mgr inż. ANDRZEJ MERKER
Przemysłowy Instytut
Automatyki i Pomiarów MERA-PIAP
Warszawa

UNIWERSALNY JĘZYK PROGRAMOWANIA PASCAL

1. Wybrane elementy języka Pascal

Język programowania Pascal został opracowany pod koniec lat sześćdziesiątych przez Niklausa Wirtha w Eidgenössische Hochschule (ETH) w Zurychu. Pomimo braku poparcia ze strony potężnych instytucji oraz koncernów produkujących sprzęt i oprogramowanie komputerowe (Fortran i PL/1 zostały wylansowane przez IBM, Cobol powstał pod auspicjami Departamentu Obrony USA), Pascal od kilku lat zdobywa sobie coraz większą popularność na świecie. Opracowano kompilatory tego języka na większość znanych typów maszyn cyfrowych.

Powstanie języków ukierunkowanych problemowo, jakimi są Fortran i Cobol, spowodowało podział dziedziny programowania na dwie podstawowe grupy: obliczenia naukowe oraz przetwarzanie danych. Zastosowanie informatyki do rozwiązywania coraz bardziej skomplikowanych problemów pokazało, że dotychczas używane języki są zbyt mało elastyczne. Stało się konieczne opracowanie języka, który spełniałby obie wyżej wymienione funkcje, a jednocześnie byłby dostatecznie efektywny.

Jednym z pierwszych języków, w których starano się spełnić powyższe zadanie, był PL/1. Poważną wadą tego języka jest jego obszerność, co w praktyce uniemożliwia jego szybkie opanowanie.

Wadę tą udało się usunąć twórcy języka Pascal Niklausowi Wirthowi. Stworzył on język, który nadaje się zarówno do nauki programowania, jak i do rozwiązywania złożonych problemów użytkowych. Pascal może być traktowany jako naturalne rozwinięcie Algolu 60. Przyjęcie struktury blokowej pozwoliło na zachowanie elegancji języka. W Pascalu wyeliminowane zostały zbyt ogólne konstrukcje,na przykład algolowską instrukcję pętli zastąpiono trzema rodzajami instrukcji:

dla (zbliżona do fortranowskiego DO z krokiem równym 1 lub −1),
powtarzaj (z warunkiem logicznym sprawdzanym na końcu pętli),
dopóki (z warunkiem logicznym sprawdzanym na początku pętli).

Utrudnione zostało korzystanie z instrukcji skoku bezwarunkowego. Wszystkie etykiety występujące w programie muszą być zadeklarowane w specjalnej sekcji. Zakazano skoku do wnętrza instrukcji złożonej. Ograniczenie to może spowodować początkowo duże kłopoty (szczególnie programiście przyzwyczajonemu do pisania programów w Fortranie), zmusza jednak programistę do gruntownego przemyślenia struktury programu i zachowania porządku w jego wnętrzu. Struktura danych w stosunku do Algolu 60 uległa znacznemu rozbudowaniu. Wprowadzenie do języka możliwości definiowania przez użytkownika rozbudowanych typów danych umożliwiło wykorzystanie Pascala w dziedzinie przetwarzania danych. Opisane wyżej zmiany zwiększyły znacznie przejrzystość programu i pozwoliły na wykorzystanie języka do celów dokumentacyjnych.

Pełne przedstawienie języka Pascal przekracza ramy tego artykułu. Czytelnik może zapoznać się z opisem języka w pracach [1],[2]. Tutaj chciałbym jedynie przedstawić dostępne w Pascalu struktury danych, które stanowią najciekawszą stronę tego języka i świadczą o jego szerokich możliwościach. Działanie każdego programu polega na wykonywaniu określonych czynności na obiektach zwanych

danymi, które są reprezentowane w programie przez zmienne. W Pascalu każda zmienna występująca w programie musi występować w deklaracji zmiennej, która wiąże jej identyfikator ze zbiorem wartości, jakie ta zmienna może przyjmować (typem zmiennej). Podstawowymi typami danych w Pascalu są typy skalarne. Zalicza się do nich cztery typy standardowe: logiczny (Boolean), całkowity (integer), rzeczywisty (real) i znakowy (char) oraz typy wyliczeniowe. Typ wyliczeniowy jest definiowany przez programistę poprzez podanie explicite zbioru wartości, jakie może przyjmować zmienna, na przykład

```
type element = (kondensator, cewka, rezystor, dioda, tranzystor, inne);
```

Umożliwienie programiście określania własnych skalarnych typów danych znacznie zwiększa przejrzystość programu. We wszystkich innych językach programista jest zmuszony do używania w programie sztucznych wartości liczbowych określających cechy danego obiektu. W Pascalu może określić swój własny typ danych opisujący je mnemonicznie.

