# X. OCENA STANU DOTYCHCZASOWEGO I TENDENCJE ROZWOJOWE PROGRAMOWANIA

1. Wydaje się, że lata obecne zamykają pierwszy etap rozwoju programowania. Cechą widoczną tego etapu jest nadprodukcja języków programowania przy równoczesnym braku teoretycznych podstaw budowy języków (ujmujących zarówno składnię, jak i semantykę oraz pragmatykę). Dotychczas stosunkowo wiele zbadano jedynie w zakresie składni. Efektem niedorozwoju teoretycznego jest niski stopień zamienności języków, czy też nawet ich poszczególnych wersji, oraz trudności związane z symulacją i emulacją.

Dorobkiem dotychczasowych prac są przede wszystkim języki powszechnego użycia: FORTRAN, ALGOL i COBOL. Przyszłość przyniesie nam zapewne poważne prace teoretyczne, które umożliwią skonstruowanie efektywnych języków, prawdopodobnie wyspecjalizowanych (na problem,nie zaś na maszynę) lecz opartych na wspólnych podstawach teoretycznych. Dotychczas obserwowaliśmy przeważnie przypadkowe (z wyjątkiem ALGOLu i COBOLu) wysiłki poszczególnych producentów czy użytkowników ukierunkowane na tworzenie nowych języków. Większość tych języków miała krótki żywot, a często jedną (czy też jedyną) ich zasługą było wprowadzenie nowej terminologii dla już uprzednio zdefiniowanych spraw. W sytuacji, kiedy mamy kilka tysięcy języków, a prawie każdy z nich operuje innymi pojęciami, bardzo jest trudno znaleźć fachowca, który potrafiłby zapamiętać wszystkie różnice i spróbował wprowadzić wspólny aparat pojęciowy, umożliwiając tym samym szeroką analizę porównawczą.
Przykłady nadmiaru pojęć mogą być następujące:

- do określenia segmentu programu: segment, program-unit, module, section-nr.
- do określenia deklaracji: declarative, klauzula, dyrektywa.

2. Wydaje się, że w perspektywie można mówić o dwóch tendencjach rozwojowych programowania. Po pierwsze, będą opracowane metajęzyki, czyli języki teoretyczne, a celem ich będzie stworzenie kryteriów oceny języków oraz reguł budowy syntaktyki i semantyki. Po drugie, na bazie języków teoretycznych powstaną języki wąskospecjalizowane, dostatecznie efektywne a równocześnie łatwe w użyciu. Języki te powinny być zbliżone do języków naturalnych (unikając równocześnie werbalności), by umożliwić łatwy kontakt z maszyną, oraz być na tyle "fachowo", by zapewnić łatwe wyrażenie specyfiki problemu. Dostateczna efektywność programu powinna być zapewniona przez środki software'owe budowane modularnie w celu możności stosowania dowolnej kompozycji modułów i dalszej rozbudowy.

Dotychczasowe koncepcje języków (X - 2o) oparte były o tzw. budowę pozycyjną danych, wskazywaną przez specjalne deklaracje i "poziomowanie" hierarchii danych. Przewiduje się, że w przyszłości budowa pozycyjna zostanie zarzucona na rzecz oceny semantycznej.

Za próbę stworzenia metajęzyka można zapewne uważać uniwersalny generator HELP (Highly Extendible Languages Processor) ogłoszony przez organizację Advanced Computer Techniques i nazwany "general-purpose natural language generator (X - 19). HELP posiada następujące możliwości:

a/ translacja każdego języka naturalnego na inny,

b/ translacja skrótowych wersji FORTRANu, COBOLu (np.COAX) i innych języków na formy żądane do kompilacji,

Teoretycznie programista może stosować więc dowolny (nawet własny) język programowania.

3. Niekiedy spotkać można w publikacjach (np. B - 11) wzmianki o tzw. palindromicznym (odwrotnym) programowaniu.
Polega ono na tym, że program wykonywany jest nie od pierwszej lecz od ostatniej instrukcji, dając ten sam rezultat (podobnie jak czytanie słów "radar", "rotor").
Oto inne przykłady pracy palindromicznej:
a/ translacja programu wewnętrznego na program w języku zewnętrznym (symbolicznym, wyższego rzędu),
b/ procedury wykrywania błędów idąc "od tyłu" (tj. od wyniku),
c/ pakiety programowe "FLOWCHART" przeznaczone do tworzenia schematów blokowych napisanych programów.
4. Jedną z cech charakterystycznych maszyn IV generacji ma być dobre oprogramowanie komunikacyjne. Należy przypuszczać, że w związku z tym znacznie wzrośnie rola języków konwersacyjnych oraz że klasyczne języki programowania zostaną odpowiednio zmodyfikowane (wzbogacone). Jest to warunek użytkowania wspólnych baz danych w systemach bieżącego informowania kierownictwa.

Tak np. ostatnio trwają prace nad włączeniem do COBOLu właściwości CCF (Cobol Communication Facility), która ma umożliwić stosowanie COBOLu w systemach pytanie-odpowiedź.
Między innymi będzie to możliwe dzięki wprowadzeniu do DATA DIVISION specjalnej sekcji COMMUNICATION SECTION, instrukcji RECEIVE, SEND, itp.

W systemach komunikacyjnych dużą rolę powinny odgrywać

tzw. przyspieszone kompilatory (incremental compilers) (R - 8), które tłumaczą każde zdanie programu zewnętrznego natychmiast po wprowadzeniu z terminalu on-line, dając od razu diagnostykę błędów. W trakcie wprowadzania programu można eliminować poprzednie zdania, wprowadzać dodatkowe zdania itp. Do tego typu translatorów zalicza się: QUIKTRAN, RUSH (wersja PL/I), FIV (wersja FORTRANu IV), CAL oraz w dużej mierze BASIC.

5. Pewną nadzieję (już od pewnego czasu zresztą) na znaczne uproszczenie programowania stwarzają języki tablicowe (tabular languages), przeznaczone do tworzenia programów na podstawie odpowiednio opisanych tablic decyzyjnych. Skonstruowano już parę takich języków (TABSOL, LOGTAB,FORTAB, DETAB 165, GECOM), żaden jednak nie uzyskał szerszego zastosowania (podobnie, jak i same tablice decyzyjne).

Wygodne jest stosowanie tzw. przedtranslatorów, pozwalających na budowę programów mieszanych, złożonych z tablic decyzyjnych i z rozkazów COBOLu (lub FORTRANu).

Tablice decyzyjne zostały po raz pierwszy użyte prawdopodobnie w latach 50-tych w firmie GE (X - 21). W latach 60-tych rozwinięto szersze prace nad tablicami decyzyjnymi (np. Decision Tables Symposium w Nowym Jorku w 1962 roku).

Tablice decyzyjne są szczególnie pomocne do przedstawiania złożonych logicznie sytuacji, które jest trudno wyrazić graficznie na schematach blokowych. Inną zaletą jest to, że umożliwiają skontrolowanie, czy uwzględniono wszystkie możliwe kombinacje warunków i wynikające z nich czynności.

6. Docelową perspektywą programowania są systemy samoprogramujące. Podobno ma być to cecha charakterystyczna maszyn V generacji.

Wydaje się nam, że do wyeliminowania programistów nie dojdzie nigdy.

- - - ... - - -