiących elementy składowe szkieletu struktury binarnej. Jeżeli istnieje takie dowiązanie, wtedy w części PH pola jest wartość 0 i zawartość części AH, po uzupełnieniu bitem $b_{23} = 0$, interpretuje się jako adres tego dowiązania. Gdy natomiast nie istnieje już dowiązanie do następnego pola, wówczas wartość części PH jest równa 1. Wartość tę interpretuje się jako symbol NIL [4], co oznacza, że do rozpatrywanego pola jest dołączona pusta struktura binarna Ω [2]. Zawartość części AH nie stanowi wtedy adresu dowiązania.

Za pomocą adresu w części AV można wiązać zarówno podstawowe pola wchodzące w skład szkieletu struktury, jak i te, które zawierają wartości atomów. Rozróżnienie rodzaju wiązań odbywa się za pomocą wartości w części PV pola. Gdy rozpatrywane pole jest związane z innym podstawowym polem pamięci, wtedy zawartość części PV jest równa 1. Gdy zaś pole jest związane z jakimś polem zawierającym wartość atomu, wówczas w części PV jest zapisana wartość 0. W obydwu wypadkach zawartość części AV służy do określenia adresu wiązania. Jeżeli zawartość części PV jest równa 1, wtedy adres zapisany w części AV musi być różny od zera. Zawartość AV równa 0 może wystąpić tylko wówczas, gdy w części PV jest wartość 0. Interpretuje się to w ten sposób, że do rozpatrywanego pola jest dołączony atom pusty Ω [2].

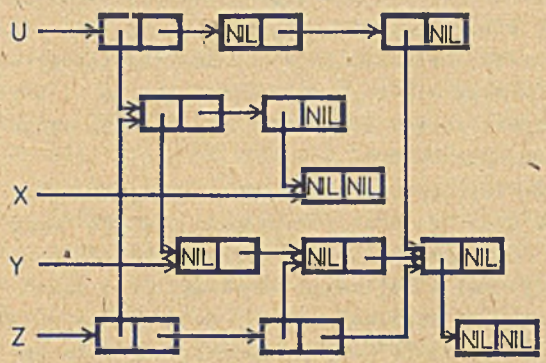
Identyfikacja typu pola, a więc określenie, że dane pole jest podstawowe a nie takie, które zawiera wartość atomu, następuje na podstawie badania wskaźnika XV. Jego wartość powinna być równa 0.

Jeżeli rozpatrywane pole jest bezpośrednio związane z polem zawierającym wartość atomu, a więc, gdy PV = 0, wówczas wykorzystuje się część LH do zapisu informacji służącej do specyfikacji tego atomu. Część LH składa się z 4 bitów, a więc pozwala na zapis 16 kombinacji binarnych. Znaczenie niektórych z nich jest ustalone a priori, znaczenia pozostałych mogą być określane programowo przez użytkowników systemu. Standardowo zostały przyjęte następujące oznaczenia atomów:

0001 1 - atom typu INTEGER (zajmujący 1 pole)
0010 2 - atom typu BOOLEAN (1 pole)
0011 3 - atom typu REAL (2 pola)
0100 4 - atom typu STRING (od 1 do 50 pól)
0101 5 - atom typu CHARACTER (1 pole)

1100 12 - atom typu ADDRESS (1 pole)
1101 13 - atom typu COMMAND (1 pole)
1110 14 - atom typu SEQUENCE (dowolna liczba pól)
1111 15 - atom pusty (nie zajmuje żadnego pola).

Jeżeli wartość wskaźnika PV jest równa 1, wtedy zawartość części LH jest nie wykorzystana i równa zero (przewiduje się wykorzystanie części LH w dalszej fazie rozbudowy systemu).



Rys. 2. Ilustracja sposobu wiązania pól pamięci w wypadku stosowania wspólnych części kilku struktur

W celu jak najlepszego wykorzystania pamięci przyjęto założenie, że każda dowolna struktura binarna lub jej fragment może stanowić część wspólną pewnej liczby innych struktur. Mówiąc inaczej, do każdego pola pamięci mogą istnieć dowiązania nie tylko z jednego pola ale z kilku pól (najwyżej piętnastu). Dzięki temu, jeżeli kilka struktur binarnych począwszy od pewnego miejsca zawiera identyczne fragmenty, wtedy zamiast budować w pamięci kilkakrotnie wiązania pól im odpowiadające, wystarczy to zrobić tylko jeden raz i we właściwych polach rozpatrywanych struktur umieścić ten sam adres dowiązania [1].