Wykorzystując typy skalarne można budować typy okrojone (podtypy), poprzez podanie najmniejszej i największej wartości w podtypie, na przykład

```
litera = 'A' . . 'Z' ;     (* typem podstawowym jest typ char * )
ebierny = kondensator . . rezystor ;   (* typem podstawowym jest element zdefiniowany
                                          w poprzednim przykładzie *)
y = 2ØØ .. 4ØØ ;
```

Wprowadzenie typów okrojonych pozwala na zwiększenie kontroli nad działaniem programu, szczególnie na etapie jego uruchamiania, gdyż uniemożliwia przyjęcie przez zmienną wartości nie przewidzianej przez programistę.

W niemal wszystkich językach programowania można grupować dane w większe obiekty strukturalne. W Pascalu występują cztery podstawowe rodzaje struktur: tablicowa, rekordowa, plikowa i zbiorowa. Tablica składa się ze ściśle określonej liczby elementów tego samego typu. Liczba wymiarów tablicy nie jest ograniczona. Typ indeksu jest określany w definicji i może być dowolnym ograniczonym typem skalarnym. Składniki tworzące tablicę mogą być dowolnego typu, na przykład

```
A = array ['a' . . 'z'] of integer ;
B = array [ Ø . . 1Ø] of array [1 . . 2] of element ;
```

Zaletą tablic w Pascalu jest ciekawa adresacja ich elementów. W stosunku do Algolu 60 wadę stanowi statyczne określanie rozmiarów tablic, co powoduje konieczność powtórnej kompilacji programu, gdy należy zmienić któryś z nich.

Kolejną strukturą, typową dla języków ukierunkowanych na przetwarzanie danych, jest rekord. Składa się on z części zwanych polami, które mogą być różnego typu. W programie wolno wykonywać operacje na całych rekordach, jak i na ich poszczególnych polach.

Przykład:

```
data = record
   dzień:     1 . . 31 ;
   miesiąc:   1 . . 12 ;
   rok:       ØØ . . 99
end ;
personal = record
   imię, nazwisko: array [1 . . 15] of char ;
   urodzony:       data
end
```

Zdarza się często, że chcemy opisać obiekty podobne, różniące się jedynie częściowo. W porównaniu z Cobolem i PL/1, Pascal oferuje bardzo przejrzysty zapis takiej sytuacji. Możemy się posłużyć rekordem posiadającym pole zmienne (warianty). Rodzaj wariantu jest wtedy określony przez zawartość pola znacznikowego (wyróżnika) znajdującego się we wspólnej dla wszystkich wariantów części, np:

```
pracownik   record
  danep       : personal ;
  uposażenie  : Ø .. 25ØØØ ;
  . . . .
  płeć:  (kobieta,mężczyzna); ( * koniec części wspólnej *)
  case oddelegowany: Boolean of  (* pole wyróżnika *)
    false:  (dział: (FA,FL . . . . . OAK, OAM . . .)) ;
    true:   (nazwazakładu, adreszakładu: array [1 . . 2Ø] of char)
end ;
```

Odpowiedni rozmiar pamięci dla rekordu jest przydzielany automatycznie. Rola programisty sprowadza się jedynie do ustawienia lub odczytania wartości pola znacznikowego. Jeśli w programie dla podanego wyżej przykładu wartość pola wyróżnika przyjmie wartość true to w rekordzie będą znajdować się pola *nazwazakładu* i *adreszakładu*, zaś w przeciwnym wypadku w rekordzie znajdzie się pole *dział*.

Struktura plikowa jest ciągiem elementów tego samego typu o dostępie sekwencyjnym (potocznie jest nazywana zbiorem danych).

Ostatnim rodzajem struktury nie występującym w innych językach jest zbiór matematyczny. Struktura zbiorowa definiuje zbiór wartości, będący podzbiorem wszystkich wartości typu podstawowego. Na przykład

```
  type zbiorl = set of 1 . . 3 ;
```

zmienna typu zbiorl może przyjmować wartości

puste, (1), (1,2), (1,2,3)

Na zmiennych typu zbiorowego można wykonywać teriomnogościowe operacje sumy, różnicy, przecięcia zbiorów. Określone są również relacje równości i zawierania się zbiorów oraz relacja przynależności elementu do zbioru.

Zmienne zadeklarowane explicite w deklaracjach są zmiennymi statycznymi. Podczas analizy deklaracji, zmiennej takiej zostaje przydzielona pamięć, do której odwołujemy się w programie poprzez identyfikator zmiennej. W Pascalu można zażądać innej metody przydziału pamięci, czyli allokacji dynamicznej. Pamięć jest wtedy przydzialana dynamicznie na żądanie programisty. Zmienna nie ma wtedy związanego z sobą identyfikatora, a jej adres jest dostarczany za pomocą wskaźnika.

