II. PIERWSZE KONCEPCJE JĘZYKÓW WYŻSZEGO RZĘDU

Wraz z rozwojem urządzeń pamięciowych i pilną potrzebą ułatwienia programowania pojawiły się koncepcje ujęcia czynności sterujących (operacyjnych) oraz obliczeniowych i logicznych, nie tylko w postaci układów konstrukcyjnych (tj.mikroprogramów) i kodów maszynowych, lecz również za pomocą instrukcji specjalnych wymagających uprzedniego tłumaczenia.

W zależności od sposobu dokonywania tłumaczenia takich instrukcji rozróżniamy systemy interpretacyjne i kompilacyjne. Systemy interpretacyjne stanowią rozwinięcie idei podprogramów. Po rozpoznaniu każdej instrukcji wywoływany jest odpowiedni podprogram i następuje jego wykonanie, po czym pobierana jest następna instrukcja.

W systemach kompilacyjnych cały program źródłowy (source program) jest najpierw tłumaczony na język wewnętrzny (object program), a dopiero później wykonywane są obliczenia w zakresie całego programu.

Pierwsze interpretacyjne programy zastosowano w celu wykonywania operacji w zmiennym przecinku na maszynach pracujących w stałym przecinku oraz symulacji pracy maszyny wieloadresowej na maszynie jednoadresowej. Jednym z pierwszych języków interpretacyjnych był interpretacyjny system kodowania wyrażeń algebraicznych opracowany w latach 1952-1953 przez J.H.Laninga i W.Zieglera w M.I.T. dla maszyny Whirlwind.

Podobne prace dla systemów kompilacyjnych prowadził w Szwajcarii Heinz Rutishauser zatrudniony w Federalnym Instytucie Technologicznym w Zurichu. Już w 1952 roku przedstawił

on metody umożliwiające wprowadzenie do maszyny wyrażeń algebraicznych w zwyczajnej formie. Ponadto Rutishauser zaproponował bardzo prostą i zwięzłą instrukcję do organizacji pętli "for k = /i/ 10".

Ograniczoną próbą zastosowania matematycznej notacji na wejściu był system SHORT CODE zaproponowany w 1949 roku przez J.Mauchley'ego /współtwórcy ENIACa/ a zastosowany w 1952 roku dla maszyn UNIVAC i BINAC.

W 1953 roku dla maszyny IBM 701 opracowano Speed Coding System, w opracowaniu którego uczestniczył J.Backus.

W 1956 roku pojawiają się pierwsze publikacje na temat MANCHESTER UNIVERSITY AUTOCODE. Firma GEC Computers na bazie tego języka opracowała w 1962 roku "autokod" (autocode) dla maszyny FERRANTI MERCURY.

Termin kompilator został wprowadzony przez Dr Grace HOPPER, zatrudnioną w firmie Remington Rand jako szef programowania. Zaproponowała ona użycie języka angielskiego do pisania instrukcji.

Już w 1952 roku opisała działanie kompilatora a następnie kierowała pracami, w wyniku których powstały zapewne pierwsze w świecie kompilatory A0 i Ai. w 1955 roku powstaje wersja A2, a następnie AT3 (nazwana ARITH-MATIC). Pierwszy opis tego języka znany był w 1957 roku. W tym samym roku pojawił się opis języka FLOW-MATIC, ukierunkowanego na zastosowanie ekonomiczno-administracyjne.

FLOW-MATIC, zwany początkowo jako język B-0, charakteryzował się szerokim użyciem języka angielskiego zarówno do pisania instrukcji jak i do opisu danych. Język ten stanowił

później jedną z podstaw języka COBOL (1959). Na poparcie tego twierdzenia przytoczymy parę instrukcji języka FLOW-MATIC, prawie żywcem wziętych do COBOLu :

ADD A B C, DIVIDE nazwa-danych -i BY nazwa-danych-2 GIVING nazwa-danych-3, MOVE n-d-i TO n-d-2, READ ITEM n-d IF END OF DATA GO TO OPERATION.....

Ważnym problemem programowania jest algorytmizacja zadania, tj. jednoznaczne określenie reguł działania procesu obliczeniowego, uwzględniające wszystkie możliwe kombinacje i etapy. Przy okazji warto zaznaczyć, że słowo "algorytm" pochodzi od nazwiska żyjącego w IX wieku uzbeckiego matematyka AL-HOREZMI (T - 3).