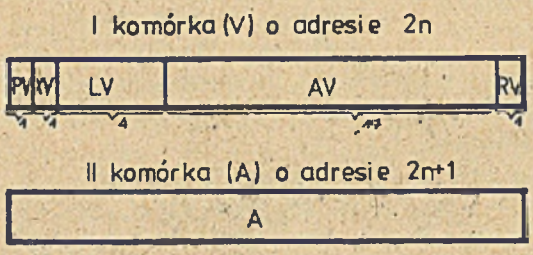
Na rys. 2 został pokazany prosty przykład ilustrujący tę zasadę.

Taki sposób wiązania pól powoduje jednak pewne problemy wtedy, gdy likwiduje się w pamięci struktury zawierające wspólne części. Wspólne części kilku struktur muszą pozostać nie naruszone tak długo, dopóki nie nastąpi skasowanie wszystkich struktur, które je wykorzystują. Wynika stąd, że musi istnieć stała kontrola pól pamięci wchodzących w skład poszczególnych struktur [7]. Celem jej jest sprawdzenie, czy poszczególne pola należą do jednej czy też wielu struktur. Kontroli tej dokonuje się wykorzystując część LV pola. W każdym takim polu, które jest tylko jednokrotnie dowiązane, w części LV zapisana jest wartość zero. Każde kolejne dowiązanie tego pola do następnej struktury powoduje zwiększenie zawartości części LV o 1. Część LV pola stanowi więc licznik ilości dodatkowych dowiązań. Należy tutaj zwrócić uwagę na to, że zawartość tego licznika zmienia się tylko w tym polu, którego bezpośrednio dotyczy dowiązanie. Zawartości części LV w następnych polach części wspólnej struktur nie ulegają zmianie.

Likwidacja dowolnej struktury polega na przekazywaniu wszystkich składających się na nią pól do tzw. listy nieużytków (pola z tej listy mogą

być następnie wykorzystane do tworzenia innych struktur). Czynność tę wykonuje się począwszy od naczelnego pola struktury i w taki sposób, by nie pominąć żadnego pola należącego do kasowanej struktury. Aby jednak nie nastąpiło również przekazanie do listy nieużytków części wspólnej czynnych jeszcze struktur, kontroluje się zawartość części LV każdego pola, które bierze się pod uwagę. Gdy zawartość ta jest równa 0, wtedy pole oddaje się do listy nieużytków i rozpatruje się następne pole wchodzące w skład kasowanej struktury. Gdy natomiast stan licznika ilości dodatkowych dowiązań pola jest różny od zera, wtedy zmniejsza się ten stan o 1, a pola tego i wszystkich z nim związanych nie przekazuje się do listy nieużytków. Proces likwidacji struktury kończy się dopiero wtedy, gdy do listy nieużytków zostaną przekazane wszystkie pola do niej należące a nie wchodzące w skład wspólnych części innych struktur.

Zezwalając na stosowanie wspólnych części wielu struktur, trzeba się liczyć z pewnym efektem ubocznym. Otóż związane z sobą pola pamięci mogą nie tylko stanowić część wspólną niezależnych struktur, ale także mogą one być dowiązane do różnych pól jednej i tej samej struktury [1]. Jeżeli pola, od których istnieją dowiązania do wspólnej części, są od siebie niezależne (nie są bezpośrednio ani pośrednio z sobą związane), to taka sytuacja jest prawidłowa i nie powoduje żadnych problemów. Gorzej natomiast, gdy istnieją powiązania tych pól. (Ma to miejsce np. wtedy, gdy w części wspólnej struktury istnieje pole, do którego jest dowiązane inne pole będące nadrzędnym w stosunku do rozpatrywanej części wspólnej). Powstają wtedy bowiem tzw. struktury cykliczne, co powoduje znaczne kłopoty realizacyjne dla systemu przetwarzania struktur danych. Rozwiązanie zagadnienia przetwarzania struktur cyklicznych jest przewidziane jako dalszy etap budowy systemu. W pierwszej fazie opracowania systemu przewidziano tylko wykrywanie faktu powstania struktury cyklicznej i wywołanie odpowiedniej sygnalizacji. Dla potrzeb systemu w warunkach powstawania struktur cyklicznych przewidziane są trzy wskaźniki w polu, oznaczone symbolami: XH, RV i RH. Zastosowanie tych wskaźników nie będzie tutaj omawiane.



