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

autokodów natrafiało na duże trudności, noszące przede wszystkim charakter psychologiczny.
Obecnie pojęcie "Autokod" traktowane jest w literaturze jako archaizm, użyty do nazwania pierwszych eksperymentalnych języków z FORTRANEM I włącznie, w których - dzięki zastosowaniu notacji zbliżonej do matematycznej wyeliminowano współczynnik kodowania 1:1 oraz pozostawiono translatorom sprawę alokacji pamięci.

Już w 1953 roku, a więc wtedy, kiedy przeprowadzono dopiero pierwsze eksperymenty nad automatyzacją programowania, na jednym z pierwszych posiedzeń ACM (Association for Computing Machinery) omawiano sprawę tzw. "prymitywistów" (zwolenników języka wewnętrznego) oraz "żołnierzy postępu" (space cadets).
Ci ostatni posądzani byli o propagowanie luksusu programowania wg "śmiesznych" schematów ograniczających myślenie.

Zarówno jedni jak i drudzy dysponowali pewnymi argumentami. Również i dzisiaj sprawa poziomu programowania nie jest zupełnie (przynajmniej dla niektórych) wolna od nieporozumień. Wydaje się jednak, że sama praktyka (popularność zewnętrznych języków programowania) spór ten rozstrzygnęła.

Najpoważniejszym argumentem przeciwko autokodom (a w szczególności przeciwko językom wyższego rzędu typu ALGOL, COBOL) jest mała efektywność programu wewnętrznego, jaki generują, oraz większe zapotrzebowanie na pamięć operacyjną. Argumenty te, aczkolwiek całkiem słuszne (jeśli poniekąd rozpatrywać je abstrakcyjnie), nie przekonywują, jeśli zważymy, że w ostatnim okresie nastąpił ogromny wzrost szybkości działania maszyny i powiększenie pojemności pamięci operacyjnej, wspomaganej przez pamięci

pomocnicze o dostępie wyrywkowym. Opieranie się na porównaniach pracy małych maszyn (dla których autokody są rzeczywiście mało efektywne) jest nie do przyjęcia, zważywszy, że produkcja ich należy do przeszłości (mamy na myśli maszyny o szybkości kilkuset operacji czy kilku tysięcy operacji na sekundę oraz wyposażenie w 4-8K pamięci operacyjnej). Straty czasu tak znaczne przy kompilatorach lat 60-tych (X - i) zostały znacznie (niekiedy kilkakrotnie) zmniejszone.

Niemniej jednak w pewnych sytuacjach wybór elementarnego języka programowania można uznać za słuszny. W szczególności dotyczy to programów standardowych czy też translatorów (niekoniecznie) oraz niektórych programów obliczeniowych o dużej częstotliwości przetwarzania (np.codziennie) i operujących na dużej ilości danych.

Porównując efektywność języków należy wziąć pod uwagę kwalifikacje programistów i jakość translatorów, jako elementy zmienne przy każdym porównaniu. Spotyka się opinie, że program przetłumaczony z języka wyższego rzędu (przez dobry kompilator) dorównuje klasą programowi w języku elementarnym napisanym przez programistę średniej klasy.

Szczególny opór, np. w stosunku do COBOLu, przejawiają programiści, którzy uprzednio długo programowali w języku symbolicznym lub wewnętrznym. Przestawienie się na inne sposoby programowania stanowi dla nich trudną barierę.

W poniższej tabeli przedstawimy zależność kosztu i efektywności od poziomu języka (wg "Management Standards for Data Processing" D.H.Brandom, 1963):

		I poziom język symb. 1:1	II poziom język symbolicz. + makro-instr.	III poziom typ COBOL ALGOL	IV poziom generatory
Dobry programista	efektywność programu w jęz.wewn.	95%	90%	60-85%	45-65%
	koszt (indeks)	100	90	40-60	20-40
Zły programista	efektywność	70%	75%	50-70%	40-60%
	koszt	110	100	50-70	30-50

Wg tego samego źródła, struktura zastosowań poszczególnych poziomów języków przedstawia się następująco:

Poziom języka	1962	przewid. 1972
1. Symboliczny 1 : 1	65%	25%
2. Symboliczny z makroinstr.	20%	15%
3. Kompilatory	5%	40%
4. Generatory	10%	20%

Inne późniejsze (1969) badania amerykańskie (B - 5) wykazały, że w zastosowaniach administracyjno-ekonomicznych (poza podprogramami standardowymi) język wyższego rzędu np. COBOL (lub inny do przetwarzania danych) dorównuje językom symbolicznym.

Przy porównywaniu języków programowania należy stosować nie jedno kryterium, lecz brać pod uwagę cały zespół czynników:

1. Czas przejścia od "problemu" do programu, czyli uczenie się języka oraz zdefiniowanie algorytmu odpowiednio do wymogów języka. Jest to czynnik, na który bardzo rzadko zwraca się uwagę.
2. Czas pisania programów.
3. Czas uruchomienia programów.
4. Efektywność translatora:
   - czas translacji,
   - pojemność pamięci operacyjnej niezbędna dla translacji,
   - efektywność programu w języku wewnętrznym (po translacji) którą można określić jako ilość rozkazów wewnętrznych, pojemność pamięci koniecznej do przechowywania programu i danych oraz czas biegu programu (czyli czas przetwarzania).