Na lata 1952-1953 przypadają początki opracowania w ZSRR (w Matematycznym Instytucie AN ZSRR) pod kierownictwem A.A. Liapunowa - tzw.operatorowego opisu algorytmów. U podstaw metody leży założenie, że algorytm składa się z szeregu powtarzalnych arytmetycznych i logicznych etapów. Każdy z nich jest realizowany za pomocą tzw. operatora, reprezentującego grupę elementarnych działań, wykonanie których może przebiegać automatycznie w powiązaniu z tzw. programującym programem. Program zewnętrzny (pisany przez programistę) zawiera takie informacje jak:

a/ typ operatora (A- arytmetyczny, P - logiczny, F/i/ - przeadresowanie, E /m;n/ - skok do operatorów z numerami m,m-1, ...n) itp. nr porządkowy operatora, kierunki skoków,

b/ wzory matematyczne dla operatorów arytmetycznych,

c/ parametry przeadresowania,

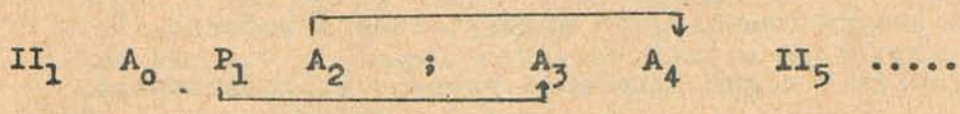
d/ podprogramy,

itp.

Są to informacje o operatorach standardowych. Operatory niestan-

dardowe muszą być całkowicie zaprogramowane w języku wewnętrznym. Główne czynności sprowadzają się do wyboru metody obliczeń, sporządzenia logicznego schematu programu (przez wypisanie ciągu operatorów), określeniu wzorów i parametrów oraz zaprogramowania niestandardowych fragmentów programów.

Oto przykład logicznego schematu w metodzie operatorowej:



Programujący program opracowano m.in. dla maszyny Strieła-4.

Opracowanie języka programowania wymaga zdefiniowania szeregu takich elementów, jak: alfabet (wykaz dozwolonych znaków), wyrażenie (instrukcja) i zdanie (zespół instrukcji zakończony znakiem końca, np. kropka), deklaracje (np. dotyczące opisu danych), dyrektywy operacyjne (związane z organizacją translacji), reguły tworzenia procedur (bloków, paragrafów), reguły wzywania podprogramów i umieszczania wstawek w innych językach itp.

Potrzebę ograniczonego słownika znaków sygnalizował już Turing w koncepcji swojej abstrakcyjnej maszyny. Ze znaków tych proponował tworzenie dowolnych kombinacji.

Problemy konstrukcji, interpretacji i stosowalności języka ujmowane są jako składnia, semantyka i pragmatyka.

Składnia określa, jakie sekwencje znaków są dopuszczalne (np. w postaci instrukcji), przykładowo A + B może stanowić prawidłową konstrukcję w języku "XX", podczas gdy +AB jest nielegalne. Semantyka obejmuje reguły określania znaczenia dopuszczalnych kombinacji, zaś pragmatyka - reguły stosowalności zarówno seman-

tyki jak i składni.

Wszystkie trzy powyższe działy powinny być opisane w sposób jednoznaczny, a jest to możliwe dzięki zastosowaniu metod sformalizowanego zapisu. Tak się złożyło, że dotychczasowe badania dotyczyły głównie składni, zaś semantyka i pragmatyka czekają dopiero na opracowanie, a właściwie stworzenie.

Badania nad składnią zapoczątkował w 1956 roku Chomsky. Zdefiniował on podstawowe elementy pragmatyki formalnej:

a/ podzbiór terminalny, tj. słownik zwrotów języka (małe litery),

b/ słownik nazw strukturalnych, tj. zmiennych (duże litery),

c/ produkcje tj. prawa podstawiania.

Odnośnie tego ostatniego elementu warto dodać, że pojęcie podstawienia, zdefiniowane przez Posta i Thuego, stanowi kluczowy kanon w metodach zapisu składni.

Duże zasługi w rozwoju notacji posiada BACKUS. W notacji (1959) swojej zmienił on znak podstawienia Chomskyego z $\rightarrow$ na : : =, zaś wprowadzenie nawiasów $\langle , \rangle$ pozwoliło na zamianę dużych liter przez opisowe słowa lub frazy.