Zmienne wskaźnikowe mogą występować jako elementy składnikowe zmiennych strukturalnych. Odpowiednie zdefiniowanie typów wskaźnikowych pozwala na łatwe budowanie kolejek, list i łańcuchów zdarzeń. Są one szczególnie przydatne do celów symulacyjnych.

Jak widać z powyższego opisu, struktury danych występujące w języku Pascal pozwalają na bardzo szeroki zakres jego zastosowań. Nadaje się on zarówno do przetwarzania danych ekonomicznych, tworzenia kompilatorów i preprocesorów dla innych języków programowania a także budowania złożonych modeli opisujących działanie systemów operacyjnych lub procesów przemysłowych.

2. Uruchamianie programów pod systemem OS/JS MFT

Realizacja programu napisanego w Pascalu składa się z kilku faz. W pierwszej z nich do pamięci operacyjnej jest ściągany kompilator, który tłumaczy program i tworzy moduły gotowe do wykorzystania przez systemowy program łączący oraz umieszcza je w pamięci zewnętrznej. Następnie program łączący (LINKAGE EDITOR) dokonuje konsolidacji modułów wyprodukowanych przez kompilator ze standardowymi procedurami bibliotecznymi i innymi modułami dostarczonymi przez użytkownika. W wyniku działania programu łączącego powstaje moduł wykonywalny, który zostaje ściągnięty do pamięci i rozpoczyna się jego wykonanie.

Opisane wyżej czynności stanowią pracę (JOB) dla systemu operacyjnego. Kolejne czynności, jakie powinien wykonać system, użytkownik musi opisać za pomocą kart sterujących, czyli zdań języka opisu

prac JCL. Aby uprościć czynności wykonywane przez programistę, często powtarzające się opisy prac są przechowywane w bibliotekach procedur skatalogowanych. Użytkownik może wywołać taką procedurę, uzupełniając lub modyfikując podane w niej informacje.
Procedury skatalogowane, opisujące przebieg programów napisanych w Pascalu są zbliżone do procedur dla innych języków programowania. Dlatego ich dokładny opis zostanie pominięty. Korzystając z nich należy pamiętać o następujących regułach:

- Dla kompilacji (krok PASC) zdanie DD INPUT określa plik, w którym znajduje się program źródłowy.
- Dla konsolidacji (krok LKED) zdanie DD SYSIN określa bibliotekę procedur dołączonych przez użytkownika.
- Dla wykonania (krok GO) pliki występujące w programie z wyjątkiem pliku OUTPUT muszą być opisane w zdaniach DD. Nazwy plików muszą odpowiadać nazwom zdań języka JCL opisujących je.

Poniżej został przedstawiony ciąg zdań języka JCL niezbędny do translacji, łączenia i wykonania programu w Pascalu, którego postać źródłowa znajduje się na kartach. Program ten nie wywołuje żadnych procedur zewnętrznych.

```
//PAS1   JOB    MSGLEVEL=1
//      EXEC  PASCLG
//PASC.INPUT  DD *
     ciąg kart z programem
/*
//GO.INPUT  DD
     plik input
/*
. . . .  opisy innych plików występujących w programie
//
```

3. Uwagi o kompilatorze

Kompilator języka Pascal włączono do oprogramowania dostępnego w Przemysłowym Instytucie Automatyki i Pomiarów w styczniu 80. Kompilator ten, opracowany w Instytucie Podstaw Informatyki PAN [3], został przekazany naszemu Instytutowi bezpłatnie. Akceptuje on pełny język Pascal i generuje kod wynikowy w postaci modułów przyjmowanych przez systemowy program łączący. Zachowanie konwencji dotyczących przekazywania parametrów umożliwia dołączanie do programu w Pascalu zarówno oddzielnie skompilowanych procedur pascalowych jak i podprogramów napisanych w innych językach (na przykład modułów z fortranowskiej biblioteki matematycznej).
Kompilator został wyposażony w narzędzia ułatwiające programiście uruchamianie i testowanie programów. Parametry kompilacji umożliwiają generowanie kodu wynikowego wykrywającego złą wartość wskaźnika, wartość poza dozwolonym zakresem, przekroczenie indeksu. Stosując tzw. profil dynamiczny można sprawdzić, czy program przeszedł przez wszystkie występujące w nim ścieżki. Błędne zakończenie programu powołuje wypisanie informacji śledzących, które podają wartość zmiennych i zagnieżdżenie wywoływanych procedur w momencie wystąpienia błędu.

Literatura

[1] Jensen K., Wirth N.: Pascal — User Manual and Report. Springer, New York 1974.

[2] Iglewski M., Madey J., Matwin S.: Pascal. Język wzorcowy — Pascal 6000. WNT, Warszawa 1979.

[3] Iglewski M.: System Pascal 360. Informatyka, nr 2/1979.