Rys. 3. Graficzne przedstawienie pola pamięci służącego do przechowywania wartości atomu

Podstawowe pola pamięci, których budowa została wyżej omówiona służą do tworzenia szkieletu struktur binarnych w pamięci maszyny. Do przechowywania wartości atomów stosuje się pola nieco innego typu. Graficzny obraz takiego pola przedstawia rys. 3.

Pola służące do przechowywania atomów, podobnie jak pola podstawowe, są zawsze złożone z dwóch komórek. Pierwsza z nich posiada identyczną budowę jak pierwsza komórka (V) podstawowego pola pamięci. Druga natomiast (A) nie jest podzielona na żadne mniejsze części. Komórka ta służy bowiem do zapisu w niej wartości atomu lub jego części.

Atom może być umieszczony w jednym tylko polu (w komórce A) lub w wielu związanych z sobą polach. Jeżeli wewnętrzna reprezentacja atomu mieści się w 24 bitach jednej komórki pamięci, wówczas wskaźnik PV pola winien mieć wartość 1. Zawartość części LV jest wtedy równa zero.

Jeżeli wewnętrzna reprezentacja atomu ma większą długość niż 24 bity, wówczas do jej zapamiętania potrzeba większej liczby pól pamięci. Pola te tworzą tzw. listę atomu. Wiązanie pól listy atomu odbywa się za pomocą adresów, które określa się na podstawie zawartości części AV po uzupełnieniu jej bitem $b_{23} = 0$. Wartość w części AV końcowego pola listy atomu jest równa zero. Określenie, czy dane pole jest końcowym polem listy atomu, następuje na podstawie badania wskaźnika PV. Dla pola końcowego listy wartość tego wskaźnika jest równa 1. W pozostałych polach listy atomu wartość w części PV jest równa zero.

W części XV każdego pola należącego do listy atomu jest zawsze umieszczona wartość 1. Wartość ta służy do rozróżniania pól zawierających atomy od podstawowych pól pamięci.

Część LV, podobnie jak w przypadku pól podstawowych, stanowi licznik ilości dodatkowych dowiązań. Każda lista atomu (a nawet jej część - od pewnego pola do końca listy) może być bowiem dowiązana do wielu różnych pól należących do jednej lub kilku struktur.

Część RV jest przeznaczona do przyszłego użytku i na razie nie jest wykorzystywana.

Wewnętrzna reprezentacja wartości atomów jest ściśle związana ze sposobem tworzenia typów wartości w języku ALGOL 1900 i uzależniona od postaci informacji obowiązujących w maszynach cyfrowych ODRA serii 1300 [9].

Wartości całkowite zapisuje się jako liczby binarne w systemie uzupełnieniowym do dwójki i przechowuje w jednej komórce pamięci. Reprezentacja atomów typu INTEGER jest identyczna, jak wartości typu INTEGER używane w programach w ALGOL-u.

To samo dotyczy wartości typu BOOLEAN. Wartości logiczne również zapisuje się w jednym miejscu pamięci. Wartości logicznej FALSE odpowiada 24-bitowy ciąg binarny 0 ... 00, a wartości logicznej TRUE ciąg 0 ... 01.

Liczby rzeczywiste używane przez programy w ALGOL-u zapisuje się zawsze w postaci zmiennoprzecinkowej w dwu kolejnych komórkach pamięci. Bardziej znaczące bity liczby stanowią jej znormalizowaną mantysę (bit b_0 drugiego słowa jest zawsze równy 0), pozostałych 9 bitów składa się na znormalizowaną cechę (cecha rzeczywista powiększona o liczbę 256). Reprezentacja atomu typu REAL różni się od postaci wartości rzeczywistej tym, że zajmuje części A dwóch oddzielnych pól pamięci. Pierwsze pole listy atomu zawiera

24 bity bardziej znaczącej części liczby rzeczywistej, natomiast drugie (końcowe) pole listy zawiera pozostałe bity mantysy oraz cechę zmiennoprzecinkową.