<u>Zalety języków wyższego rzędu</u> ująć można następująco:

1/ pracochłonność programowania jest co najmniej kilkakrotnie niższa,

2/ języki te z reguły są łatwiejsze do opanowania (m.in. ze względu na użycie słów naturalnego języka),

3/ łatwiej i krócej uruchamia się programy (krótszy program zewnętrzny, łatwiejsza diagnostyka błędów, mniej pomyłek programisty, łatwiejsze korekty itp.),

4/ program stanowi sam w sobie nieźłą dokumentację (odnosi się to szczególnie do COBOLu), co nie jest bez znaczenia dla organizacji podziału pracy, fluktuacji kadr, wprowadzania zmian do programu oraz wymiany doświadczeń,

5/ względna niezależność programu (napisanego w języku powszechnego użycia: COBOL, FORTRAN, ALGOL) od typu maszyny, co umożliwia użytkownikowi zachowanie ciągłości przetwarzania przy zmianie typu maszyny (po dokonaniu pewnych przeróbek nie zmieniających jednak podstawowych rozwiązań programowych) zaś programiście - zachowanie poprzednich kwalifikacji,

6/ logika złożonego programu może zostać przedstawiona stosunkowo prosto (syntetycznie), co ułatwia tzw. suche (przy biurku) sprawdzanie biegu programu,

7/ programista może bardziej koncenrtować się na treści problemu, dla którego należy opracować program, niż na technice programowania i właściwościach konstrukcyjnych maszyny; dlatego też możliwe jest posiadanie dobrych analityków, będących równocześnie dobrymi programistami w COBOLu,

8/ języki wyższego rzędu są niezastąpione w systemach konwersacyjnych człowiek-maszyna typu pytanie-odpowiedź, gdzie chodzi o łatwy i szybki dostęp do maszyny dla nieprogramistów.

Niektóre z wyżej podanych zalet stają się bardzo ważne w sytuacji ostrego deficytu kadr specjalistów oraz postępu technicznego (zmiany maszyn,a więc i języków maszynowo zorientowanych). Ponadto języki wyższego rzędu są szczególnie przydatne w przypadku konieczności szybkiego oprogramowania systemu i przy tworzeniu tzw. pierwszych wersji programów (mających na celu sprawdzenie ich logiki) wyprzedzających prace długofalowe nad optymalnymi pakietowymi rozwiązaniami.

Jeśli mimo tych zalet, występowały duże trudności wdrażania języków wyższego rzędu, to wynikało to nie tyle z oporu użytkowników ile niechęci producentów. Można twierdzić, że właściwie żaden język nie uzyskiwał "prawa obywatelstwa" bez walki.

Oto przykłady:

Firmowy język IBM - FORTRAN opracowany w latach 1954 - 1955 (niewątpliwie jeden z najstarszych języków programowania wśród obecnie stosowanych) aż do 1961 roku był ignorowany przez innych producentów (1961 - pierwsza wersja FORTRANu I dla UNIVAC Solid State 80), zaś późniejsza akceptacja języka powodowana była głównie chęcią przejęcia klientów od IBM.

Z kolei firma IBM usiłowała zbojkotować język COBOL; mimo własnego udziału w pracach nad jego opracowaniem. Twierdząc, że język ten jest niedostatecznie zdefiniowany (co było częściowo prawdą), firma ta opracowała własną kontrpropozycję: języki COMMERCIAL TRANSLATOR i AUTOCODER. Zważywszy, że w owym czasie 80% ogółu użytkowników maszyn było klientami IBM, sprawa przybrała poważny obrót.

Nie tylko zresztą IBM występował przeciwko COBOLowi. Firma NCR pozostawała przy języku NEAT (National Electronic Autocoding Techniques), zaś Honeywell zaproponował inny język do przetwarzania danych FACT (Fully Automatic Compiling Technique), który pod względem swoich możliwości niekiedy przewyższał COBOL.

W pierwszym okresie istnienia COBOLu (1960) jedynie firmy RCA i Remington Rand opracowały translatory dla tego języka.

Chcąc pokonać opór producentów rząd USA ogłosił w 1964 roku blokadę zakupów (dla rządowych instytucji) maszyn bez translatorów języka COBOL i ogłosił politykę konkursowych zakupów maszyn wyposażonych w COBOL (chodziło o zakup priorytetowy maszyn z najlepszymi translatorami). Ponieważ zakupy rządowe były dosyć znaczne, np. tylko w 1963 roku US Air Force zakupił 500 maszyn, taktyka ta przyniosła pełny sukces.

Podwójną grę w stosunku do COBOLu początkowo prowadziła

firma HONEYWELL. Mimo, iż przedstawiciele tej firmy wchodzili w skład zespołów roboczych COBOLu, utrzymywała ona w tajemnicy prace prowadzone nad językiem FACT. Pierwszy opis tego języka opublikowany na jesieni 1959 roku był kompletnym zaskoczeniem dla Komitetu COBOLu. Pod wpływem presji rządu firma ta zaakceptowała jednak COBOL, a nawet rozpoczęła prace nad rozwojem tego języka (dochodząc do bardzo dobrych translatorów). Język FACT został wdrożony jedynie na maszynie H 800.

Mimo początkowych trudności wdrożeniowych stosunkowo szybko przystąpiono do opracowywania różnego rodzaju języków automatycznego programowania (czy też raczej należałoby użyć określenia - kodowania) i w krótce doszło nawet do wyraźnej ich nadprodukcji. Języki tworzono bez dostatecznej podstawy teoretycznej i często do użytku ad hoc.

Na początku 1963 roku występowało około 290 języków, z tego:

a/ 74 zorientowanych maszynowo,

b/ 56 przestarzałych (wycofanych),

c/ 108 do obliczeń matematycznych,

d/ 16 do zastosowań administracyjno-ekonomicznych,

e/ 6 do zastosowań algebraiczno-ekonomicznych,

f/ 7 do symulacji procesów technologicznych.

W owym czasie istniało 39 translatorów języka COBOL.

W 1966 roku naliczono już 1200 języków automatycznego programowania, nie licząc podzbiorów ALGOLu, COBOLu itp. Pojawił się również uniwersalny język programowania PL/I.