Notacje Backusa wykorzystano w ALGOL REPORT /1960/ zredagowanym przez Naura. Stąd też pochodzi skrótowa nazwa notacji BNF (Backus Normal Form lub Backus Naur Form). Rozwinięciem zapisu BNF jest metoda van Wijngaardena, przewidująca nieskończoną liczbę produkcji. Niekiedy spotkać można oznaczenie PNF (Paniani Backus

Form). Paniani był nauczycielem perskim, żyjącym ok. 600 lat p.n.e., który opracował system notacji zbliżony do formy BNF (B - 4).

K.E.Iverson zaproponował (1964) notację w pewnym sensie pośrednią pomiędzy "rozwlekłym" zapisem Backusa i "anonimowym" zapisem Chomsky'ego. Notacja Iversona umożliwia precyzyjny opis złożonych procesów sekwencyjnych. Oto jej przykłady:

a/ operacje arytmetyczne: + - x ÷

b/ operacje logiczne: "I" W ← U∧V

"LUB" W ← U∨V

C/ inne: "przesuń na lewo X o K miejsc/ Z ← K↑X

"przesuń na prawo X o K miejsc/ Z ← K↓X

W ośrodku IBM w Wiedniu opracowano (publ. 1968) tzw. wiedeńską metodę opisu języków programowania. Użyto jej m.in. do opisu języków PL/I i ALGOL 60. W metodzie tej zastosowano aparat formalny tzw. obiektów abstrakcyjnych. Do opisu struktury syntaktycznej oraz procesów używa się grafów (w postaci drzew). Każdy wierzchołek grafu procesu związany jest z instrukcją. Wykonanie instrukcji powoduje usunięcie związanego z nią wierzchołka drzewa sterowania.

Również o graficzną metodę przedstawiania programu opiera się metoda Floyda. Stosuje się w niej schematy blokowe, składające się z bloków operacyjnych i warunkowych, połączonych strzałkami. Dla każdej strzałki w sposób formalny (stosując specjalrą notację) podaje się warunki dotyczące zmiennych. Program sprawdzany jest przez przypisanie zmiennym aktualnych wartcści i porównanie ich z warunkami. Metoda ta stanowi więc pewną analogię do znanej metody sprawdzania programów przez tzw. suche przebiegi. Opis metody Floyda podany jest w publikacji A.Mazurkiewicza /M - 2/.

Pierwszy system oznaczeń graficznych (schematów blokowych) został zaproponowany przez Burksa, Goldstine'a i von Neumanna, przy czym był on wygodny dla maszyn jednoadresowych, podobnych do tych, które zostały zaprojektowane w Institute of Advanced Study. Mimo, iż schematy te były ukierunkowane na konkretne maszyny, zapoczątkowały one metodę uniwersalną, pozwalając na dokładne wyrażenie logiki programów niezależnie od rodzaju maszyny. Jest to więc pewne przybliżenie do języka uniwersalnego.

III. PRZYCZYNY PRAC NAD AUTOMATYZACJĄ PROGRAMOWANIA ORAZ TRUDNOŚCI WDRAŻANIA JĘZYKÓW PROGRAMOWANIA

Do dziś jeszcze spotykamy przeciwników posługiwania się autokodami. Przy okazji wyjaśniamy, że przez "autokod" rozumiemy tutaj język AUTOmatycznego KODowania, tj. taki język, który na podstawie jednej zewnętrznej instrukcji generuje sekwencję rozkazów w języku wewnętrznym. Tak więc definicja ta nie obejmuje języków symbolicznych o współczynniku kodowania "jeden do jeden", jako że stanowią one po prostu zmodyfikowaną wersję języka wewnętrznego (przykładem takiego języka jest w zasadzie SSK - Sistiema Simwoliczeskowo Kodirowanija - dla emc Mińsk 32). Wyraz "autokod" pochodzi prawdopodobnie od nazwy jednego z pierwszych języków MANCHESTER UNIVERSITY AUTOCODE, opracowanego w II połowie lat 50-tych w Wielkiej Brytanii. Często, naszym zdaniem, niesłusznie pojęcie autokodu utożsamiane jest wyłącznie z językami symbolicznymi, co zostało być może spowodowane nazwą IBM-owskiego języka symbolicznego AUTOCODER.

Podobnie, jak przy każdym innym rodzaju postępu, wprowadzanie