W systemie przetwarzania struktur danych zostały wprowadzone pewne typy atomów nie mające wprost odpowiedników w typach wartości występujących w języku ALGOL 1900.

Jednym z nich jest typ CHARACTER. Atomy tego typu służą do przechowywania wewnętrznej, binarnej reprezentacji znaków pisarskich akceptowanych przez urządzenia zewnętrzne współpracujące w zestawie maszyny cyfrowej. Kody wewnętrzne znaków pisarskich są 6-bitowe, z tym że przedstawienie kodowe każdego znaku może być poprzedzone wewnętrznym przedstawieniem tzw. znaku przesunięcia określającym, do jakiej grupy należy znak pisarski. (W systemie ODRA 1300 znaki pisarskie są podzielone na trzy grupy: α, β, δ).

Dla każdego atomu typu CHARACTER przeznacza się jedno pole pamięci, przy czym reprezentacja znaku pisarskiego poprzedzona kodem przesunięcia zajmuje 12 mniej znaczących bitów części A pola, a pozostałe bity są wyzerowane.

Do przechowywania przedstawień kodowych ciągów znaków pisarskich (łańcuchów) używa się atomów typu STRING. Zewnętrzną postać znaków pisarskich ogranicza się symbolami @, gdy łańcuch wprowadza się za pośrednictwem urządzenia wejściowego lub nawiasami łańcuchowymi ‘ ’, gdy łańcuch występuje jako stała w programie. Wewnętrzna postać łańcucha nie zawiera reprezentacji tych symboli. Kolejne znaki łańcucha zapisuje się w części A pól składających się na listę atomu, z tym że do jednego pola zapisuje się co najwyżej cztery przedstawienia kodowe znaków (niektóre z nich mogą być bowiem poprzedzone znakami przesunięć). Jeżeli w łańcuchu występują kolejno po sobie znaki należące do jednej z grup α lub β, wówczas odpowiedni znak przesunięcia zostanie zapisany tylko jeden raz tak, że będzie poprzedzał pierwszy ze znaków należących do danej grupy. Natomiast każdy znak z grupy δ jest poprzedzony odpowiednim znakiem przesunięcia. Ze względów organizacyjnych liczba znaków wchodzących w skład jednego łańcucha jest ograniczona i wynosi maksymalnie 200 (wliczając w to również znaki przesunięć). Pierwszy znak łańcucha poprzedzony kodem przesunięcia swej grupy zajmuje zawsze najbardziej znaczące bity części A pierwszego pola listy atomu, natomiast ostatni znak łańcucha znajduje się w polu końcowym listy. Wewnętrzną reprezentację łańcucha uzupełnia się zawsze przedstawieniami kodowymi znaków odstępu w takiej liczbie, by wszystkie znaki w komórce A końcowego pola listy atomu były określone.

Ze względu na przewidywane zastosowania systemu przetwarzania struktur danych wprowadzono trzy specjalne typy atomów: ADDRESS, COMMAND i SEQUENCE.

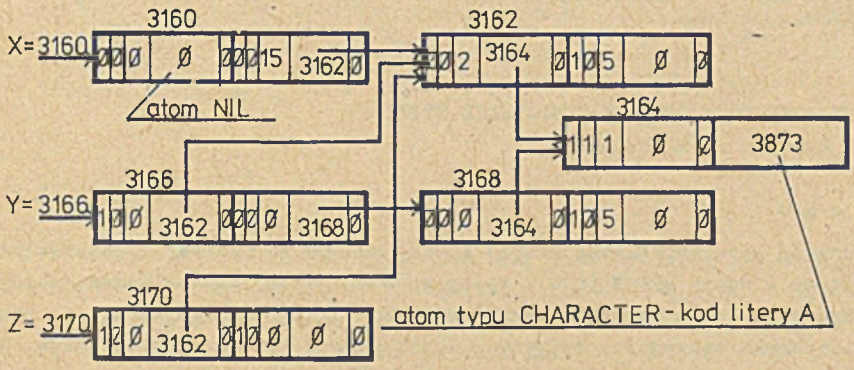
Atom ADDRESS zajmuje jedno pole pamięci. W bitach b_6 ... b_{23} części A pola zapisuje się adres dowolnej komórki pamięci, w bitach b_0 ... b_1 – numer znaku w tej komórce (gdy numer ten jest nieistotny, wówczas wartości bitów b_0 i b_1 powinne być równe 0), a bity b_2 ... b_5 są zerowane.

Również atom COMMAND zajmuje jedno pole pamięci. W części A pola zapisuje się dowolny rozkaz z listy rozkazów maszyny o postaci przyjętej w systemie ODRA 1300.

Wartością atomu typu SEQUENCE jest ciąg rozkazów maszyny cyfrowej. Każdy rozkaz umieszcza się w oddzielnym polu listy atomu. Długość listy atomu typu SEQUENCE może być dowolna.

Opracowany system przetwarzania struktur danych umożliwia jego użytkownikom wprowadzanie dalszych jeszcze, niestandardowych typów atomów. Wewnętrzne reprezentacje takich atomów muszą być wtedy ustalane w programach, które je wykorzystują.

Omówiony sposób budowy pól pamięci i organizacji ich wiązań pozwala na tworzenie w pamięci maszyny cyfrowej dowolnych struktur binarnych. Na rys. 4 w formie przykładu został pokazany graficzny obraz kilku prostych struktur z kompletnym zapisem wartości (w postaci dziesiętnej) we wszystkich częściach pól, które na te struktury się składają.



Rys. 4. Przedstawienie struktur z kompletnym zapisem wartości w ich polach

W niniejszym artykule zostały przedstawione podstawowe elementy wewnętrznej organizacji pamięci maszyny cyfrowej. Służą one potrzebom systemu przetwarzania struktur danych w języku ALGOL 1900. Dzięki nim system może tworzyć w pamięci maszyny cyfrowej dowolne struktury, a także dokonywać ich przekształceń, a użytkownik systemu korzysta przy tym ze środków programowania dostępnych w języku ALGOL. Poprzez wykorzystanie odpowiednich procedur obiekty stanowiące dziedzinę systemu wiąże się ze standardowymi obiektami stosowanymi w języku ALGOL 1900, co stwarza spójność systemu oraz języka, którego potrzebom służy system.

Wysoka elastyczność organizacji systemu zezwala na dalszą jego rozbudowę i rozszerzenie zakresu działania.

LITERATURA

[1] Berztiss A.T.: Data Structures. Theory and Practice. Academic Press. New York 1975.

[2] Bruski J.: Struktury binarne i ich własności. Zeszyty Naukowe Pol.Śl., seria Informatyka z. 1.

[3] Bruski J.: Zbiory charakterystyczne struktur binarnych. Zeszyty Naukowe Pol.Śl., seria Informatyka z. 1.

[4] Bruski J., Grudziński B.: O wykorzystaniu strukturalnej organizacji jednorodnej pamięci maszyny cyfrowej i jej związku z systemem struktur binarnych. Zeszyty Naukowe Pol.Śl. seria Informatyka z.1.

[5] Clark D.W., Green C.C.: An Empirical Study of List Structure in LISP. Communications of the ACM. Vol. 20. nr 2. 1977.

[6] Dahl O.J., Dijkstra E.W., Hoare C.A.R.: Structured Programming. Academic Press. London, New York, 1972.

[7] Turski W.M.: Struktury danych. WNT, Warszawa 1976.

[8] Wegner P.: Języki programowania, struktury informacji i organizacja maszyny cyfrowej. PWN, Warszawa 1979.

[9] Oprogramowanie maszyn cyfrowych ODRA serii 1300. Publikacje WZE Mera Elwro, Wrocław.

КОНЦЕПЦИЯ СИСТЕМЫ СЛУЖАЩЕЙ ОБРАБОТКЕ СТРУКТУР ДАННЫХ В ЯЗЫКЕ АЛГОЛ 1900

Р е з ю м е

В статье рассматривается вопрос почему построено систему обработки структур данных в языке АЛГОЛ 1900 и кроме того принципы выбора языка, которого потребностям служит система. Подробно рассматривается организация памяти использованная системой и структуру полей памяти для этой организации. Приводятся тоже способы хранения элементарных информации принятых как стандартные для системы.

AN IDEA OF THE SYSTEM OF DATA STRUCTURES PROCESSING IN ALGOL 1900

S u m m a r y

The genesis of the system of data structures processing in Algol 1900 and the reason for the choice of programming language were presented in the paper. Organization of memory used by the system and structure of memory's fields were treated in detail. Methods of storage of elementary information accepted as standard for the system were also shown.